Prototype COCOFIL3

Documents

1. Cahier des charges modifié le 29 juin
2. Plan de tests d'acceptation
3. Spécifications externes modifié le 29 juin
4. Plan de développement modifié le 29 juin
5. Plan qualité
6. Plan Test Système
7. Plan Test Unitaires
8. Dossier de conception globale modifié le 20 Aout
9. Dossier de conception détaillée modifié le 20 aout
10. Réalisation de tests système ajouté le 24 aout
11. Réalisation de tests d'acceptation ajouté le 24 aout
12. Bilan qualité ajouté le 28 aout
13. Dossier d'évaluation
14. Dossier d'installation ajouté le 22 aout (non diffusé)

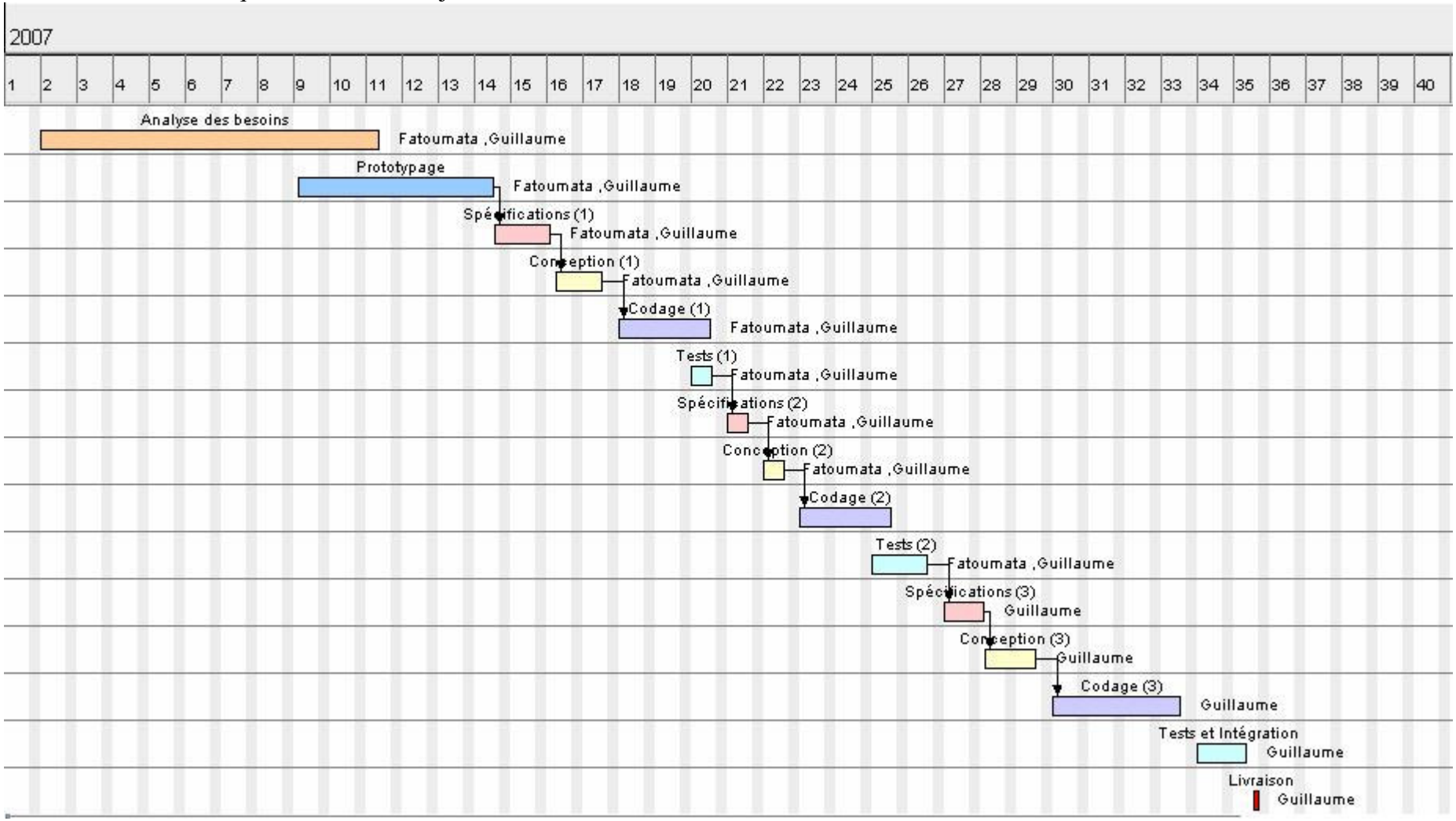
Implémentation

Incrément 1	
Fonctionnalité	Implementée
S'identifier	Oui
Fournir les données personnelles	Oui
Se déconnecter	Oui
Rechercher un film	Oui
Consulter les détails d'un film	Oui
Consulter ses recommandations	Oui
Consulter ses évaluations	Oui
Evaluer le film	Oui
Visualiser les communautés selon le critère « contenu »	Oui
Visualiser les communautés selon le critère « qualité »	Oui
Visualiser le profil d'un utilisateur	Oui
Passage à la nouvelle base de données	Oui

Incrément 2	
Fonctionnalité	Implementée
Visualiser son profil des communautés	Oui
Modifier ses données personnelles	Oui
Consulter la liste des contacts	Non
Ajouter un contact	Non
Supprimer un contact	Non
Bloquer un contact	Non
Consulter sa liste des favoris	Oui
Ajouter le film à sa liste des favoris	Oui
Supprimer le contact de sa liste des favoris	Oui

Planning Prévisionnel

Nom	Date de début	Date de fin	Durée
Analyse des besoins	08/01/07	15/03/07	48
Prototypage	27/02/07	06/04/07	28
Spécifications (1)	09/04/07	17/04/07	6
Conception (1)	18/04/07	27/04/07	7
Codage (1)	30/04/07	18/05/07	14
Tests (1)	14/05/07	18/05/07	4
Spécifications (2)	21/05/07	25/05/07	4
Conception (2)	28/05/07	01/06/07	4
Codage (2)	04/06/07	22/06/07	14
Tests (2)	18/06/07	29/06/07	9
Spécifications (3)	02/07/07	10/07/07	6
Conception (3)	11/07/07	19/07/07	6
Codage (3)	23/07/07	18/08/07	20
Tests et Intégration	20/08/07	30/08/07	8
Livraison	31/08/07	01/09/07	1


2007
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Analyse des besoins
Fatoumata ,Guillaume
Prototypage
Fatoumata ,Guillaume
Spécifications (1)
Fatoumata ,Guillaume
Conception (1)
Fatoumata ,Guillaume
Codage (1)
Fatoumata ,Guillaume
Tests (1)
Fatoumata ,Guillaume
Spécifications (2)
Fatoumata ,Guillaume
Conception (2)
Fatoumata ,Guillaume
Codage (2)
Tests (2)
Fatoumata ,Guillaume
Spécifications (3)
Guillaume
Conception (3)
Guillaume
Codage (3)
Guillaume
Tests et Intégration
Guillaume
Livraison
Guillaume

Planning Prévisionnel ajusté

GANTT project

Nom	Date de début	Date de fin	Durée
Analyse des besoins	08/01/07	15/03/07	48
phase d'apprentissage	27/02/07	06/04/07	28
Spécifications (1)	09/04/07	17/04/07	6
Apprentissage aux technologies JSF et AJAX4JSF	18/04/07	28/04/07	8
Conception (1)	30/04/07	11/05/07	9
Codage (1)	14/05/07	08/06/07	19
Tests(1)	11/06/07	15/06/07	4
Spécifications (2)	18/06/07	19/06/07	1
Conception (2)	20/06/07	22/06/07	2
Codage (2)	25/06/07	06/07/07	9
Tests (2)	02/07/07	06/07/07	4
Spécifications (3)	09/07/07	17/07/07	6
Conception (3)	18/07/07	27/07/07	7
Codage (3)	30/07/07	25/08/07	20
Tests et Intégration	20/08/07	30/08/07	8
Livraison	31/08/07	01/09/07	1

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Cahier des charges

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	12/06/2007
Version	2.2

Historique des versions

Version	Date	Modifications
1.0	06/02/2007	Début de rédaction
1.1	15/02/2007	Des modifications ont été apportées au plan du cahier des charges. Un nouveau format a été défini pour représenter les cas d'utilisation.
2.0	21/03/2007	• Modification suite aux remarques faites lors du 1er audit. • Modification suite aux remarques de Nathalie DENOS après lecture du document.
2.1	11/05/2007	Modification suite aux remarques faites lors du 2ème audit.
2.2	12/06/07	Modification suite aux remarques de Nathalie DENOS après relecture du document.

Sommaire

1 Introduction

1.1 But et portée du document

Le but du présent document est de présenter de façon précise et complète le logiciel à réaliser ainsi que ses différentes fonctionnalités.

Dans un premier temps, nous faisons une brève présentation du projet et son contexte afin de mieux cerner les objectifs du projet. Une description de l'existant vient ensuite approfondir les notions du domaine.

Puis une description des acteurs du système précède le détail des cas d'utilisation. Les cas d'utilisation sont passés en revue avec la description de leur sous cas.

Nous présentons ensuite les exigences non fonctionnelles de l'application.

Le cahier des charges, une fois validée par le client, servira de base à la spécification et la conception du logiciel à livrer. Ce document est donc destiné :

- à notre client : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

1.2 Équipe du projet

Membre	Statut/Laboratoire	Rôle dans le projet
ARGOUD Guillaume	Etudiant M2P GI, UFR IMA	Analyste programmeur Responsable qualité
BERRUT Catherine	Enseignant Chercheur (LIG - CLIPS)	Chef de projet
CAMARA Fatoumata Goundo	Etudiante M2P GI, UFR IMA	Analyste Programmeur Responsable développement
CUNIN Pierre-Yves	Enseignant Chercheur (LIG - LSR)	Consultant
DENOS Nathalie	Enseignant Chercheur (LIG - CLIPS)	Chef de projet

1.3 Présentation résumée du contexte et des objectifs du projet

Le projet APMD (http://apmd.prism.uvsq.fr, ACI Masses de données, projet MD-33 2004-2007) a donné lieu a des travaux de recherche portant sur l'accès à l'information notamment par le filtrage d'informations. Plus spécifiquement, il s'agit de personnalisation de l'accès à l'information en exploitant un profil utilisateur. Plusieurs prototypes ont permis d'évaluer, via des expériences « offline », certains des processus de filtrage collaboratif issus de ces travaux. En vue de procéder à des expériences impliquant des utilisateurs, on souhaite concevoir et développer une application Web exploitant ces processus : il s'agit d'un système de recommandation de films s'appuyant fortement sur la notion de communautés d'utilisateurs.
C'est cette application, appelée COCoFil3, qui fait l'objet du stage de M2 Pro Génie Informatique de Fatoumata Goundo Camara et Guillaume Argoud, de janvier à août 2007.
L'objectif de ce stage est de concevoir et développer une première version de cette application, qui permette de définir et valider des modalités d'interaction entre un utilisateur et un moteur de recommandations fondé sur le principe de communautés d'utilisateurs multidimensionnelles.
Nous présentons ci-dessous de façon plus précise les modules existants (appelés COCoFil2) et leur contexte d'application. Nous reformulons ensuite les objectifs résumés du projet à la lumière de cet existant.

1.3.1 Les modules existants dans COCoFil2

Le principe de recommandation fondé sur des communautés d'utilisateurs multidimensionnelles est appelé « espaces de communautés », et a été conçu par An-Te Nguyen dans sa thèse. Ces travaux de recherche ont donné lieu au développement de modules mettant en œuvre l'approche proposée en vue de l'évaluer dans un contexte académique. Ces modules ont permis de mener des expériences en laboratoire qui ont validé l'intérêt de l'approche. Mais ces modules ne permettent pas aujourd'hui de mener des expériences avec des utilisateurs, car toute l'interactivité avec le système n'a pas été conçue ni développée.
De façon plus précise, ces expériences en laboratoire exploitent un jeu de données appelé « MovieLens », qui a été collecté par l'équipe de recherche GroupLens (http://www.grouplens.org/) au cours de l'exploitation de leur système de recommandation de films. Ce jeu de données rassemble les « évaluations » (« ratings » en anglais) que de vrais utilisateurs ont faites sur des films qu'ils ont vus : ces évaluations consistent en une note traduisant à quel point le film leur a plu. Pour chaque évaluation, la date à laquelle elle a été faite est connue, et cela permet de concevoir des tests pour mesurer la qualité des recommandations produites par une méthode de recommandation à évaluer. En effet, on peut simuler une situation de recommandation à un instant t, et utiliser les évaluations ultérieures comme base de référence pour comparer les recommandations issues de la méthode en question.
Ainsi les processus existants, ont été conçus pour être lancés « offline » sur un jeu de données de test, et pour produire des résultats sous forme tabulée qui sont ensuite analysés pour évaluer les performances de l'approche proposée.

1.3.2 Les objectifs du projet au vu de l'existant

L'objectif premier du projet est donc de mettre en place une application Web de recommandation de films permettant une utilisation interactive du système par des utilisateurs réels. Le système devra intégrer l'approche des espaces de communautés, et adapter au mieux les processus déjà développés en vue de les intégrer dans le noyau de cette application interactive.

Un second objectif, moins prioritaire, est de permettre d'accéder à un jeu de données plus complet quant à la description des films : la base de données IMDB (http://www.imdb.com/interfaces) doit permettre cela. A terme, et au-delà du cadre du présent projet, cette nouvelle possibilité permettra de concevoir de nouveaux espaces de communautés et d'évaluer leur apport dans la qualité des recommandations. En concevant le système, il faudra penser à faciliter une évolution aisée vers une version intégrant les données de IMDB.

2 Présentation de l'existant

Dans cette partie, nous présentons l'existant selon les étapes suivantes :

- les systèmes de recommandation, que nous définissons au travers de la terminologie du domaine et d'une série d'exemples
- les grandes lignes de l'approche des « espaces de communautés » de An-Te Nguyen
- les modules de COCoFil2, leurs caractéristiques techniques et leurs fonctionnalités

Nous concluons cette étude en resituant l'objectif du projet par rapport aux systèmes existants.

2.1 Systèmes de recommandation

2.1.1 Glossaire

Ce glossaire introduit les termes et les concepts propres au domaine dans lequel on travaille : les systèmes de recommandation.
Le problème de surcharge d'information est un réel problème auquel on est tous confronté.
Au lieu de laisser l'utilisateur dépenser son temps à chercher l'information dont il a besoin, les mécanismes de recommandation lui font parvenir des informations pertinentes sélectionnées parmi un grand nombre de données.

L'objectif principal d'un système de filtrage d'information, ou système de recommandation (***Recommender System***), est de filtrer un flux entrant d'informations de façon personnalisée pour chaque utilisateur, tout en s'adaptant en permanence à son besoin d'information.
[An-Te Nguyen, 2006].

Les domaines d'application du filtrage d'information sont nombreux : la recherche documentaire, le commerce électronique, les loisirs (films, musique,…) etc. Il existe plusieurs techniques de filtrage d'information : le filtrage d'information basé sur le contenu (***Content- based filtering***), le filtrage collaboratif (***Collaborative filtering***).
La liste des documents envoyés à un utilisateur par un système de recommandation est constituée à partir de son profil.

Profil similaire : Deux utilisateurs ont un profil similaire lorsqu'il en commun des documents dont l'évaluation diffère peu.

Filtrage basé sur le contenu : Le filtrage basé sur le contenu s'appuie sur le contenu des documents (thèmes abordés) pour les comparer à un profil lui-même constitué de thèmes.

[An-Te Nguyen]. Les moteurs de recherche sont une application du filtrage basé sur le contenu.

Filtrage collaboratif : Le filtrage collaboratif a pour principe d'exploiter les évaluations que des utilisateurs ont faites, afin de recommander ces mêmes documents à d'autres utilisateurs, et sans qu'il soit nécessaire d'analyser le contenu des documents. ***[An-Te Nguyen].*** Si deux utilisateurs A et B ont évalué un certain nombre de documents de façon similaire, il y a de forte chance que A aime ce que B aime, et inversement. Donc les documents que A a aimés peuvent être recommandés à B (et inversement).

Filtrage collaboratif actif : L'utilisateur fait parvenir des documents à d'autres personnes qu'il connaît. Le filtrage collaboratif actif représente une solution au problème du démarrage à froid.

Démarrage à froid : Ces termes signifient que pendant les premiers temps d'utilisation, le système ne peut faire de recoupements entre profils car ceux-ci sont disjoints ou vides.

Communauté : Une communauté est un ensemble d'utilisateurs qui ont un profil similaire.
Dans le contexte d'un système de filtrage collaboratif, les « communautés » des utilisateurs sont généralement formées par la proximité des évaluations qu'ils ont faites des informations qui leur ont été présentées. ***[An-Te Nguyen].***
La proximité des évaluations n'est pas le seul critère de formation des communautés, elles peuvent être formées selon d'autres critères dépendant du domaine. Nous pouvons imaginer des critères comme : la situation géographique, l'âge ou même les centres d'intérêt des utilisateurs.

2.1.2 Les systèmes de recommandation existants

De nombreux systèmes de recommandation existent. Ces systèmes utilisent aussi bien le filtrage basé sur le contenu que le filtrage collaboratif.
Les systèmes de vente en ligne et de musique en ligne constituent des applications du filtrage basé sur le contenu.
Parmi les systèmes basés sur le filtrage collaboratif, nous pouvons citer : Movielens (recommandations de film), U-lik (recommandations en tout genre : musique, voyage, etc.).

2.1.3 Confidentialité et anonymat : Un aspect important

La confidentialité est un facteur important dans un système de recommandation. Il faut garantir l'anonymat pour les utilisateurs qui ne souhaitent pas être connus de tous. Ce choix est laissé à l'internaute lors de la configuration du système dans le processus d'inscription (section 5.4.3). Dans COCoFil3, nous souhaitons proposer deux niveaux d'anonymat :

- L'anonymat total : le profil et toutes les évaluations de l'utilisateur sont anonymes.
- L'anonymat des évaluations : toutes les évaluations de l'utilisateur sont anonymes. Les autres utilisateurs du système n'ont pas la possibilité de visualiser la liste des films qu'il a évalués.

2.1.4 Limites des systèmes de recommandation actuels

Les systèmes de filtrage collaboratif souffrent du démarrage à froid. Lorsqu'un nouvel utilisateur arrive, le système est incapable de lui fournir des recommandations vu qu'il ne dispose d'aucune information sur les centres d'intérêt de ce dernier.

1. Le filtrage actif constitue un moyen de résoudre ce problème ; mais cela ne fonctionne que dans un système où les utilisateurs se connaissent entre eux.
2. Un autre moyen pour contourner le problème du démarrage à froid consiste à forcer l'utilisateur à faire des évaluations dès les premiers jours d'utilisation du système.

Ces deux techniques permettent de mettre en place le profil de l'utilisateur qui s'enrichira au cours du temps grâce aux évaluations produites par l'utilisateur.
MovieLens exige d'un nouvel utilisateur qu'il évalue au moins 15 films avant de pouvoir recevoir des recommandations.

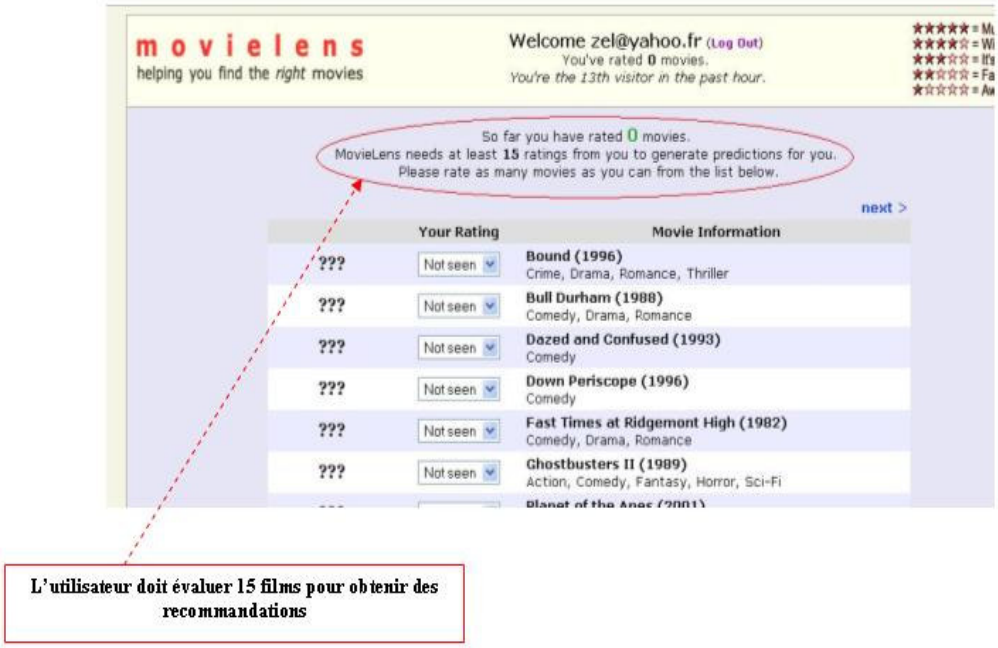


Figure 1: Première utilisation de MovieLens par l'utilisateur zel@yahoo.fr

Le noyau d'un système de filtrage collaboratif repose sur la formation des communautés. Pour calculer des recommandations pour un utilisateur donné, le système détermine tout d'abord sa communauté puis lui recommande les documents appréciés par sa communauté. De façon traditionnelle, les communautés sont calculées à partir de la proximité des évaluations. Les membres d'une même communauté ont un certain nombre d'évaluations en commun.
Par exemple, dans la figure ci-dessous, les utilisateurs sont divisés en deux communautés C1 et C2. On peut remarquer que les utilisateurs A et B ont évalué le document D2 de la même façon. Le système a donc recommandé à B le document D1, que l'utilisateur A apprécie.

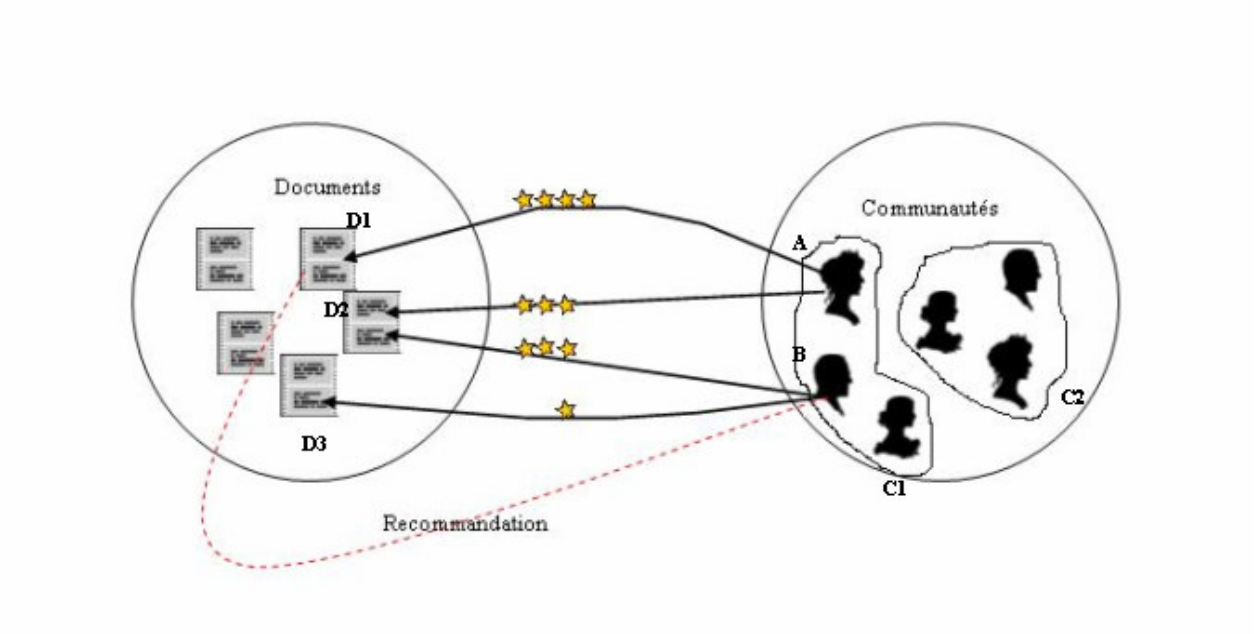


Figure 2: Principe de fonctionnement d'un système de filtrage collaboratif

Dans les systèmes existants, cette notion de communauté reste un concept interne au système. Ils n'offrent pas la possibilité à l'utilisateur de percevoir les communautés autour de lui. Pourtant, la perception des communautés peut jouer un rôle important. Elle offrirait aux l'utilisateur une vision globale sur les autres participants, en outre elle leur permettrait d'explorer les autres communautés en vue de connaître certaines informations tels que les films les plus regardés par une communauté, les dernières recommandations fournies à une communauté, etc. Par ailleurs, en prenant connaissance de certaines informations sur les communautés, l'utilisateur pourrait changer de communauté pour une autre qui lui semble plus intéressante.

2.2 Travaux de l'équipe MRIM au sein du laboratoire LIG

L'importance de la perception des communautés dans un système de filtrage collaboratif a conduit l'équipe MRIM à mener des travaux autour de ce concept. Parmi ces travaux, nous pouvons citer : le projet TIPS et la thèse de An-Te Nguyen intitulée « ***COCoFil2 : Un nouveau système de filtrage collaboratif basé sur le modèle des espaces de communautés* ».**

2.2.1 Le projet TIPS

Le projet TIPS a conduit à la réalisation d'un système de recommandation « ***COCoFil*** » basé sur le filtrage collaboratif à propos d'articles scientifiques. En plus des fonctionnalités offertes par un système de recommandation classique, le système fournit d'autres fonctionnalités spécifiquement orientées vers la notion de communauté :

- Le carnet d'adresses : Il contient l'ensemble des chercheurs qui « intéressent » l'utilisateur c'est à dire les personnes qui sont connues de l'utilisateur, où celles qui ont un profil proche du sien. Dans la gestion de son carnet d'adresses, l'utilisateur peut ajouter et supprimer des contacts.

- Perception des autres : Possibilité pour l'utilisateur de consulter la liste des personnes qui ont un profil similaire au sien. A partir de cette liste, l'utilisateur a la possibilité d'ajouter à son carnet d'adresses les personnes qui l'intéressent.

2.2.2 Le projet TIPS

Critère « contenu » : Dans le cadre du projet COCoFil3 qui est un système de recommandation sur les films, « contenu » représente le genre de film (aventure, comédie, drame, policier, etc.). Les recommandations produites selon le critère « contenu » sont calculées à partir du genre préféré des utilisateurs.

Critère « qualité » : Le critère « qualité » représente les évaluations. Les recommandations produites selon le critère « qualité » sont calculées à partir du rapprochement des évaluations des utilisateurs.

Utilisateur moyen : Utilisateur virtuel dont le profil correspond à une moyenne des profils des autres utilisateurs d'une communauté donnée.

2.2.3 COCoFil2

Les travaux de la thèse partent sur la plate forme développée dans le cadre du projet TIPS. Le système développé au cours de ce projet n'exploite que partiellement la notion de communauté. L'utilisateur ne peut percevoir que ses semblables pris un par un, il n'a pas une vision globale de tous les utilisateurs du système. Au cours de ces travaux de recherche, An-Te Nguyen travailla sur la perception globale des communautés dans un système de recommandation en mettant en place un système de recommandation de films appelé COCoFil2.
Dans COCoFil2, l'utilisateur a la possibilité de visualiser toutes les communautés du système.
Même si les communautés sont présentes de façon interne dans les systèmes de recommandation existants, leur formation est basée sur le seul critère des évaluations. COCoFil2 offre la possibilité à l'utilisateur de visualiser les communautés selon plusieurs critères : « contenu », « qualité », âge, genre de film préféré, évaluations, profession, situation géographique, etc.
Outre, la perception des communautés, il intègre des algorithmes qui permettent de contourner le démarrage à froid. Ces algorithmes reposent sur des règles de décision. Ainsi lorsqu'un nouvel utilisateur arrive, il n'a ni besoin d'évaluer un certain nombre de documents comme dans MovieLens, ni besoin de se faire recommander des documents par d'autres utilisateurs (filtrage actif). A partir de l'âge, la profession, la situation géographique de l'utilisateur, COCoFil2 est capable d'induire les communautés auxquelles il appartient.

En conclusion, les interactions entre l'utilisateur et le système sont plus intéressantes dans COCoFil2. L'utilisateur explore l'ensemble des communautés d'utilisateur du système selon divers critères, comprend a quoi servent ses évaluations et d'où viennent ses recommandations. Il ne passe plus par des évaluations forcées, et n'est pas obligé de décrire son besoin pour construire son profil, ce qui constitue souvent des tâches pénibles et complexes pour les utilisateurs.

2.3 Modules de COCoFil2

COCoFil2 est composé de plusieurs modules: extraction des profils, calcul des communautés, projection dur un plan 2D, calcul des prédictions.

- Le module d'extraction des profils extrait les profils utilisateurs et les informations nécessaires dans une base de données afin de les fournir aux modules de calcul des communautés et des prédictions.
- Le module de calcul des communautés construit les espaces de communautés selon les critères disponibles dans les profils d'utilisateurs pour alimenter la table de communautés. Parmi les critères pris en compte dans COCoFil2, nous pouvons citer : la qualité, le contenu (voir 2.3.1), l'âge, la profession et la situation géographique.
- Le module de projection sur un plan 2D complète le deuxième module pour améliorer la table de communautés. En effet, il se sert des résultats du calcul des communautés pour créer les cartes 2D selon les deux critères de qualité et de contenu.
- Le module de calcul des prédictions produit les recommandations en prenant en compte l'opinion des communautés multicritères présentes dans la table de communautés. Les recommandations générées pour un utilisateur à partir de ses propres communautés peuvent être exploitées séparément pour chaque critère de formation ou de façon combinée selon une stratégie d'hybridation particulière. Ce module gère également la production de recommandations pour les nouveaux utilisateurs (démarrage à froid).

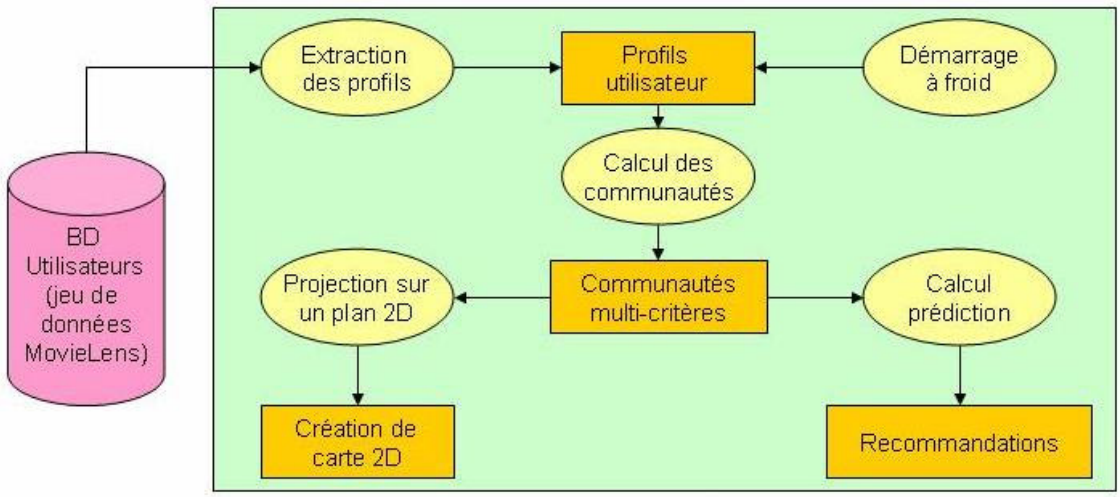


Figure 3: Modules de COCoFil2

2.4 Synthèse

L'application doit offrir les fonctionnalités classiques d'un système de recommandations, c'est-à-dire que l'utilisateur peut s'inscrire et définir son profil, il peut consulter les fiches des films et évaluer ceux qu'il a vus, et il peut consulter les recommandations que le système produit.

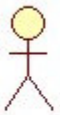
L'utilisation des bases de données de IMDB permettra de fournir à l'utilisateur toutes les informations nécessaires sur un film : distribution (acteurs et actrices jouant dans le film), metteur en scène, résumé, date de sortie (dans les salles et en DVD), etc. La mise en avant des communautés devra également être un élément essentiel de l'application, fournissant à l'utilisateur un moyen de se situer par rapport aux autres utilisateurs.

L'utilisation de cartes 2D servira de support à la visualisation des communautés. Pour renforcer cet aspect communautaire, l'utilisateur aura à sa disposition un carnet d'adresse à travers lequel il pourra tenir à jour une liste de contact. D'autres fonctionnalités, comme la gestion d'une liste de films « favoris » ou encore la possibilité de commenter les films, viennent renforcer la notion de profil et sa personnalisation.
Une fois l'application mise en place, une étude des performances des fonctionnalités du noyau permettra d'évaluer les éventuels problèmes rencontrés lors de la montée en charge de l'application. Suite à cette étude, des solutions seront élaborées afin d'éventuellement remédier aux problèmes rencontrés.
Nous détaillons dans la suite l'ensemble des fonctionnalités de l'application.

3 Acteurs du système

Membre

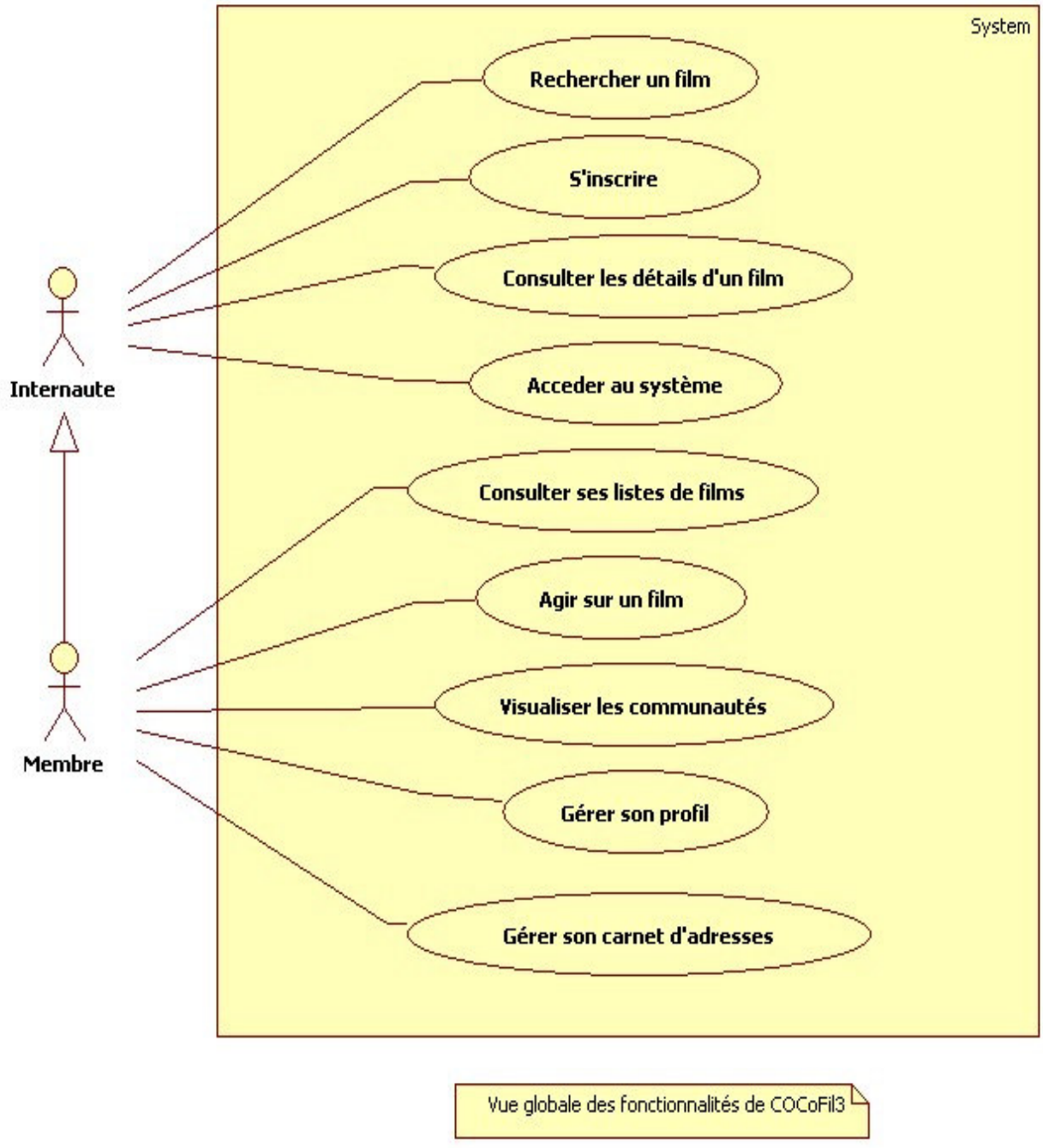
Un membre est un utilisateur connecté au système avec ses identifiants.
Il a accès à toutes les fonctionnalités.



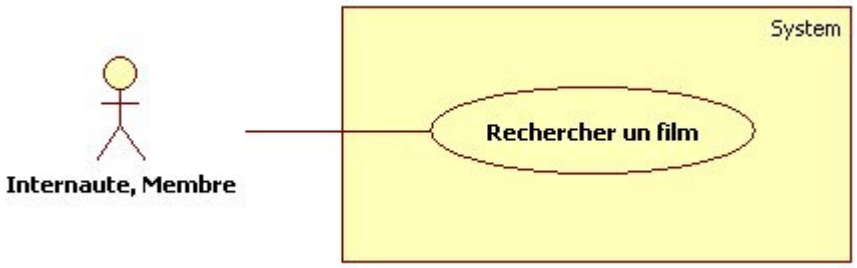
Internaute

Un internaute est une personne qui utilise le système.
Il a un accès limité aux fonctionnalités du système; Seules les fonctionnalités de recherche, de consultation de détails sur les films et d'inscription dans le système lui sont accessibles.

4 Cas d'utilisation

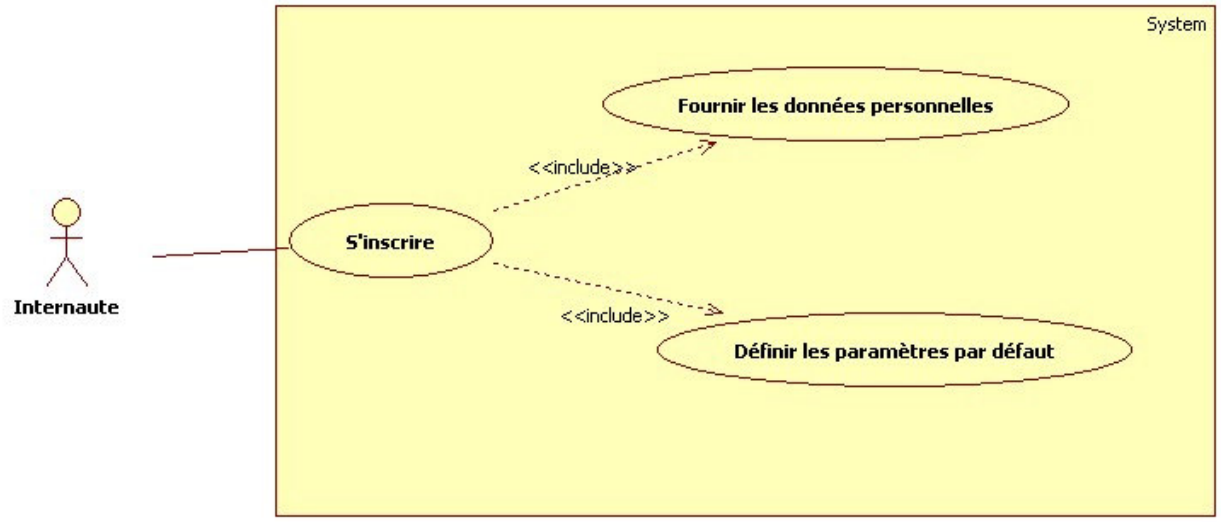


Cas 1 : Rechercher un film



Acteur : Internaute, Membre
Contexte : L'utilisateur peut à tout moment accéder à la fonctionnalité « recherche » afin de trouver un film.
Cas : Un système de recommandation a pour but de fournir à l'utilisateur des articles (films, musique, documents scientifiques…) intéressants pour lui. En effet, les systèmes de recommandation représentent un moyen pour l'utilisateur de trouver facilement l'information pertinente. Donc la fonctionnalité de recherche ne représente pas une fonctionnalité majeure de notre système et pourrait au contraire aller à l'encontre d'un tel système. Néanmoins, cette fonctionnalité pourrait servir à rechercher un film qui n'a pas été recommandé par le système afin de l'évaluer ; elle sera donc réalisée de façon très basique (par mots-clés sur le titre, le nom du réalisateur, ou le nom de l'un des acteurs du film par exemple).
Lorsque l'utilisateur effectue une recherche, une liste de films correspondant à sa est affichée.
Importance : Important

Cas 2 : S'inscrire



Acteur : Internaute
Contexte : L'internaute ne doit pas être déjà membre.
Cas : Cette fonctionnalité permet à un internaute de s'enregistrer auprès du système. A la fin du processus d'inscription, l'internaute peut désormais se connecter au système à tout moment avec le nom d'utilisateur et le mot de passe qu'il a choisis.
Importance : Très important.
Sous cas : L'inscription se fait en deux étapes. Ces deux étapes correspondent aux sous cas :

- Fournir les données personnelles (cas 2.1)
- Définir les paramètres par défaut (cas 2.2)

Cas 2.1 : Fournir les données personnelles

Acteur : Internaute
Contexte : L'utilisateur ne peut choisir un nom d'utilisateur qui existe déjà.
Cas : Dans cette première étape de l'inscription, l'utilisateur choisit son nom d'utilisateur et son mot de passe puis il fournit son adresse (code postal de 5 chiffres), sa profession dans une liste prédéfinie et son âge. Ces données sont utilisées dans le processus de démarrage à froid de COCoFil2 pour produire les premières recommandations à l'utilisateur.
Par la suite, il peut modifier l'ensemble de ses données excepté le nom d'utilisateur.
Importance : Très important.

Cas 2.2 : Définir les paramètres par défaut

Acteur : Internaute
Contexte : L'utilisateur doit avoir passé avec succès la première étape de l'inscription.
Cas : Cette deuxième étape de l'inscription permet à l'utilisateur de choisir les paramètres par défaut afin de configurer le système pour sa propre utilisation. L'internaute peut sauter cette étape de configuration, dans ce cas, les valeurs par défaut sont prises en compte. Selon les fonctionnalités qui seront implémentées, l'utilisateur aura la possibilité de paramétrer le système par rapport à :

- L'anonymat : l'internaute peut choisir l'anonymat total ou anonymat des évaluations. Par défaut, le système considère que l'utilisateur ne souhaite être anonyme ni pour les évaluations ni totalement.

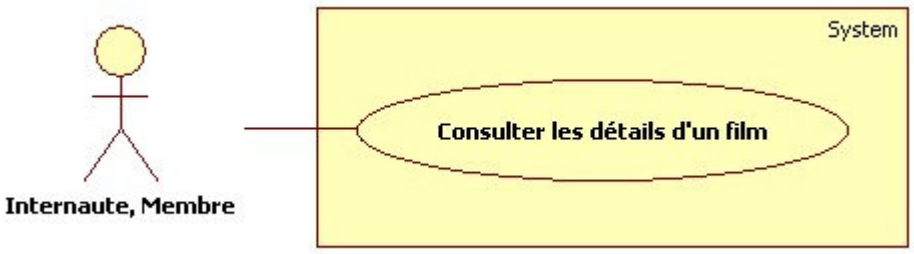
- Le carnet d'adresses : dans le carnet d'adresses, l'internaute peut choisir les informations de ses contacts qu'il souhaite visualiser (Par ex, nom, prénom, adresse email, …). Par défaut, le système affiche le nom d'utilisateur.

- La confidentialité : Il peut choisir ses données personnelles qui seront visibles par les autres utilisateurs. Par défaut le système affiche le nom d'utilisateur et l'adresse email.

- L'affichage des films : l'utilisateur ne peut configurer les informations qu'il souhaite visualiser lorsqu'un film s'affiche. Seul le titre des films est affiché.

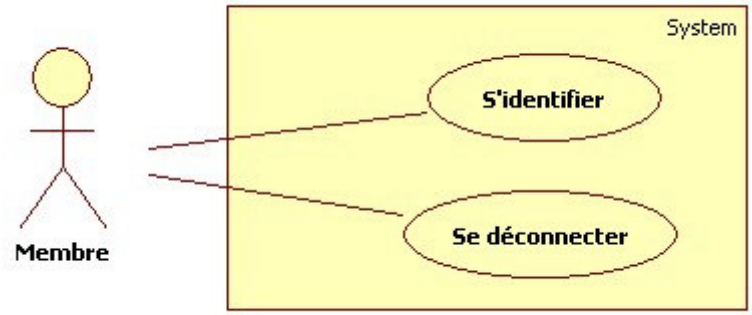
Importance : Peu important.

Cas 3 : Consulter les détails d'un film



Acteur : Internaute, Membre
Contexte : L'utilisateur a obtenu une liste de films suite à une recherche (ou pour un membre suite à l'affichage de ses favoris ou de ses recommandations, etc.).
Cas : La sélection d'un film entraîne l'ouverture d'une fiche contenant des informations sur ce dernier : titre, année, producteur, acteur, description, etc. Si l'utilisateur est membre, il peut également agir sur le film comme détaillé dans le cas 6.
Importance : Très important.

Cas 4 : Accéder au système



Acteur : Membre
Contexte : L'utilisateur doit être préalablement inscrit.
Cas : Afin d'accéder à ses données personnelles (profil, recommandations, carnet d'adresses, favoris, etc.), l'utilisateur doit se connecter au système. Une fois authentifié auprès du système, sa session débute.
L'utilisateur peut mettre fin à sa session en se déconnectant, ou en fermant le navigateur.
Importance : Très important
Sous cas : L'accès au système se décompose en deux sous cas :

- S'identifier (cas 4.1)
- Se déconnecter (cas 4.2)

Cas 4.1 : S'identifier

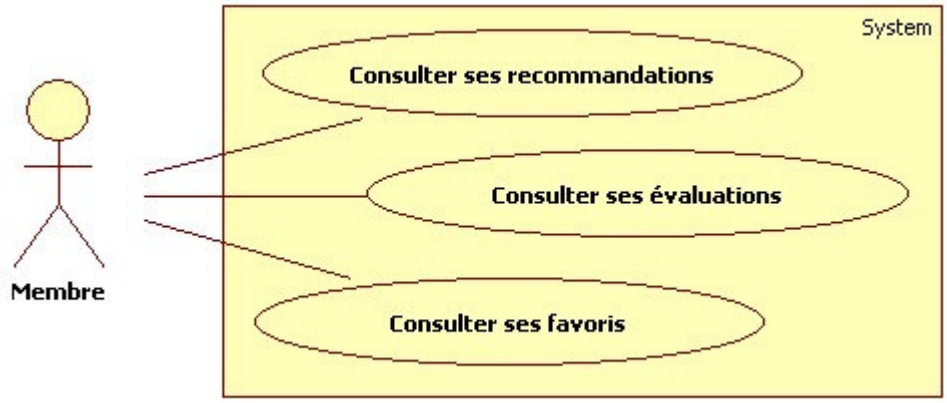
Acteur : Membre
Contexte : L'utilisateur n'a aucune session en cours.
Cas : L'utilisateur peut à tout moment se connecter au système en s'identifiant avec le nom d'utilisateur et le mot de passe choisis pendant l'inscription. Il est alors reconnu par le système comme membre et peut accéder à toutes ses données personnelles. Si

l'utilisateur oublie son mot de passe, il a la possibilité d'en obtenir un nouveau en spécifiant son nom d'utilisateur ; alors un message lui est envoyé avec son mot de passe.
Importance : Très important

Cas 4.2 : Se déconnecter

Acteur : Membre
Contexte : L'utilisateur est en cours de session (il s'est identifié).
Cas : L'utilisateur peut se déconnecter à tout moment. Il met fin à sa session en cours.
Importance : Très important

Cas 5 : Consulter ses listes de film



Acteur : Membre
Contexte : L'utilisateur vient de demander l'affichage d'une liste de films
Cas : L'utilisateur consulte sous forme d'une liste de films ses recommandations, ses évaluations et ses favoris. A partir de cette liste, il a la possibilité d'agir sur les films (voir cas 6).
Importance : Très important.
Sous cas : La consultation des fiches de films se décompose en trois sous cas :

- Consulter ses recommandations (cas 5.1)
- Consulter ses évaluations (cas 5.2)
- Consulter ses favoris (cas 5.3)

Cas 5.1 : Consulter ses recommandations

Acteur : Membre
Contexte : L'utilisateur vient de demander l'affichage de ses recommandations
Cas : Lorsque le système produit des recommandations pour l'utilisateur, ce dernier peut les consulter sous forme de liste de films. L'utilisateur a le choix de voir uniquement les dernières recommandations ou toutes les recommandations qui lui ont été faites.
Importance : Très important.

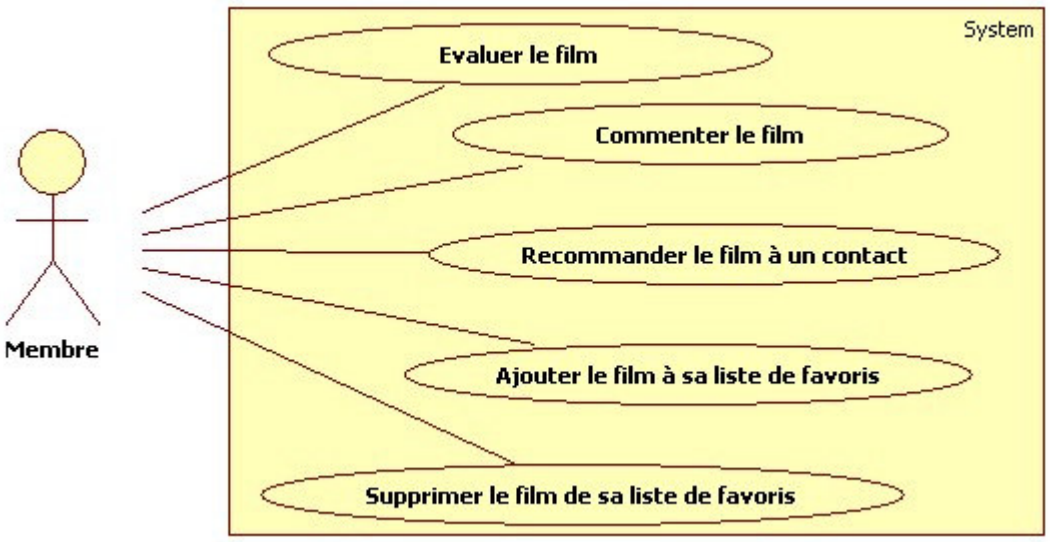
Cas 5.2 : Consulter ses évaluations

Acteur : Membre
Contexte : L'utilisateur vient de demander l'affichage de ses évaluations
Cas : L'utilisateur peut consulter l'historique de ses évaluations sous forme de liste de films.
Importance : Très important.

Cas 5.3 : Consulter sa liste de favoris

Acteur : Membre
Contexte : L'utilisateur vient de demander l'affichage de ses favoris.
Cas : Une liste de films contenant tous les favoris de l'utilisateur est affichée.
Importance : Peu important.

Cas 6 : Agir sur un film



Acteur : Membre
Contexte : L'utilisateur est en train de consulter le résultat d'une recherche, une liste de films ou les détails d'un film.
Cas : L'utilisateur a la possibilité d'agir sur un film.
Sous cas : Les différentes actions que l'utilisateur peut effectuer sur un film sont détaillées dans cinq sous cas :

- Evaluer le film (cas 6.1)
- Commenter le film (cas 6.2)
- Recommander le film à un contact (cas 6.3)
- Ajouter le film à sa liste de favoris (cas 6.4)
- Supprimer un film de sa liste de favoris (cas 6.5)

Cas 6.1 : Evaluer le film

Acteur : Membre
Contexte : L'utilisateur est en train de consulter le résultat d'une recherche, une liste de films ou les détails d'un film.
Cas : L'utilisateur peut évaluer un film. L'évaluation d'un film consiste à lui donner une note entre 1 et 5 dont la sémantique est la suivante :

- 1 = Médiocre
- 2 = Passable
- 3 = Assez - Bien
- 4 = Bien
- 5 = Excellent

Cette évaluation va ensuite être prise en compte pour les recommandations futures faites à l'utilisateur. L'utilisateur peut revenir quand il le veut sur l'évaluation d'un film, par exemple parce qu'il la revu ou encore parce que ses goûts ont évolués depuis la dernière fois.
Importance : Très important

Cas 6.2 : Commenter le film

Acteur : Membre
Contexte : L'utilisateur est en train de consulter le résultat d'une recherche, une liste de films ou les détails d'un film. Afin de pouvoir commenter un film, l'utilisateur doit l'avoir évalué.
Cas : L'utilisateur a la possibilité de s'exprimer sur son appréciation d'un film en laissant un commentaire. Laisser un commentaire permet de donner plus de précision sur le film aidant ainsi les autres utilisateurs à se faire une meilleure idée du film. Les commentaires d'un film sont accessibles à partir de sa fiche.
Importance : Peu important.

Cas 6.3 : Recommander le film à un contact

Acteur : Membre
Contexte : L'utilisateur est en train de consulter le résultat d'une recherche, une liste de films ou les détails d'un film. Le contact à qui il veut recommander le film doit figurer dans son carnet d'adresse.
Cas : Si l'utilisateur pense que le film sera apprécié par un de ses contacts, il peut le lui recommander directement (filtrage actif). Le système ajoute alors le film à la liste des recommandations du contact visé.
Importance : Peu important.

Cas 6.4 : Ajouter le film à sa liste de favoris

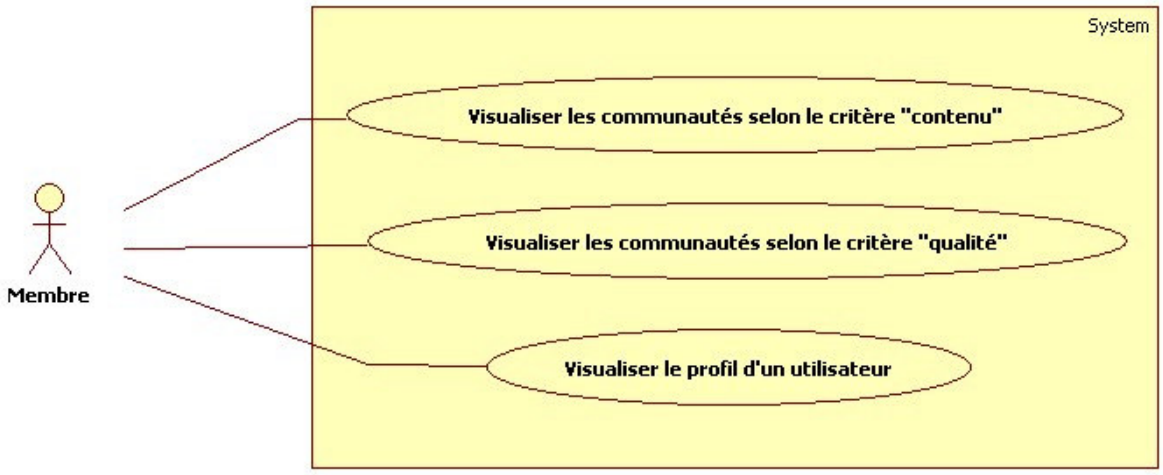
Acteur : Membre
Contexte : L'utilisateur est en train de consulter le résultat d'une recherche, une liste de films ou les détails d'un film.
Cas : Si l'utilisateur est intéressé par un film, il peut l'ajouter à sa liste de favoris. La liste de favoris est simplement une liste de films vide à l'inscription de l'utilisateur et dans laquelle il peut par la suite ajouter un film qu'il rencontre. Cela lui permet de se souvenir d'un film en attendant de le voir plus tard et de le retrouver plus facilement

pour l'évaluer, ajouter un commentaire, etc. Un film ne peut être présent qu'une seule fois dans la liste des favoris.
Importance : Peu important.

Cas 6.5 : Supprimer le film de sa liste des favoris

Acteur : Membre
Contexte : Le film qu'il souhaite supprimer doit exister dans sa liste des favoris.
Cas : L'utilisateur supprime un film de sa liste des favoris. Le film disparaît de la liste des favoris mais l'utilisateur a la possibilité de l'ajouter à nouveau (cf. cas 6.4).
Importance : Peu important.

Cas 7 : Visualiser les communautés



Acteur : Membre
Contexte : L'utilisateur vient de demander l'affichage de son profil.
Cas : L'utilisateur a la possibilité de visualiser toutes les communautés selon les deux critères de « contenu » et « qualité ».
Importance : Très important.
Sous-cas : La visualisation des communautés se décompose en 3 sous cas :

- Visualiser les communautés selon le critère « contenu » (cas 7.1)
- Visualiser les communautés selon le critère « qualité » (cas 7.2)
- Visualiser le profil d'un utilisateur (cas 7.3)

Cas 7.1 : Visualiser les communautés selon le critère « contenu »

Acteur : Membre
Contexte : L'utilisateur vient de demander l'affichage de son profil selon le critère « contenu ».
Cas : L'utilisateur peut visualiser toutes communautés selon le critère « contenu ». Les communautés sont représentées sur une carte 2D. Sa position et celle de l'utilisateur moyen de chaque communauté lui sont indiquées. Une fiche descriptive de chaque utilisateur est accessible, sauf pour les utilisateurs de niveau d'anonymat total ; elle contient les informations que l'utilisateur cible n'a pas filtrées. En visualisant la carte

des communautés, l'utilisateur peut aussi repérer tous les utilisateurs qui appartiennent à son carnet d'adresses (cf. cas 9).
Importance : Très important.

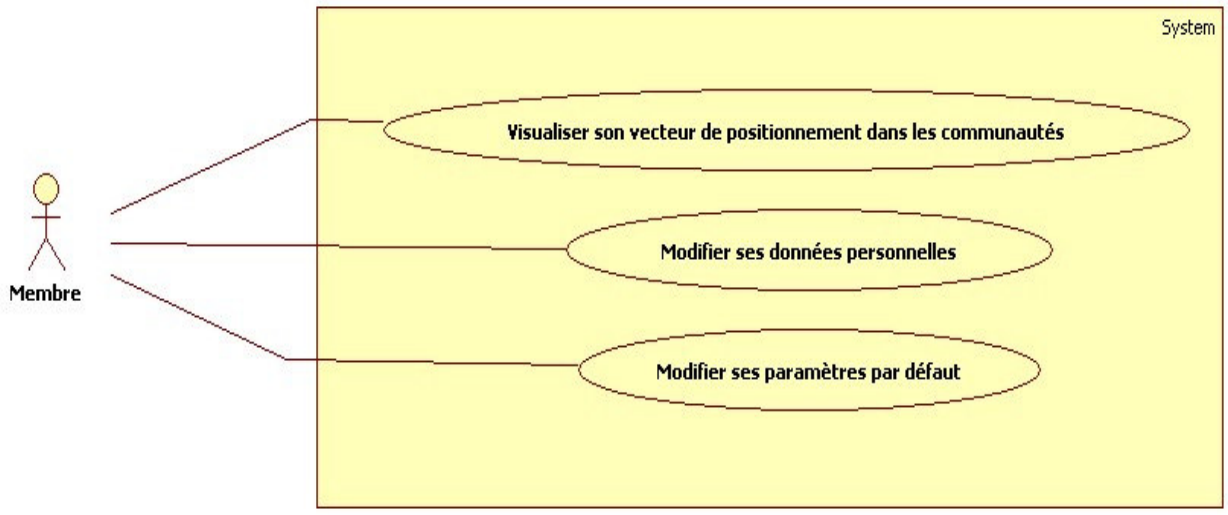
Cas 7.2 : Visualiser les communautés selon le critère « qualité »

Acteur : Membre
Contexte : L'utilisateur vient de demander l'affichage de son profil selon le critère « contenu ».
Cas : L'utilisateur peut visualiser toutes les communautés selon le critère « qualité ». Les communautés sont représentées sur une carte 2D. Sa position et celle de l'utilisateur moyen de chaque communauté lui sont indiquées. Une fiche descriptive de chaque utilisateur est accessible, sauf pour les utilisateurs de niveau d'anonymat total ; elle contient les informations que l'utilisateur cible n'a pas filtrée. En visualisant la carte des communautés, l'utilisateur peut aussi repérer tous les utilisateurs qui appartiennent à son carnet d'adresses (cf. cas 9).
Importance : Très important.

Cas 7.3 : Visualiser le profil d'un utilisateur

Acteur : Membre
Contexte : L'utilisateur visualise une carte des communautés.
Cas : Lorsque l'utilisateur observe une carte des communautés, il peut agir dessus en visualisant les profils d'autres utilisateurs. Il peut prendre connaissance des informations personnelles de n'importe quel utilisateur y compris lui même (à condition que cet utilisateur ne soit pas anonyme), visualiser la liste des films qu'il a évalués, la liste des films qui lui ont été recommandés mais aussi l'ajouter à son carnet d'adresses s'il le désire (cas 9.1).
Importance : Très important.

Cas 8 : Gérer son profil



Acteur : Membre
Contexte : L'utilisateur membre doit être connecté au système.
Cas : Cette fonctionnalité permet à l'utilisateur de gérer toutes les données qui lui sont propres.
Importance : Très important.

Sous cas : La gestion du profil se décompose en trois sous cas :

- Visualiser son profil des communautés (cas 8.1)
- Modifier ses données personnelles (cas 8.2)
- Modifier ses paramètres par défaut (cas 8.3)

Cas 8.1 : Visualiser son vecteur de positionnement dans les communautés

Acteur : Membre
Contexte : L'utilisateur doit être connecté au système.
Cas : Cette fonctionnalité permet à l'utilisateur d'avoir une vision globale des communautés auxquelles il appartient, c'est-à-dire, pour chaque critère de formation des communautés (contenu, qualité, âge, profession, situation géographique), on lui indique son profil. Par exemple, lors de la visualisation de son profil des communautés l'utilisateur Zel prend connaissance des informations suivantes :

Critère	Contenu	Qualité	Age	Profession	Situation géographique
Communauté	Comédie	Communauté 1	Ado	Etudiant	Rhône Alpes

En plus de la visualisation globale des communautés, on fournit aussi à l'utilisateur les informations sur son appréciation des genres de films. Là aussi, l'utilisateur aura par genre une appréciation sous forme de pourcentage.
Importance : Important.

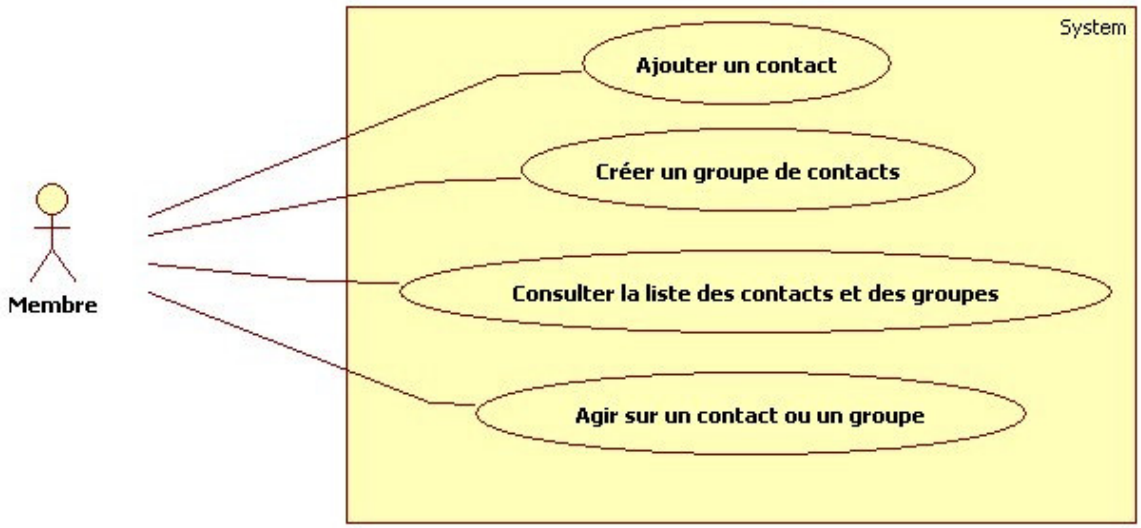
Cas 8.2 : Modifier ses données personnelles

Acteur : Membre
Contexte : L'utilisateur doit être connecté au système.
Cas : S'il le désire, l'utilisateur peut modifier une ou plusieurs de ses données personnelles, excepté le nom d'utilisateur.
Importance : Important.

Cas 8.3 : Modifier ses paramètres par défaut

Acteur : Membre
Contexte : L'utilisateur doit être connecté au système.
Cas : L'utilisateur a la possibilité de modifier ses paramètres par défaut afin de changer son niveau d'anonymat, les informations qu'il souhaite visualiser sur un contact et ses informations qui seront accessibles aux autres utilisateurs.
Importance : Peu important.

Cas 9 : Gérer son carnet d'adresses



Acteur : Membre
Contexte : L'utilisateur doit être connecté au système.
Cas : Le carnet d'adresses contient les contacts de l'utilisateur. Le carnet d'adresse d'un nouvel utilisateur est vide, c'est à lui de le remplir en y ajoutant les autres utilisateurs du système qui l'intéresse, avec qui il souhaite interagir.
Importance : Important.
Sous cas : Cette fonctionnalité se décompose en 4 sous cas :

- Ajouter un contact (cas 9.1)
- Créer un groupe de contacts (cas 9.2)
- Consulter la liste des contacts et des groupes (9.3)
- Agir sur un contact ou un groupe (cas 9.4)

Cas 9.1 : Ajouter un contact

Acteur : Membre
Contexte : L'utilisateur doit être connecté au système.
Cas : Deux possibilités s'offrent à l'utilisateur pour l'ajout d'un contact :

- Ajouter un contact lors de la visualisation du profil d'un utilisateur (cas 7.3).
- Ajouter un contact en indiquant les deux premières lettres de son nom. Une liste est alors proposée à l'utilisateur dans laquelle il choisit l'utilisateur qui l'intéresse.

Importance : Important.

Cas 9.2 : Créer un groupe de contacts

Acteur : Membre
Contexte : L'utilisateur doit être connecté au système.
Cas : Cette fonctionnalité crée un groupe de contacts avec le nom que l'utilisateur précise dans le processus de création.
Importance : Peu important.

Cas 9.3 : Consulter la liste des contacts et des groupes

Acteur : Membre
Contexte :
Cas : L'utilisateur consulte le contenu de son carnet d'adresse sous forme d'une liste de contacts et de groupes de contacts, la consultation des contacts d'un groupe se fait sous la même forme. A partir de cette liste, il a la possibilité d'agir sur un contact ou sur un groupe de contacts (cas 9.2).
Importance : Important.

Cas 9.4 : Agir sur un contact ou un groupe

Acteur : Membre
Contexte :
Cas : L'utilisateur peut agir sur un contact ou sur un groupe de contacts. Il peut :

- Supprimer un contact : La suppression d'un contact le retire du carnet d'adresses. Cette action est irréversible.
- Bloquer un contact : Lorsqu'un utilisateur bloque un contact, il ne reçoit plus aucune recommandation de la part de ce contact. Par contre, il a la possibilité d'émettre des recommandations à ce contact. L'utilisateur peut aussi à tout moment décider de débloquer le contact et ainsi recevoir à nouveau des recommandations de sa part.
- Supprimer un groupe de contacts : Cette action supprime le groupe de contact et l'ensemble de son contenu.
- Ajouter un contact dans un groupe : Plusieurs possibilités s'offrent à l'utilisateur pour l'ajout d'un contact dans un groupe : il peut soit indiquer le groupe du contact au moment de son ajout dans le carnet d'adresses (cas 9.1) ou déplacer un contact déjà présent dans le carnet d'adresse dans un groupe. L'utilisateur a aussi la possibilité de déplacer un contact d'un groupe à un autre.
- Supprimer un contact d'un groupe : Le contact est supprimé du groupe.

Importance : Important.

4.1 Récapitulatif des cas d'utilisation

	Numéro	Cas	Importance
Rechercher un film	1	Rechercher suivant un mot-clé	Important
S'inscrire	2.1	Fournir les données personnelles	Très important
	2.2	Définir les paramètres par défaut	Peu important
Consulter les détails d'un film	3	Consulter les détails d'un film	Très important
Accéder au système	4.1	S'identifier	Très important
	4.2	Se déconnecter	Très important
Consulter ses listes des films	5.1	Consulter ses recommandations	Très important
	5.2	Consulter ses évaluations	Très important
	5.3	Consulter sa liste des favoris	Peu important
Agir sur un film	6.1	Evaluer le film	Très important
	6.2	Commenter le film	Peu important
	6.3	Recommander le film à un contact	Peu important
	6.4	Ajouter le film à sa liste des favoris	Peu important

	6.5	Supprimer le film de sa liste des favoris	Peu important
Visualiser les communautés	7.1	Visualiser les communautés selon le critère « contenu »	Très important
	7.2	Visualiser les communautés selon le critère « qualité »	Très important
	7.3	Visualiser le profil d'un utilisateur	Très important
Gérer son profil	8.1	Visualiser son vecteur de positionnement dans les communautés	Important
	8.2	Modifier ses données personnelles	Important
	8.3	Modifier ses paramètres par défaut	Peu important
Gérer son carnet d'adresses	9.1	Ajouter un contact	Important
	9.2	Créer un groupe de contacts	Peu important
	9.3	Consulter la liste des contacts et des groupes	Important
	9.4	Agir sur un contact ou un groupe	Important

5 Exigences non fonctionnelles

5.1 Aspects ergonomiques

L'interface du système COCoFil3 devra être facile d'utilisation et compréhensible. Elle doit satisfaire les critères d'ergonomie « ***Bastin et Scapin*** » suivants:

- Le guidage de l'utilisateur : L'interface de COCoFil3 devra être suffisamment simple de façon à acquérir un guidage « implicite » de l'utilisateur. Les boutons et les liens indiqueront clairement les actions qu'ils représentent, des informations

concises seront fournies afin d'éviter de surcharger les pages garantissant ainsi la lisibilité.

- La gestion des erreurs : Pour prévenir les erreurs, des alertes avertissent l'utilisateur lorsque ce dernier souhaite effectuer une tâche dangereuse (supprimer une recommandation par exemple). Les messages d'erreur et les alertes seront explicites afin de permettre à l'utilisateur de savoir ce qui se passe.
- La charge de travail : L'utilisateur doit atteindre son but en un minimum de clics. Le nombre de clics minimum par tâche est défini dans le plan de qualité logicielle. Toujours dans le but de réduire la charge de travail de l'utilisateur, le système proposera des valeurs par défaut pour les différents champs d'entrée.
- L'homogénéité/cohérence : La charte graphique sera la même sur toutes les pages du site pour éviter que l'utilisateur se perde dans la navigation. Par exemple, la liste des recommandations fournies s'affichera toujours à gauche, sur toutes les pages. Les images et les logos seront conformes au contexte qu'ils représentent.

5.2 Spécifications techniques de la plate forme d'accueil pour l'utilisateur

La plate-forme COCoFil3 sera un site Web, destiné à être utilisé par l'intermédiaire de multiples navigateurs. Nous allons mener ce projet afin que le site soit utilisable sur les navigateurs suivants :

- Microsoft Internet explorer 7
- Mozilla Firefox 2

Par ailleurs, on peut préciser que la navigation sur le site devra aussi bien fonctionner sur une station de travail que sur un ordinateur portable.

5.3 Spécifications techniques de la plate forme d'accueil du système

Le système devra tourner sur les deux plate-formes suivantes : Microsoft Windows (XP, Vista) et Linux.

5.4 Facteurs et Critères de qualité

Dans cette partie, nous donnons les facteurs et critères que le système COCoFil3 devrait respecter ainsi qu'une mesure pour chacun de ses critères.

- Utilisabilité : Capacité du logiciel à proposer une interface facile à apprendre et à utiliser.

- Maintenabilité : Capacité de pouvoir maintenir le logiciel de manière cohérente et à moindre coût. Un des critères choisi pour ce facteur est la modularité (forte cohésion, faible couplage). Afin d'atteindre une forte cohésion, nous avons opté pour une cohésion fonctionnelle ; ainsi les fonctionnalités liées aux recommandations seront regroupées, celles liées aux utilisateurs aussi…En ce qui concerne le couplage, il sera fait par les données : les modules vont échanger des données simples via leurs interfaces.

Facteur	Critère	Mesure
Utilisabilité	Facilité d'apprentissage et d'utilisation	Les mesures seront faites à partir de questionnaires soumis aux utilisateurs après leur utilisation du système.
	Bonne compréhension des entrées/sorties (communicativeness)	
	Performance : temps de réponse	Le temps de réponse moyen à une requête utilisateur doit être de 10 secondes au maximum.
Maintenabilité	Modularité : faible couplage	L'outil metrics sera utilisé pour mesurer le degré de couplage entre modules.
	Simplicité	Une classe doit compter au maximum 500 lignes. Une fonction doit compter au maximum 150.

Equipe MRIM
Laboratoire CLIPS-IMAG
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Plan de tests d'acceptation

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	14/03/2007
Version	1.1

Historique des versions

Version	Date	Modifications
1.0	14/03/2007	Début de rédaction
1.1	12/04/2007	Modification suite aux remarques faites lors du 1er audit : le document soumis représentant plus un dossier de tests système que de tests d'acception.

Sommaire

1. Introduction

But et portée du document

L'objectif du plan de tests d'acceptation est de vérifier que le système répond au cahier des charges.
Ce document est destiné :

- à notre client : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

A propos du projet

Le projet COCoFil3 consiste à développer une application Web autour d'un système de recommandations basé sur les films. Au cours de précédents travaux, le noyau d'un système de recommandation orienté vers les communautés, nommé COCoFil2 a été développé par An-Te Nguyen. COCoFil3 vise à rendre ce système interactif et dynamique.

2. Utilisation des tests d'acceptation

Les tests d'acceptation servent à montrer que le logiciel est conforme au cahier des charges. Ils sont rédigés à l'étape d'analyse des besoins et sont effectués par l'équipe de développement, en présence du maître d'ouvrage pour la validation.

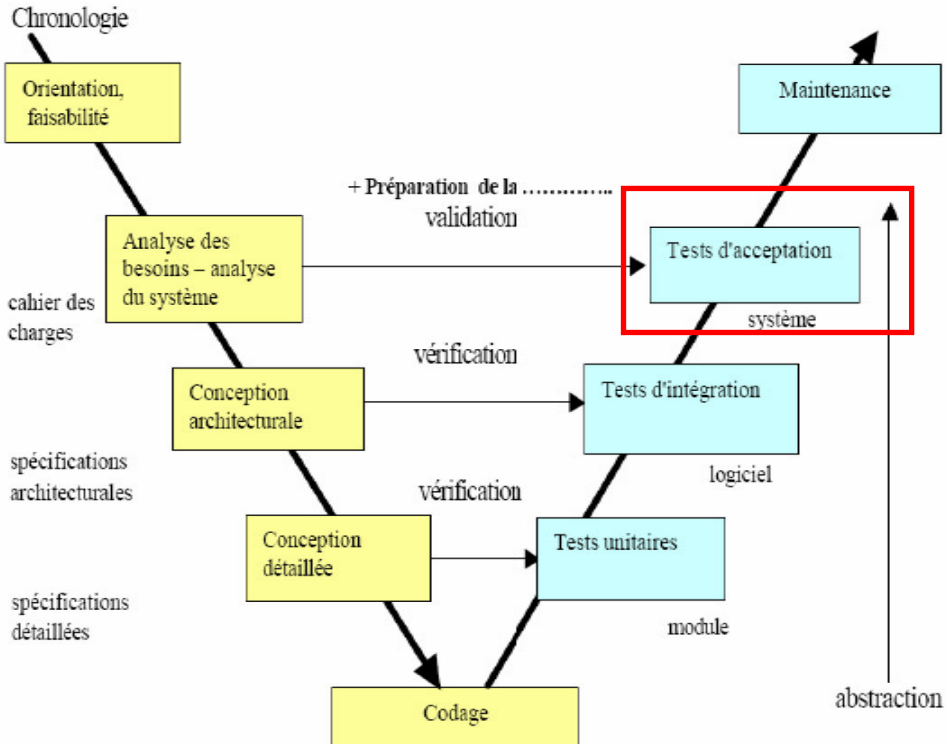


Figure 1: Cycle de vie par incréments e tests d'acceptation

3. Scénarios d'acceptation

Scénario : « Routine »

Rémi utilise régulièrement Cocofil3 : véritable cinéphile il se connecte au système presque toutes les semaines pour y intégrer les évaluations des films qu'il a vus récemment, et pour consulter les recommandations que le système lui propose. Cela fait déjà au moins 5 mois que Rémi est inscrit et il a déjà entré plus de 60 évaluations.
Aujourd'hui Rémi se connecte pour évaluer 2 films qu'il a vus la semaine dernière, pour cela il va consulter sa liste de recommandations pour retrouver les deux films. Il évalue le premier film et avant d'évaluer le second, il l'ajoute dans ses favoris puis l'évalue.
Maintenant, Rémi va consulter sa liste des favoris à laquelle il avait récemment ajouté un film après avoir lu sa fiche ; Rémi note le film en question avant de le supprimer de sa liste des favoris.

Fonctionnalités couvertes par le scénario

Fonctionnalité	Must	Should	May
S'identifier	*		
Consulter les recommandations	*		
Evaluer un film	*		
Ajouter le film à sa liste des favoris			*
Consulter sa liste des favoris			*
Supprimer le film de sa liste des favoris			*

Scénario : « Découverte »

Aujourd'hui, Isabelle a découvert par hasard le système Cocofil3. Curieuse de ce qu'un tel système pourrait lui apporter, elle va l'essayer.
Elle commence par effectuer une recherche de films pour avoir une idée des films disponibles dans le système, ensuite, elle décide de s'inscrire et va passer par toutes les étapes lui permettant de recevoir les premières recommandations du système, afin de se rendre compte de la qualité de ces recommandations.

Fonctionnalités couvertes par le scénario

Fonctionnalité	Must	Should	May
Rechercher un film		*	
Fournir ses données personnelles	*		
Définir les paramètres par défaut			*
Consulter ces recommandations	*		

Scénario : « Communautés »

Stéphane s'intéresse au cinéma, et il utilise Cocofil3 depuis déjà un mois, mais il est plutôt déçu des recommandations que le système lui a fournies jusqu'ici.
Aujourd'hui, il a décidé de se connecter pour comprendre comment le système le situe par rapport aux autres utilisateurs. Il va étudier les évaluations des utilisateurs que le système dit proches de lui selon l'une ou l'autre des communautés, pour voir si cela correspond à sa propre perception. Il espère en déduire quelques repères pour savoir quels espaces de communautés sont les plus adaptés pour recevoir des recommandations qui lui plaisent.
Toujours dans le but de comprendre sa situation par rapport aux autres utilisateurs, Stéphane consulte son vecteur de positionnement et consulte des informations sur les communautés auxquelles il appartient.
Enfin, Stéphane se déconnecte du système.

Fonctionnalités couvertes par le scénario

Fonctionnalité	Must	Should	May
S'identifier	*		
Visualiser la carte selon le critère « contenu »	*		
Visualiser la carte selon le critère « qualité »	*		

Visualiser le profil d'un utilisateur	*		
Visualiser son vecteur de positionnement		*	
Se déconnecter	*		

Scénario : « Bouche à oreille » (filtrage actif)

Olivia a un ami, José, qui lui a dit qu'il utilisait Cocofil3 ainsi que quelques autres de ses amis cinéphiles.
Aujourd'hui, Olivia s'inscrit pour recommander à José, et peut-être à ses autres amis, un vieux film qu'elle a vu récemment au cinéclub, qu'elle a aimé et qu'elle pense qu'ils aimeront aussi. Après son inscription, elle se connecte au système, ajoute José et quelques autres amis à sa liste des contacts. Ensuite, elle effectue une recherche pour trouver le film qu'elle souhaite recommander à ses amis. Elle parcourt les résultats de recherche trouve le film, consulte sa fiche, puis le recommande à José.

Fonctionnalités couvertes par le scénario

Fonctionnalité	Must	Should	May
S'inscrire	*		
S'identifier	*		
Ajouter un contact		*	
Recommander un film à un contact			*

4. Exécution des tests

Stratégie de tests et Production des jeux de données

Tous les scénarios sont exécutés les uns après les autres par le maître d'œuvre.
Pour chaque scénario, les jeux de données requis seront disponibles. Ils seront crées manuellement par l'équipe de développement.

Le déroulement sera le même pour chacun des scénarios :

- Lancer le navigateur
- Suivre le scénario
- Analyser les résultats.

Analyse des résultats

Les résultats seront examinés pour voir s'ils correspondent aux attentes. Si une erreur est révélée lors de ces tests, une fiche sera créé et traitée de la manière suivante par l'équipe de développement.

- analyse de l'erreur afin de déterminer son origine et les modifications à effectuer pour la corriger.
- estimation du coût en nombre d'hommes/jour pour réaliser la modification.
- si le coût est acceptable, modification et nouvelle acceptation grâce à de nouveaux tests d'acceptation.
- si le coût n'est pas acceptable, rédaction d'un plan de résolution d'erreurs à destination de la future équipe de développement.

Equipe MRIM
Laboratoire CLIPS-IMAG
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Dossier de Spécifications Externes

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	14/05/2007
Version	1.2

Historique des versions

Version	Date	Modifications
1.0	10/04/2007	Début de la rédaction
1.1	14/05/2007	Modification suite aux remarques faites lors du 2ème audit (principalement sur les arbres des tâches)
1.2	12/06/2007	Modification suite aux remarques faites après relecture par Nathalie DENOS.

Sommaire

1. Introduction

But du document

Ce document présente les spécifications externes du projet de réalisation d'une application Web pour un système de recommandation de films grâce au filtrage collaboratif basé sur les communautés.

Les spécifications externes entrent dans le cadre de l'organisation et de la gestion du développement logiciel. Elles ont pour objectif de décrire exactement ce que sera le système du point de vue de l'utilisateur.

Portée du document

Ce document s'adresse :

- Au responsable de stage : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

Documentation de référence

Le présent document fait référence au Cahier des Charges du projet et est rédigé en fonction des clauses qualités définies dans le Plan d'Assurance Qualité Logicielle.

2. Spécifications fonctionnelles

Les spécifications fonctionnelles seront organisées par groupe de fonctionnalités, celles qui concernent les films, les utilisateurs, les sessions, les recommandations, les communautés, les évaluations, les commentaires, les favoris et le carnet d'adresses.

Pour chaque fonctionnalité, le cas d'utilisation correspondant dans le Cahier des Charges est précisé ainsi que la priorité dans l'ordre d'implémentation (1 étant le plus prioritaire et 3 le moins prioritaire).

Fonctionnalités spécifiques aux films

Recherche d'un film

But : Rechercher un film par mots-clés.
Pré-condition : Aucune.
Exécution : L'utilisateur tape les mots-clés correspondant au film recherché et lance sa requête.
Post-condition : Une liste de films correspondant à la requête de l'utilisateur est affichée (ou le message « Aucun film trouvé » si aucun film ne correspond à la requête de l'utilisateur).
Priorité : 2
Cas d'utilisation : cas 1.

Description d'un film

But : Consulter les détails d'un film.
Pré-condition : L'utilisateur consulte une liste de films (résultat d'une recherche, film recommandé, favoris, etc.).
Exécution : L'utilisateur clique sur le titre d'un film dans la liste.
Post-condition : La description du film est affichée : genres, mots-clés, acteurs, producteur, etc. De plus, les commentaires ajouter par les utilisateurs sont visibles.
Priorité : 1
Cas d'utilisation : cas 3.

Fonctionnalités spécifiques aux utilisateurs

S'inscrire

But : L'inscription permet à un utilisateur de conserver et d'enrichir son profil au cours des connexions futures.
Pré-condition : L'utilisateur n'est pas identifié (il n'a pas ouvert de session). Un utilisateur ne peut s'inscrire qu'une et une seule fois.
Exécution : L'utilisateur fournit toutes les informations nécessaires : nom d'utilisateur (adresse email), mot de passe, adresse (ville et code postal), profession (choix dans une liste déroulante), âge et paramètres (anonymat, carnet d'adresse, confidentialité, affichage de film).
Post-condition : L'utilisateur dispose d'un nom et d'un mot de passe avec lesquels il peut se connecter en tant que membre. Il reçoit ses premières recommandations. Un carnet d'adresses vide et une liste de favoris vide lui sont assignés.
Priorité : 1
Cas d'utilisation : cas 2.

Modifier les données personnelles

But : Modifier les données personnelles permet à l'utilisateur de mettre à jour son profil en cas de changement de situation (profession, adresse) ou en cas d'erreur de saisie à l'enregistrement.
Pré-condition : Être identifié.
Exécution : Les données personnelles de l'utilisateur sont affichées et il peut alors modifier son mot de passe, son adresse, sa profession et son âge.
Post-condition : Les nouvelles données sont enregistrées à la place des anciennes (qui elles sont perdues).
Priorité : 2
Cas d'utilisation : cas 8.2.

Modifier ses paramètres par défaut

But : Modifier ses paramètres par défaut.
Pré-condition :L'utilisateur doit être identifié.
Exécution : Après avoir affiché ses paramètres, l'utilisateur peut les modifier. Il valide ensuite ses modifications afin de les enregistrer.
Post-condition : Une fois enregistrées, les nouvelles valeurs des paramètres sont prises en compte par le système.
Priorité : 3

Cas d'utilisation : cas 8.3.

Fonctionnalités spécifiques aux sessions

S'identifier

But : Ouvrir une session permet d'accéder à toutes les fonctionnalités : gestion de profil, consultation des recommandations, visualisation des communautés…
Pré-condition : Etre enregistré mais pas encore identifié.
Exécution : L'utilisateur rentre son nom d'utilisateur et son mot de passe puis valide sa saisie.
Post-condition : Si l'identification se passe bien, l'utilisateur arrive sur la page d'accueil où il accède aux fonctionnalités ; sinon un message d'erreur l'avertit (en indiquant le(s) champ(s) erroné(s)) puis l'invite à essayer de nouveau.
Priorité : 1
Cas d'utilisation : cas 4.1.

Se déconnecter

But : Afin de fermer sa connexion au site, l'utilisateur peut se déconnecter.
Pré-condition : Avoir une session en cours.
Exécution : L'utilisateur sélectionne l'option de déconnexion.
Post-condition : La session de l'utilisateur est fermée.
Priorité : 1
Cas d'utilisation : cas 4.2.

Fonctionnalités spécifiques aux recommandations

Consulter ses recommandations

But : Obtenir ses recommandations.
Pré-condition : L'utilisateur doit s'être identifié.
Exécution : L'utilisateur sélectionne dans le menu le lien correspondant à la consultation de ses recommandations.
Post-condition : Une liste de films correspondant aux recommandations faites à l'utilisateur est affichée (ou le message « Aucun film recommandé » si aucune recommandation n'a encore été produite pour cet utilisateur).
Priorité : 1
Cas d'utilisation : cas 5.1.

Recommander un film à un contact

But : Recommander un film à un de ses contacts
Pré-condition : L'utilisateur doit s'être identifié et avoir au moins un contact dans son carnet d'adresse. L'utilisateur consulte ses recommandations, ses favoris ou les détails d'un film.
Exécution : L'utilisateur choisit le contact à qui il veut recommander le film.
Post-condition : Si le contact a déjà évalué le film, l'utilisateur est averti par un message. Si le contact n'a pas encore évalué le film, le film est ajouté à sa liste de recommandations de ce contact.
Priorité : 3
Cas d'utilisation : cas 6.3.

Fonctionnalités spécifiques aux communautés

Visualiser la carte suivant le critère « contenu »

But : Obtenir la carte des communautés suivant le critère « contenu »
Pré-condition : L'utilisateur doit s'être identifié.
Exécution : L'utilisateur sélectionne dans le menu le lien correspondant à la visualisation de carte suivant le critère « contenu ».
Post-condition : Une carte représentant les communautés des utilisateurs suivant le critère « contenu » est affichée.
Priorité : 1
Cas d'utilisation : cas 7.1.

Carte suivant le critère « qualité »

But : Obtenir la carte des communautés suivant le critère « qualité »
Pré-condition :L'utilisateur doit s'être identifié.
Exécution : L'utilisateur sélectionne dans le menu le lien correspondant à la visualisation de carte suivant le critère « qualité ».
Post-condition : Une carte représentant les communautés des utilisateurs suivant le critère « qualité » est affichée.
Priorité : 1
Cas d'utilisation : cas 7.2.

Visualiser le profil d'un utilisateur

But : Obtenir des informations sur un utilisateur.
Pré-condition : L'utilisateur est en train de visualiser une carte suivant le critère « contenu » ou « qualité ».
Exécution : L'utilisateur sélectionne l'un des utilisateurs dont on peut regarder le profil (utilisateur représenté par un rectangle au bord arrondi ou utilisateur représentatif).
Post-condition : La fiche décrivant le profil de cet utilisateur sélectionné est affichée ; elle comporte le pourcentage des films évalués par genre pour l'utilisateur et celui qu'il a sélectionné ainsi que les notes des films qu'ils ont tous les deux évalués ; la fiche permet donc à l'utilisateur de comparer son profil à celui de l'autre utilisateur.
Priorité : 1
Cas d'utilisation : cas 7.3.

Fonctionnalités spécifiques aux évaluations

Evaluer un film

But : Enregistrer une évaluation sur un film
Pré-condition : L'utilisateur doit être identifié. Il consulte ses recommandations ou ses favoris.
Exécution : L'utilisateur évalue le film en lui attribuant une note entre 1 et 5.
Post-condition : L'évaluation est enregistrée et elle sera affichée dans la liste des évaluations.
Priorité : 1
Cas d'utilisation : cas 6.1.

Visualiser ses évaluations

But : Obtenir la liste de ses évaluations.
Pré-condition : L'utilisateur doit s'être identifié.
Exécution : L'utilisateur sélectionne dans le menu le lien correspondant à la consultation des évaluations.
Post-condition : Une liste de films correspondant aux évaluations faites par l'utilisateur est affichée (ou le message « Aucun film évalué » s'il n'a fait aucune évaluation). L'utilisateur a alors la possibilité de modifier les évaluations qu'il avait fait.
Priorité : 2
Cas d'utilisation : cas 5.2.

Fonctionnalités spécifiques aux commentaires

Commenter un film

But : Ajouter un commentaire sur un film.
Pré-condition : L'utilisateur doit être identifié et avoir évalué le film à commenter. Il est en train de consulter la fiche du film.
Exécution : L'utilisateur tape son avis sur le film dans le champ texte correspondant au commentaire. Une fois fini, il valide sa saisie afin d'enregistrer son commentaire.
Post-condition : Le commentaire saisi est ajouté aux autres commentaires du film.
Priorité : 3
Cas d'utilisation : cas 6.2

Consulter les commentaires d'un film

But : Obtenir la liste des commentaires d'un film
Pré-condition : L'utilisateur doit être identifié, il consulte les détails d'un film.
Exécution : L'utilisateur sélectionne le lien pour consulter les commentaires.
Post-condition : Tous les commentaires sur le film sont affichés.
Priorité : 3
Cas d'utilisation : cas 3.

Fonctionnalités spécifiques aux favoris

Ajouter un film aux favoris

But : Ajouter un film à sa liste de favoris
Pré-condition : L'utilisateur doit s'être identifié. Il consulte une liste de films (sauf la liste des favoris) ou il consulte les détails d'un film. Le film à ajouter aux favoris ne doit pas en faire déjà parti.
Exécution : L'utilisateur clique sur la « checkbox » servant à indiquer si le film appartient ou non aux favoris.
Post-condition : Le film est ajouté aux favoris et la « checkbox » de ce film sera cochée tant que le film fera parti des favoris.
Priorité : 3
Cas d'utilisation : cas 6.4.

Supprimer un film des favoris

But : Supprimer un film de sa liste de favoris
Pré-condition : L'utilisateur doit s'être identifié. Il consulte une liste de films ou il consulte les détails d'un film. Le film à supprimer des favoris doit en faire partie.
Exécution : L'utilisateur clique sur la « checkbox » servant à indiquer si le film appartient ou non aux favoris.
Post-condition : le film est supprimé des favoris et la « checkbox » de ce film n'est plus cochée.
Priorité : 3
Cas d'utilisation : cas 6.5.

Consulter ses favoris

But : Obtenir sa liste de favoris.
Pré-condition : L'utilisateur doit s'être identifié
Exécution : L'utilisateur sélectionne dans le menu le lien correspondant à la consultation de ses favoris.
Post-condition : Une liste de films correspondante aux favoris de l'utilisateur est affichée (ou le message « Aucun film favori » si l'utilisateur n'a pas de favori).
Priorité : 3
Cas d'utilisation : cas 5.3.

Fonctionnalités spécifiques au carnet d'adresses

Ajouter un contact

But : Ajouter un contact à son carnet d'adresses.
Pré-condition : L'utilisateur doit s'être identifié.
Exécution : Soit l'utilisateur est en train de consulter le profil d'un utilisateur, il peut alors directement l'ajouter à ses contacts en cliquant sur le bouton correspondant. Soit l'utilisateur est en train de gérer son carnet d'adresses, il peut alors ajouter un contact en rentrant les premières lettres de son nom d'utilisateur afin d'obtenir une liste correspondant aux utilisateurs possibles. Puis il sélectionne l'utilisateur désiré et valide son choix. Si le contact fait déjà partie du carnet d'adresse, un message d'erreur est affiché à l'utilisateur.
Post-condition : Le contact est ajouté dans le carnet d'adresse de l'utilisateur.
Priorité : 3
Cas d'utilisation : cas 9.1.

Supprimer un contact

But : Supprimer un contact de son carnet d'adresses.
Pré-condition : L'utilisateur doit s'être identifié, il consulte le contenu de son carnet d'adresses. Le contact à supprimer figure dans le carnet d'adresses.
Exécution : L'utilisateur sélectionne le contact à supprimer et clique sur le bouton de suppression puis confirme son choix afin de finaliser la suppression.
Post-condition : Le contact ne figure plus dans le carnet d'adresses.
Priorité : 3
Cas d'utilisation : cas 9.4.

Bloquer / débloquer un contact

But : Bloquer ou débloquer un contact de son carnet d'adresses.
Pré-condition : L'utilisateur doit s'être identifié. Il consulte le contenu de son carnet d'adresses.
Exécution : L'utilisateur sélectionne le contact à bloquer / débloquer puis clique sur le bouton correspondant afin de changer cette caractéristique.
Post-condition : Si le contact devient « bloqué », il ne pourra plus envoyer de recommandations à l'utilisateur. Si le contact devient « débloqué », il peut à nouveau envoyer des recommandations à l'utilisateur.
Priorité : 3
Cas d'utilisation : cas 9.4.

Créer un groupe de contact

But : Créer un groupe de contact dans son carnet d'adresses.
Pré-condition : L'utilisateur doit s'être identifié. Il consulte le contenu de son carnet d'adresses.
Exécution : L'utilisateur clique sur le bouton servant à ajouter un groupe de contact. Il tape le nom de ce groupe.
Post-condition : Un groupe portant le nom rentré est ajouté à la racine du carnet d'adresses.
Priorité : 3
Cas d'utilisation : cas 9.2.

Supprimer un groupe de contact

But : Supprimer un groupe de contacts de son carnet d'adresses.
Pré-condition : L'utilisateur doit s'être identifié. Il consulte le contenu de son carnet d'adresses.
Exécution : L'utilisateur sélectionne le groupe qu'il désire supprimer puis clique sur le bouton de suppression. On l'informe que la suppression d'un groupe entraîne la suppression de tous les contacts le composant. Si l'utilisateur confirme son choix le groupe des contacts est supprimé.
Post-condition : Le groupe et l'ensemble de son contenu sont supprimés.
Priorité : 3
Cas d'utilisation : cas 9.4.

Ajouter un contact dans un groupe

But : Ajouter un contact dans un groupe de contacts de son carnet d'adresses.
Pré-condition : L'utilisateur doit s'être identifié.
Exécution : Soit l'utilisateur ajoute un contact dans son carnet d'adresses, il précise le groupe auquel il veut qu'il appartient ou dans la consultation du contenu de son carnet d'adresses, il sélectionne le contact puis clique sur le bouton déplacer pour indiquer le groupe vers lequel le contact doit être déplacé.
Post-condition : Le contact fait désormais parti du groupe choisi par l'utilisateur.
Priorité : 1
Cas d'utilisation : cas 9.4.

Supprimer un contact d'un groupe

But : Supprimer un contact d'un groupe de contacts.
Pré-condition : L'utilisateur doit s'être identifié. Il consulte le contenu d'un groupe de contacts dont le contact à supprimer fait partie.
Exécution : L'utilisateur sélectionne le contact qu'il désire supprimer puis clique sur le bouton de suppression. On lui demande de confirmer. Si l'utilisateur confirme son choix le contact est supprimé du groupe.
Post-condition : Le groupe et l'ensemble de son contenu sont supprimés.
Priorité : 3
Cas d'utilisation : cas 9.4.

3. Arbre de tâches

Les fonctionnalités proposées par l'application COCoFil3 peuvent être regroupées au sein d'un arbre des tâches. Cet arbre (ainsi que ses sous arbres) montre l'enchaînement des actions qui correspondent aux cas d'utilisation.

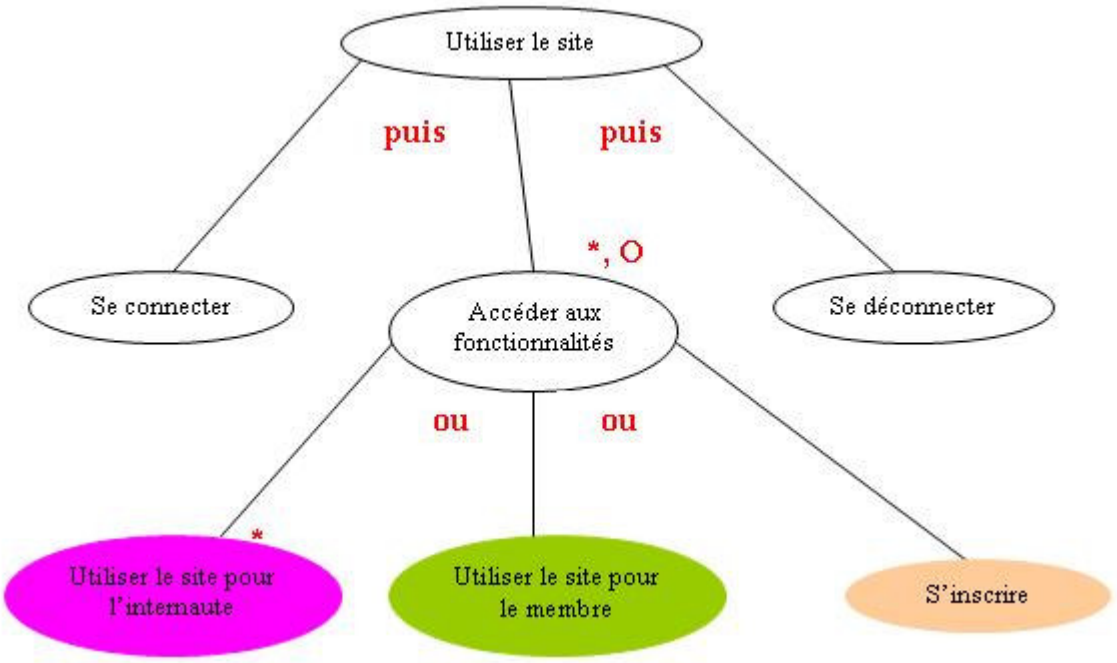
Notations utilisées

- T = T1 **puis** T2 signifie qu'il faut réaliser T1 puis T2 pour réaliser T.
- T = T1 **ou** T2 signifie qu'il faut réaliser T1 ou T2 pour réaliser T. Ce « **ou** » est exclusif.
- T = T1 ↔ T2 signifie qu'il faut réaliser T1 et T2 dans un ordre quelconque pour réaliser T.
- Les parenthèses servent à grouper des tâches.
- * signifie que la tâche peut être répétée une ou plusieurs fois.
- **O** signifie que la tâche est optionnelle.
- **C** signifie qu'il s'agit d'une tâche critique.
- **F** signifie qu'il s'agit d'une tâche fréquente.
- Entre accolades **{…}**, on spécifie la priorité de développement. Si aucune spécification alors il s'agit d'une priorité must.
- Entre crochet **[…]**, on spécifie le(s) concept(s) manipulés par la tâche.

Lorsqu'une tâche est divisée en sous tâches elle est en couleur.

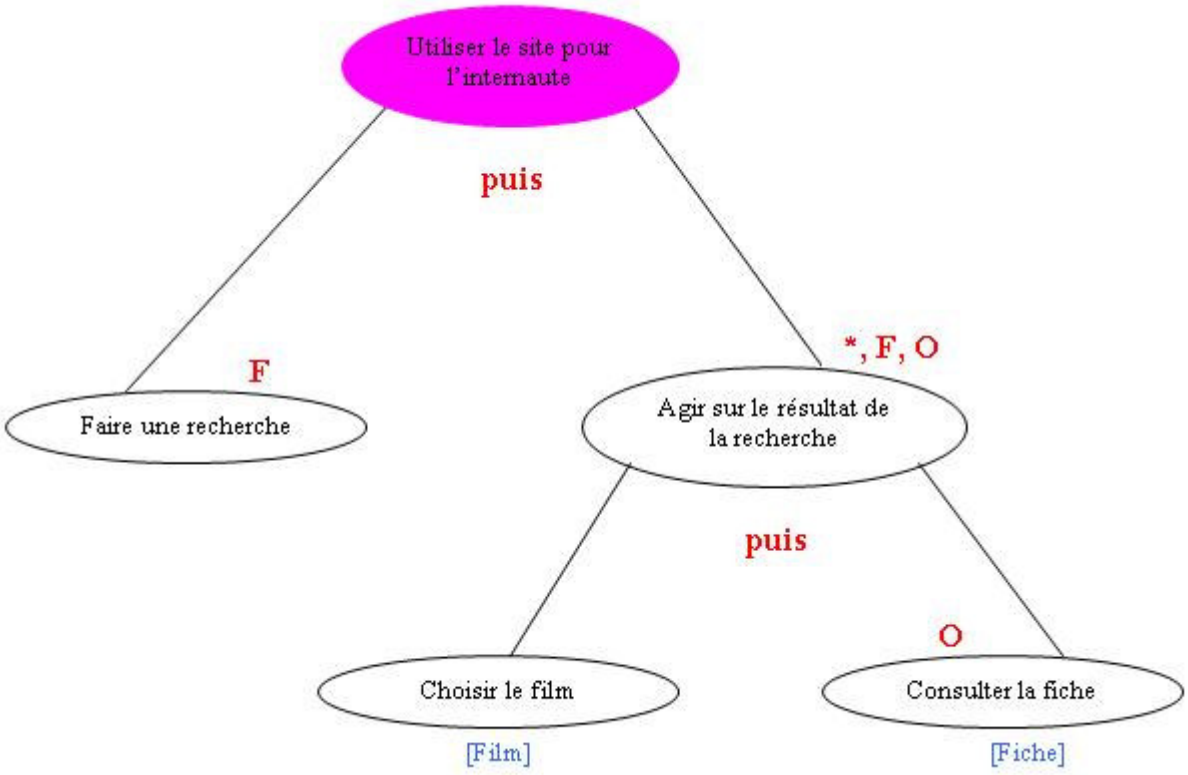
Utiliser le site

Pour utiliser le site, l'utilisateur doit tout d'abord se connecter. Une fois connecté au site, il peut utiliser le site en tant qu'internaute puis accéder à toutes les fonctionnalités offertes. L'utilisateur peut à tout moment se déconnecter du site.



Utiliser le site pour l'internaute

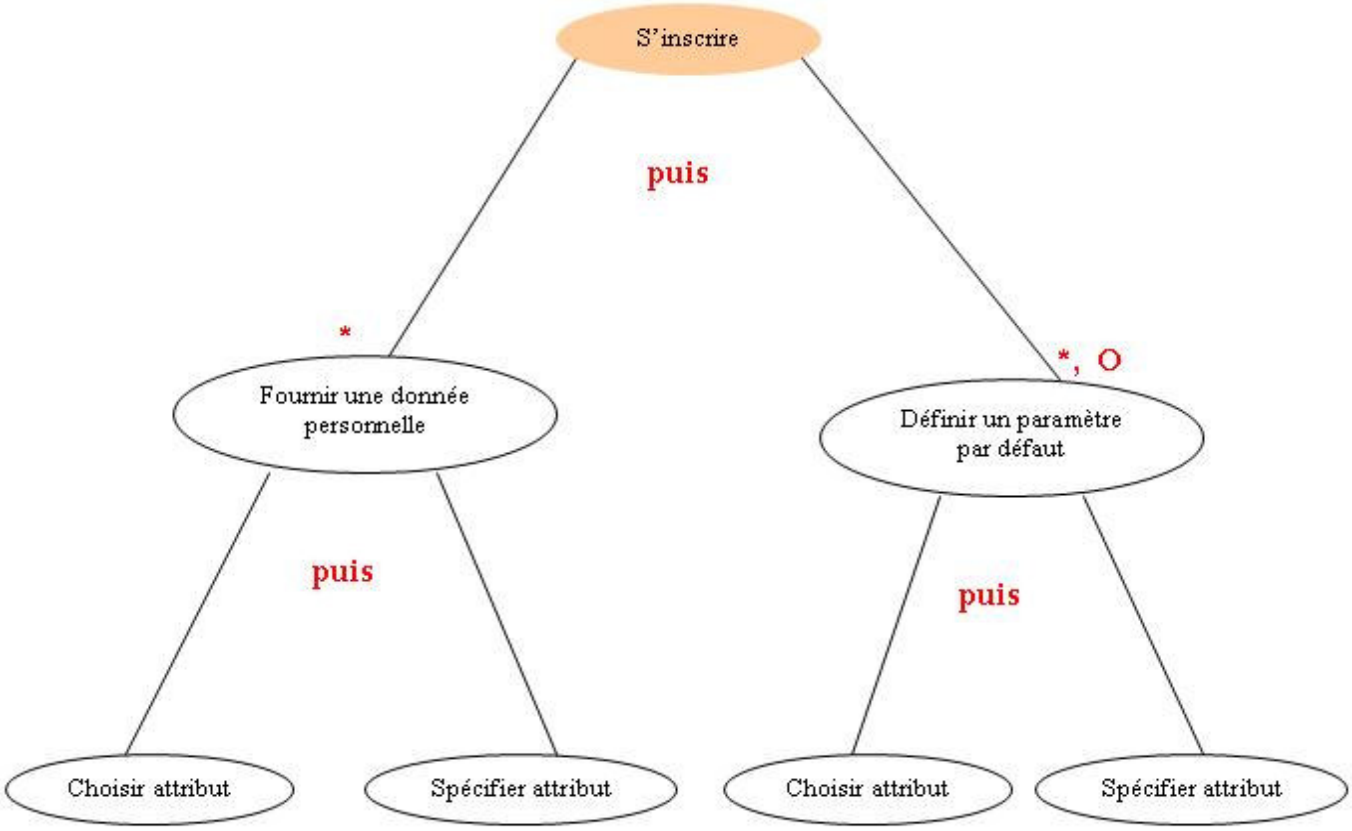
Lorsqu'un utilisateur utilise le système en tant qu'internaute, il peut effectuer une recherche de films.
Le résultat d'une recherche est affiché sous forme d'une liste de films sur lesquels il peut agir en consultant la fiche.



S'inscrire

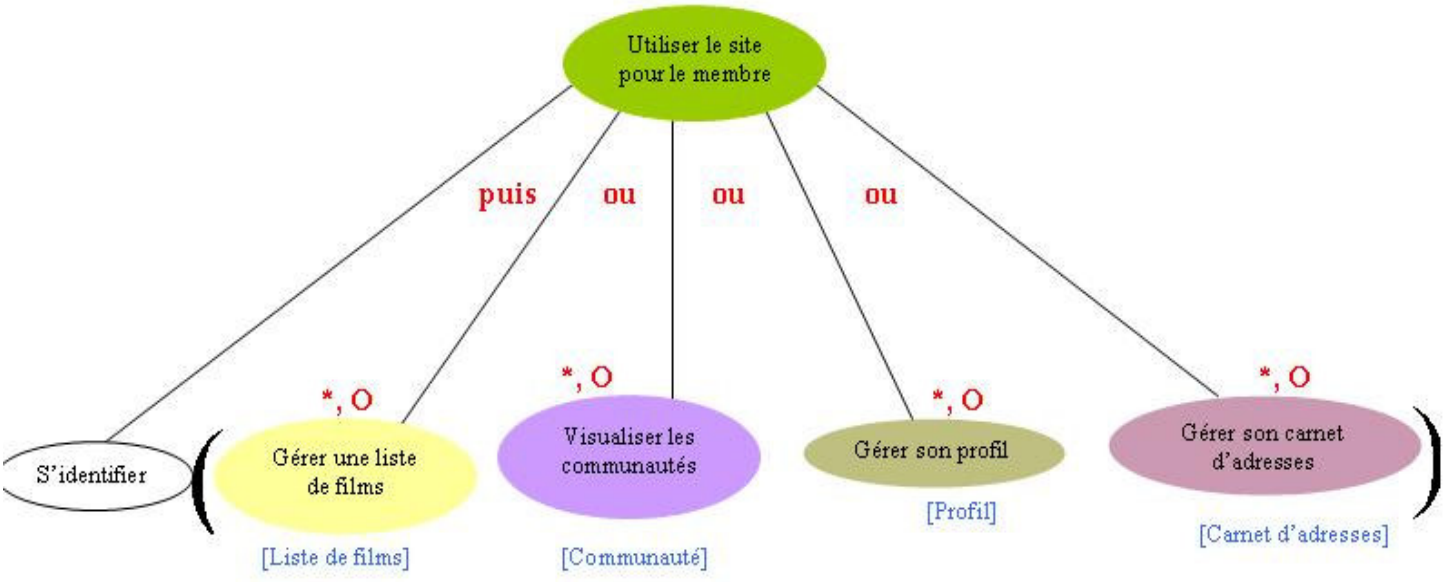
L'inscription se fait en deux étapes :

- Dans un premier temps, l'utilisateur doit fournir ses données personnelles (nom d'utilisateur, mot de passe, âge, adresse et profession).
- Dans un deuxième temps, il définit ses paramètres par défaut concernant l'anonymat, le carnet d'adresses, la confidentialité et l'affichage des films.



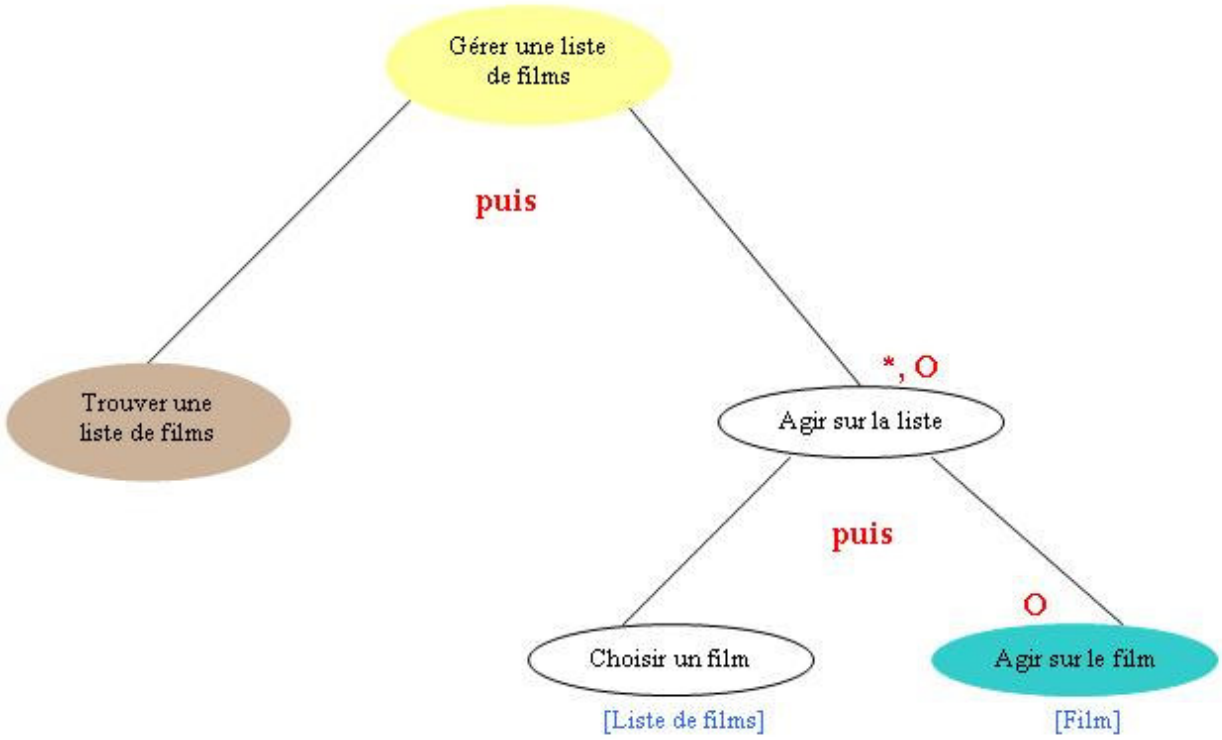
Utiliser le site pour le membre

Pour utiliser le site en tant que membre, l'utilisateur doit s'identifier. Une fois identifié, il devient membre et accède à toutes les fonctionnalités destinées aux membres : gestion du profil, gestion du carnet d'adresses, visualisation des communautés…



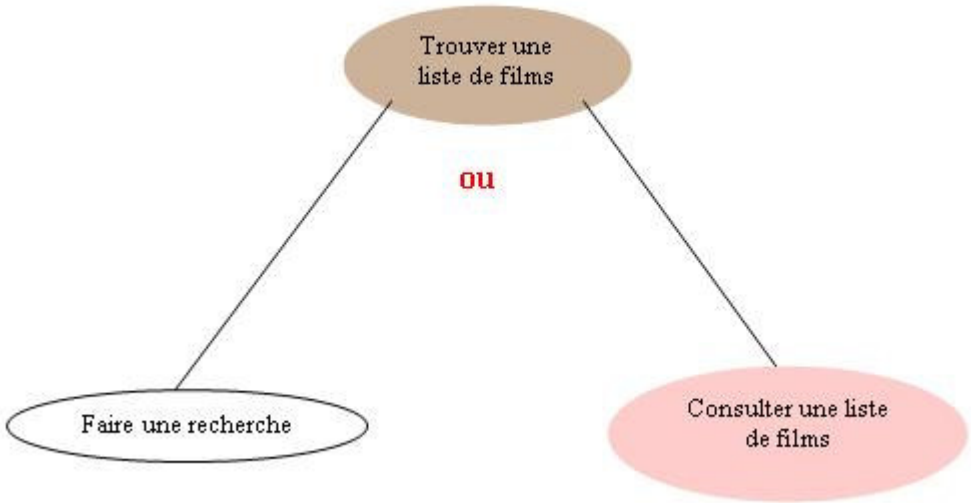
Gérer une liste de films

Pour agir sur un film, l'utilisateur doit tout d'abord trouver une liste de films.



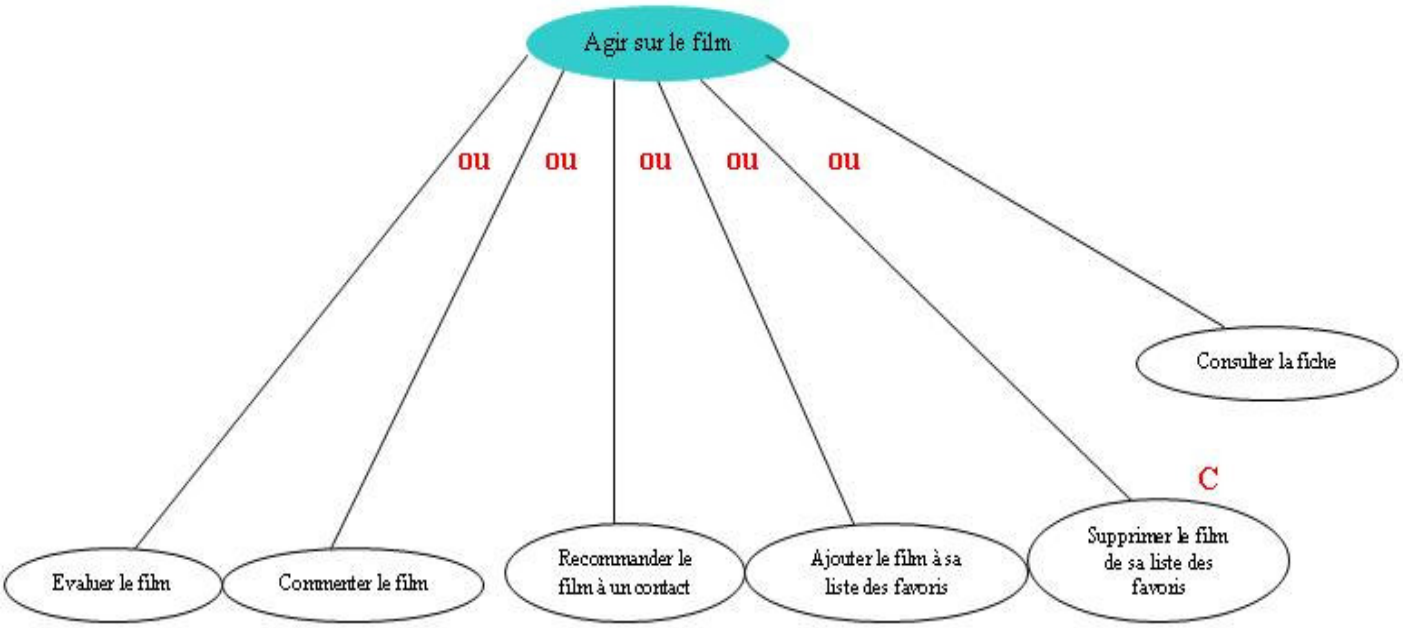
Trouver une liste de films

L'utilisateur a deux moyens de trouver une liste de films soit en effectuant une recherche ou en consultant ses listes de films.



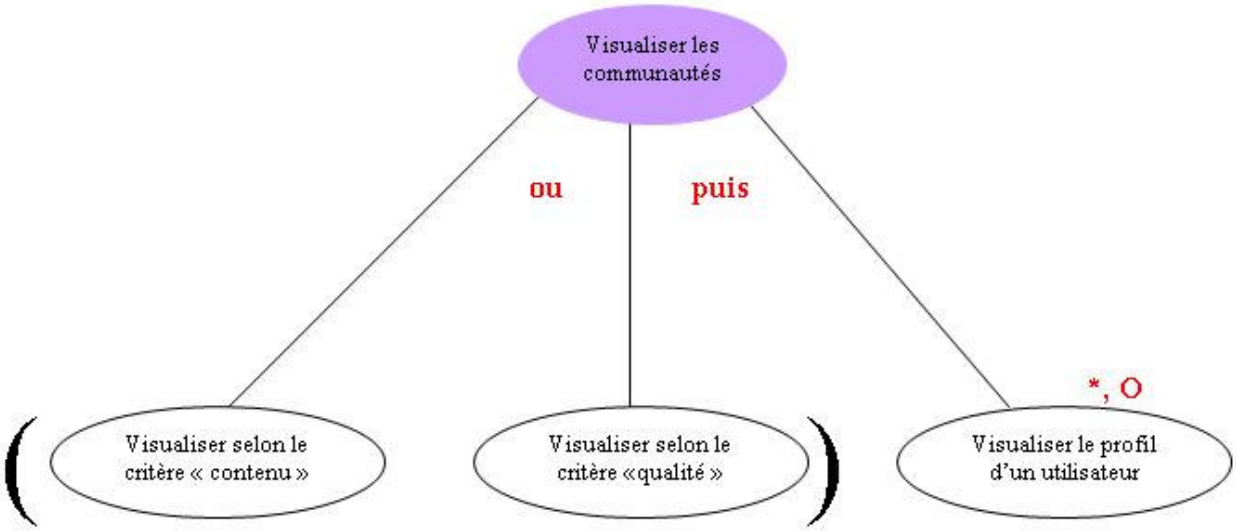
Agir sur le film

L'utilisateur peut agir sur un film, il peut l'évaluer, le commenter, le recommander à un contact, l'ajouter à sa liste des favoris, le supprimer de sa liste des favoris ou consulter sa fiche.



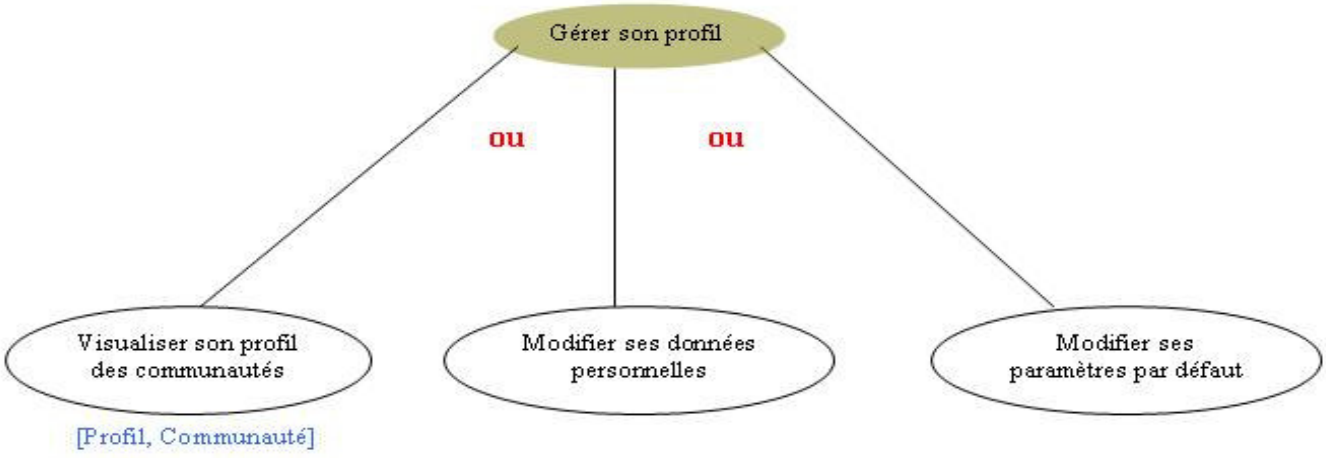
Visualiser les communautés

L'utilisateur peut visualiser les communautés selon le critère « contenu » ou selon le critère « qualité » puis il peut agir sur la carte des communautés en visualisant le profil d'un autre utilisateur.



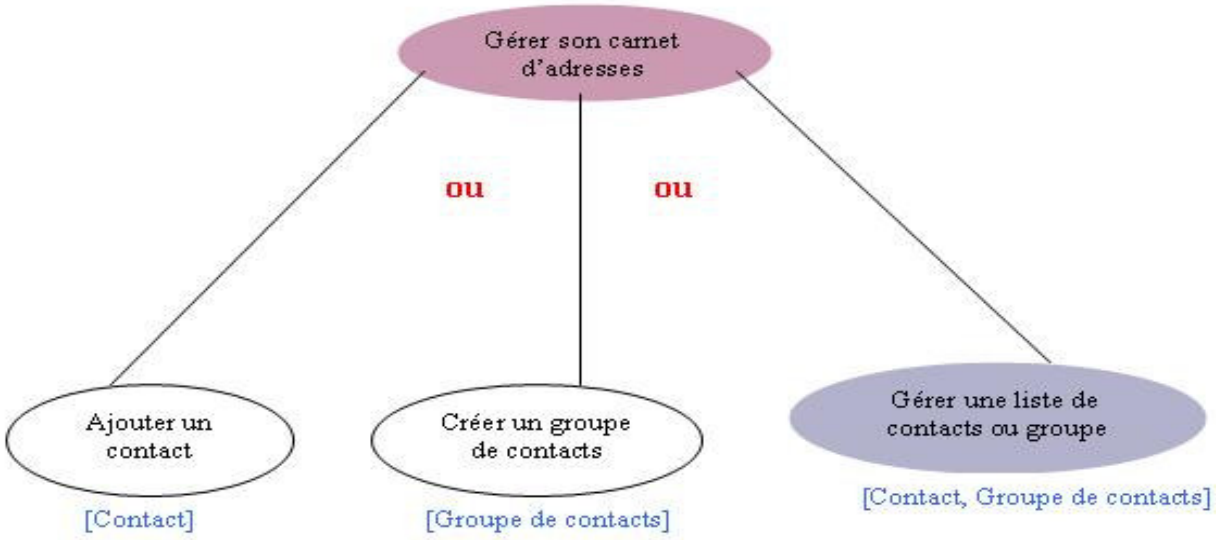
Gérer son profil

Dans la gestion de son profil, l'utilisateur peut visualiser vecteur de positionnement dans les communautés, modifier ses données personnelles ou modifier ses paramètres par défaut.



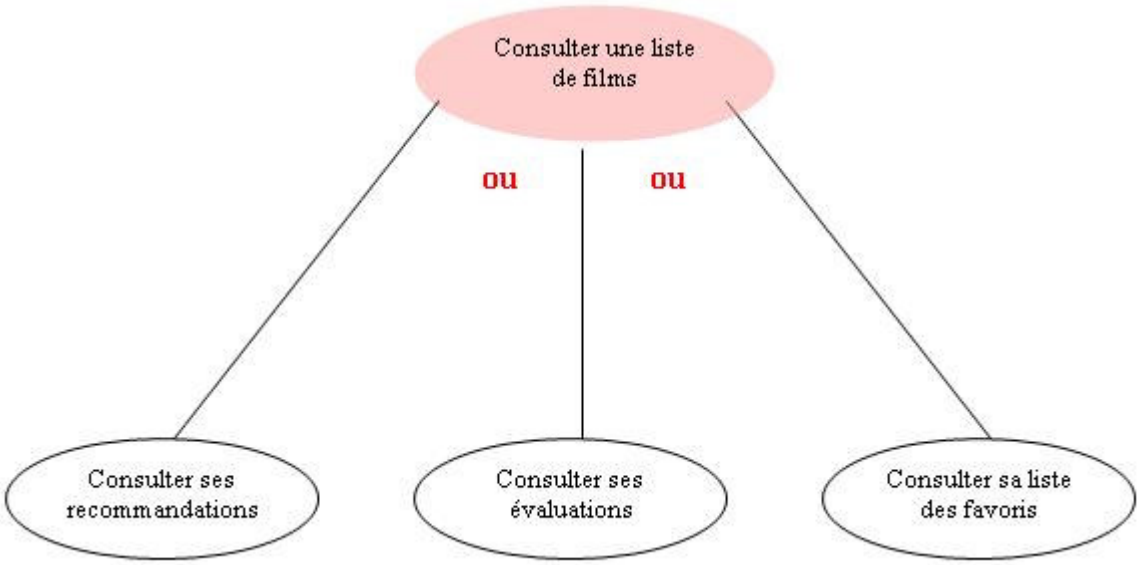
Gérer son carnet d'adresses

Dans la gestion de son carnet d'adresses, l'utilisateur peut ajouter un contact, créer un groupe de contacts ou gérer une liste de contacts ou groupe.



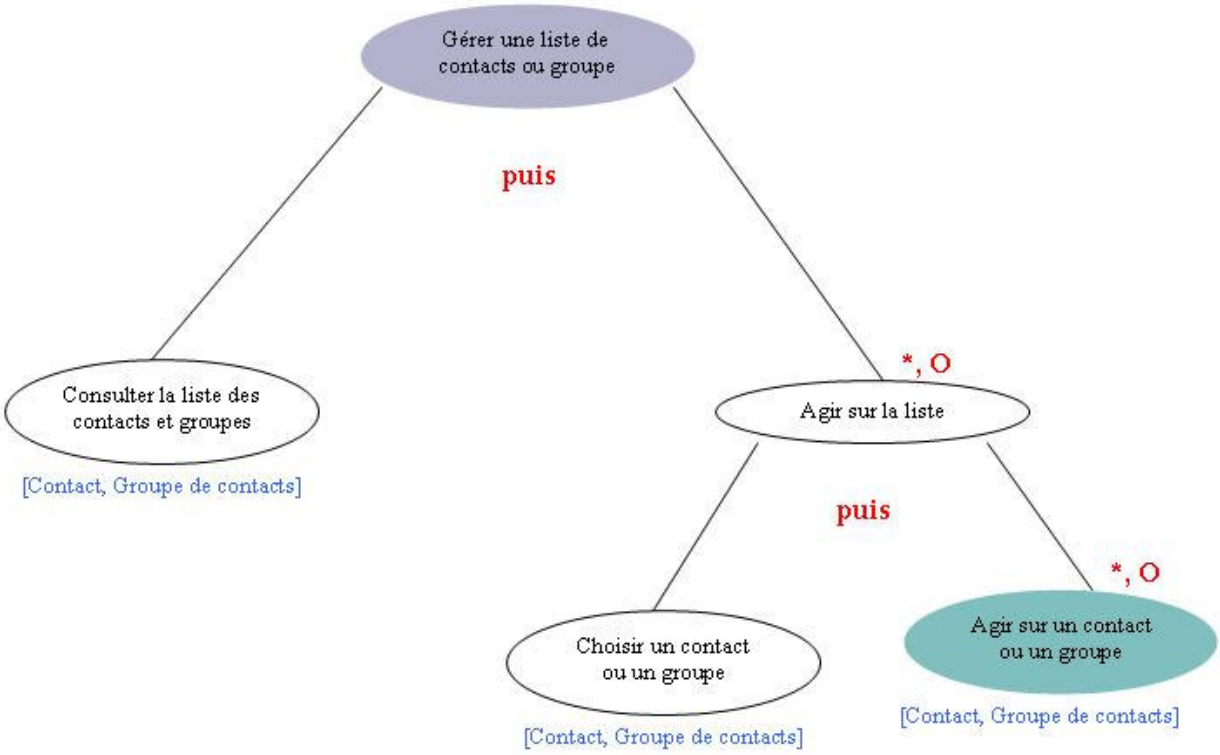
Consulter une liste de films

Les recommandations, les évaluations et la liste des favoris correspondent à des listes de films que l'utilisateur peut consulter à tout moment.



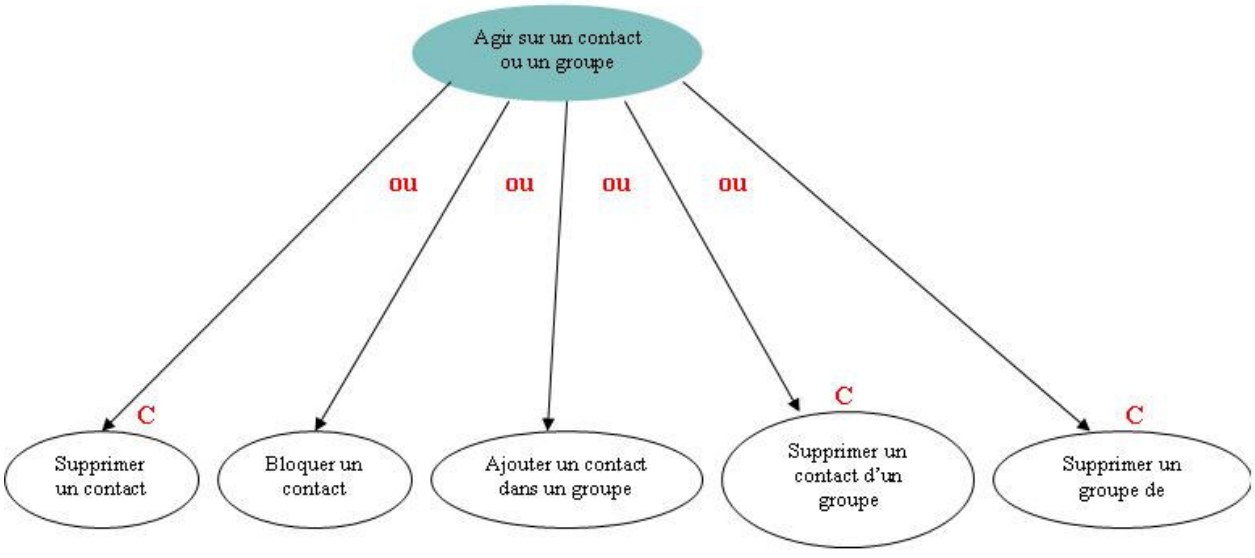
Gérer une liste de contacts ou groupe

L'utilisateur a la possibilité de consulter le contenu de son carnet d'adresses. Ensuite, il peut choisir un contact ou un groupe puis agir dessus.



Agir sur un contact ou un groupe

L'utilisateur peut agir sur un contact en le bloquant, le supprimant, l'ajoutant ou le supprimant d'un groupe de contact. Il peut aussi agir sur un groupe de contacts en le supprimant.



4. Spécifications IHM

COCoFil3 offre un accès très restreint à un internaute (utilisateur non identifié), il accède essentiellement à trois fonctionnalités : la recherche, l'identification et l'inscription. Par conséquent, l'interface d'accueil pour un internaute qui constitue aussi la page d'accueil de l'application est différente de l'interface d'accueil pour un membre (utilisateur identifié).
Nous présentons ci-dessous l'IHM abstraite correspondante à la page d'accueil et celle correspondantes aux autres pages.

IHM abstraite de l'accueil

L'IHM abstraite suivante présente les différents espaces de travail de la page d'accueil COCoFil3 ainsi que leur agencement.

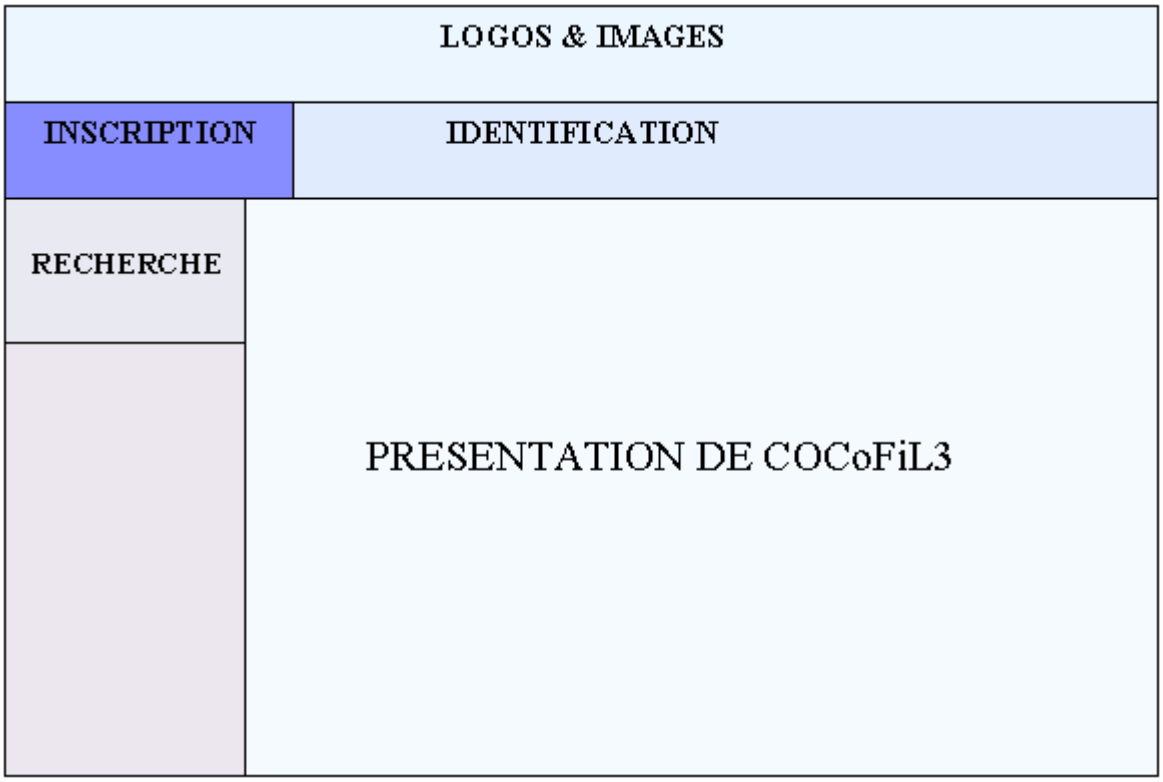


Figure 1:IHM abstraite de l'accueil

4.1.1 Zone LOGOS&IMAGES

Cette zone correspond à l'en-tête de l'application.

4.1.2 Zone INSCRIPTION

La zone d'inscription permet à l'utilisateur de s'inscrire.

4.1.3 Zone IDENTIFICATION

La zone d'identification permet à l'utilisateur de s'identifier. Cette zone offre deux champs qui permettent à l'utilisateur d'entrer son login et son mot de passe. Elle offre également un bouton qui permet de lancer la requête d'identification.

4.1.4 Zone RECHERCHE

Cette zone est destinée à la recherche. Elle comprend un champ dans lequel l'utilisateur tape sa requête et un bouton pour la lancer.

4.1.5 Zone PRESENTATION DE COCoFil3

Dans cette zone, nous présentons COCoFil3. Cette zone sert également de zone de travail pour un internaute. On y affichera la liste des films correspondant au résultat d'une recherche mais aussi le formulaire d'inscription.

IHM abstraite

L'IHM abstraite suivante présente les différents espaces de travails de la page d'accueil COCoFil3 pour un utilisateur identifié ainsi que leur agencement.

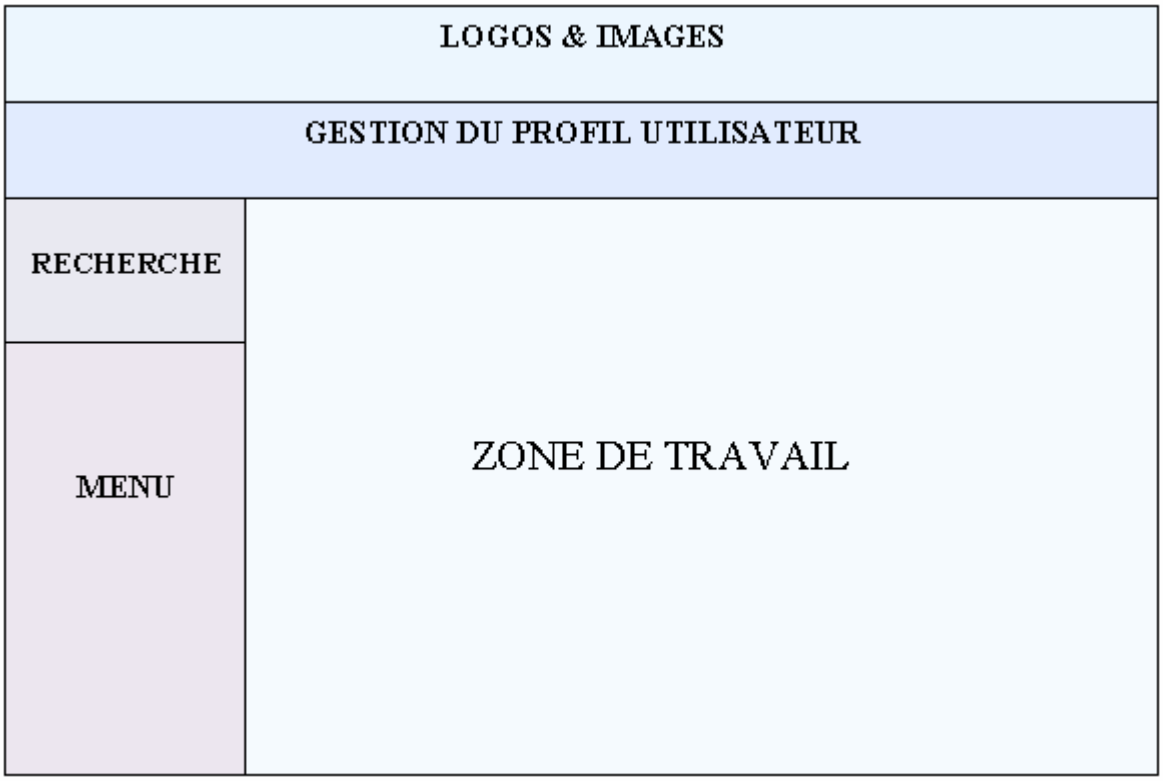


Figure 2:IHM abstraite

4.2.1 Zone GESTION DU PROFIL UTILISATEUR

Cette zone marque un message de bienvenue pour le membre ainsi que la date de sa dernière utilisation du système. En outre, elle lui permet de gérer son profil ou de se déconnecter.

4.2.2 Zone MENU

Le menu correspond aux fonctionnalités proposées au membre telle que la consultation des recommandations et des évaluations, la visualisation des communautés et la gestion du carnet d'adresses.

4.2.3 ZONE DE TRAVAIL

Elle permet l'affichage des résultats des requêtes de l'utilisateur.

IHM concrète de l'accueil

Ici, nous présentons l'application tel qu'elle va être.

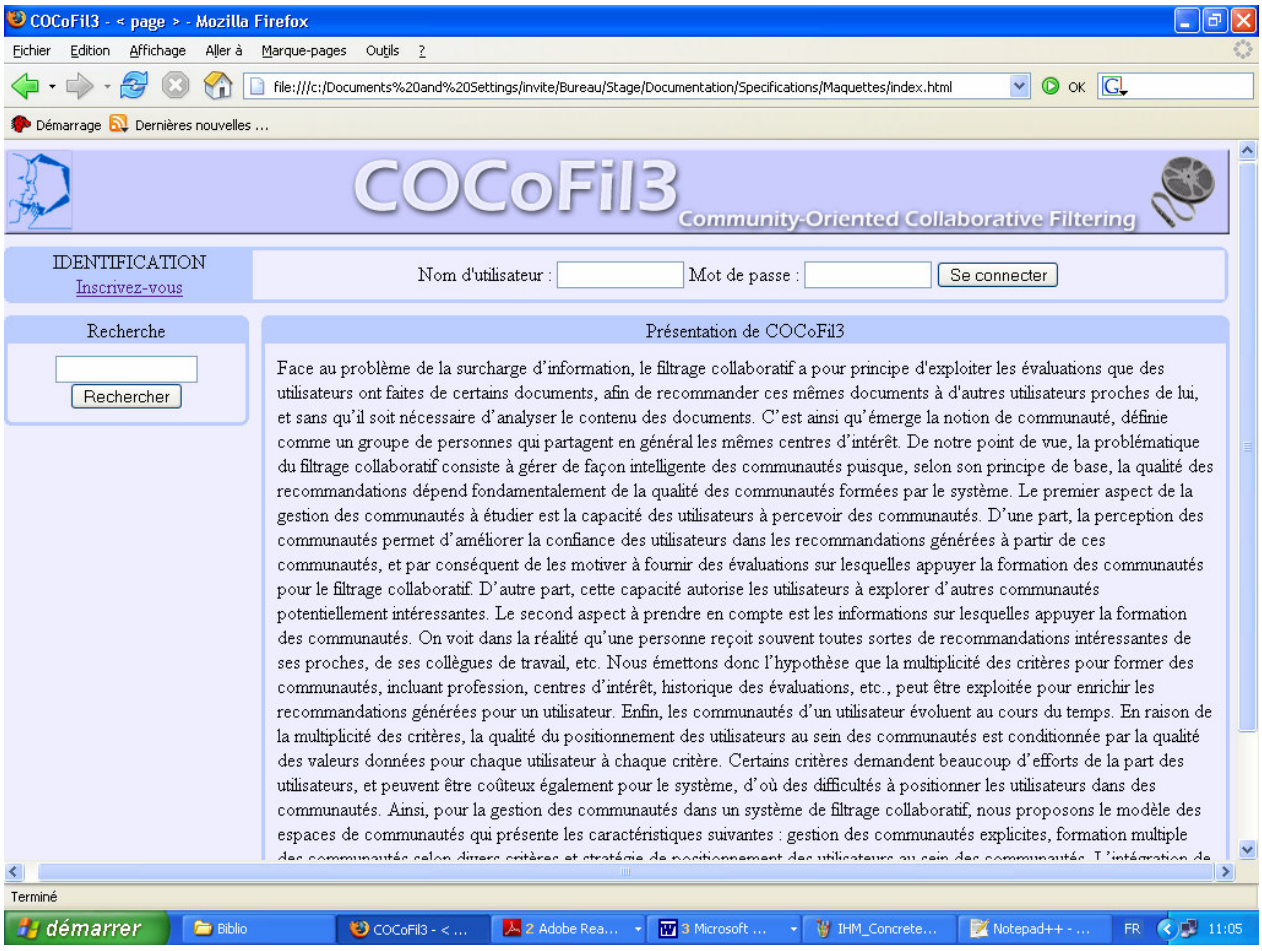


Figure 3:IHM concrète de l'accueil

Figure 4:IHM concrète de consultation des recommandations

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Plan de développement

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	14/05/2007
Version	2.0

Historique des versions

Version	Date	Modifications
1.0	08/02/2007	Début de la rédaction
1.1	02/04/2007	Modification suite aux remarques faites lors du 1er audit.
2.0	14/05/2007	Définition des incréments et conditions de passage d'un incrément à l'autre.

Sommaire

1. Introduction

But et portée du document

Le but de ce document est de présenter les moyens mis en œuvre, les taches nécessaires pour la réalisation du projet COCoFil3. On y trouve également le planning global et détaillé pour les délais à respecter ainsi que la gestion des risques.

Ce document s'adresse:

- à notre client : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

A propos du projet

Le projet COCoFil3 consiste à développer une application Web autour d'un système de recommandations basé sur les films. Au cours de précédents travaux, le noyau d'un système de recommandation orienté vers les communautés, nommé COCoFil2 a été développé par An-Te Nguyen. COCoFil3 vise à rendre ce système interactif et dynamique.

2. Organisation du projet

Structure de l'organisation

Le schéma ci-dessous représente les différents intervenants du projet, leur(s) rôle(s) ainsi que les flux de communication entre ces intervenants.

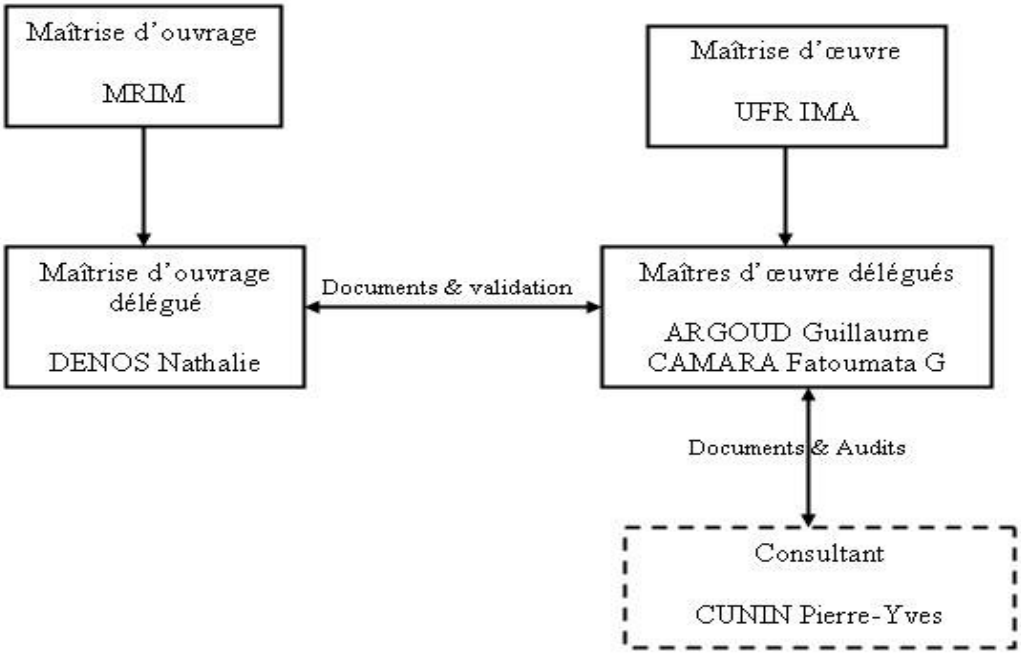


Figure 1: Organisation du projet

Ressources

Ressources humaines

Deux étudiants du Master 2 GI sont chargés de réaliser le projet. Le tableau ci-dessus récapitule leur disponibilité et leurs rôles sur la période allouée à la réalisation du projet.

Ressource	Mi-temps	Plein temps	Rôle
ARGOUD Guillaume	Janvier – Mars (2 jours/semaine)	Avril – Août	Chef de projet (Juillet - Août) Analyste programmeur Responsable qualité (Janvier - Juin) Responsable développement (Juillet - Août)
CAMARA Fatoumata Goundo	Janvier – Mars (3 jours/semaine)	Avril – Juin	Chef de projet (Janvier - Juin) Analyste programmeur Responsable qualité (Juillet - Août) Responsable développement (Janvier - Juin)

Ressources matérielles

Le projet se déroule dans le bureau des stagiaires du laboratoire CLIPS. Les deux machines suivantes sont mises à leur disposition :

- Berlioz1 : PC, Pentium IV (2.00 GHz de CPU et 512 Mo de RAM).
- Debussy5 : PC, Pentium IV (2.00 GHz de CPU et 512 Mo de RAM).
- Korsakov1 : machine linux Fedora Core 4 32 bits, Pentitm IV (4, 3.2 GHz, 2Go de RAM, disque 0.6 To)

Responsabilités

Chef de projet

Il doit vérifier la progression du projet. Il doit coordonner les tâches entre les différents membres de l'équipe de développement. Il est également chargé de prévoir la livraison du produit COCoFil3.

Analyste programmeur

Chaque analyste programmeur a pour but de mener à bien le projet. Chacun a la responsabilité de ce qu'il entreprend : conception, implémentation…

Responsable qualité

Sa principale tâche est le suivi de l'application du plan d'assurance qualité. Avant le début du projet, il est chargé de mettre en place la documentation. (Entête, présentation…). Il sera chargé de vérifier le code après la phase d'implémentation et de tests unitaires. (Assez grand nombre de commentaires, code clair…) Il est également chargé de l'archivage du projet. Tout au long du projet, il devra gérer les documents et leurs versions successives.

Responsable développement

Dès le début du projet, il est chargé d'apporter des solutions techniques, c'est à dire des logiciels (gratuits) permettant le développement du système COCoFil3). Il doit aussi apporter un avis sur les possibilités techniques, sur ce qui est réalisable et sur ce qui ne l'est pas. Si un problème technique se présente, il sera chargé de le résoudre. Il est chargé de la rédaction du manuel d'installation du logiciel.

3. Cycle de vie

Le cycle de vie d'un logiciel désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition.
Dans cette partie, nous prestons le cycle qui a été choisi pour le projet COCoFil3. Ce choix a été mûrement réfléchi et prend en compte toutes les exigences exprimées par le client.

3.1 Phase d'apprentissage

Le principe est de construire une mini application jetable autour du système existant (COCoFil2) pour permettre de comprendre les points durs (exigences, technologies).

Par ailleurs, cette phase d'apprentissage servira à maîtriser la base de données c'est-à-dire comprendre le rôle de chaque table et les relations entre elles.

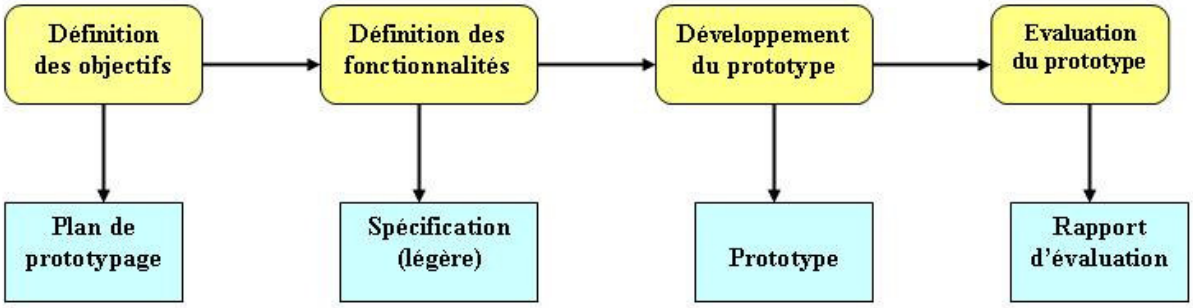
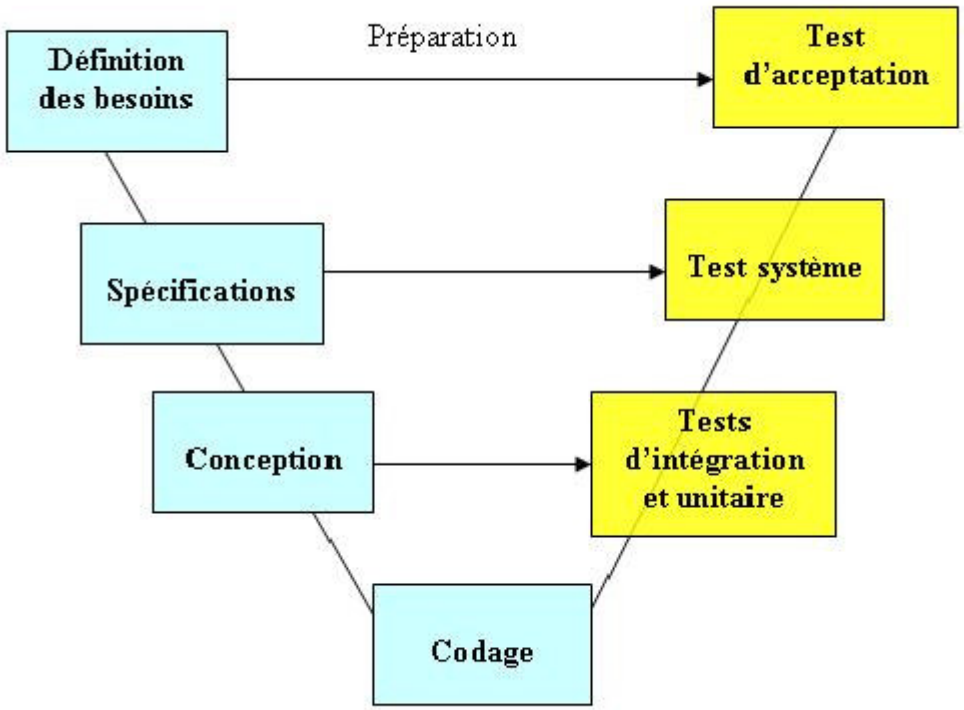


Figure 2 : Prototypage

3.2 Cycle de vie en V

Le principe de ce modèle est qu'avec toute décomposition doit être décrite la recomposition, et que toute description d'un composant est accompagnée de tests qui permettront de s'assurer qu'il correspond à sa description.
Ceci rend explicite la préparation des dernières phases (validation - vérification) par les premières (construction du logiciel), et permet ainsi d'éviter un écueil bien connu de la spécification du logiciel : énoncer une propriété qu'il est impossible de vérifier objectivement après la réalisation. Le but de ce cycle est donc de mettre en évidence l'activité de validation du produit. Ainsi cela permet une validation intégrée et planifiée dans le modèle.



3.3 Cycle de vie par incréments

Le cycle de vie par incréments a pour principe de diviser le projet en incréments. Un incrément est une sous partie fonctionnelle cohérente du produit final. Chaque incrément ajoute de nouvelles fonctions et est testé comme un produit final. Les incréments sont définis à priori (Classification des exigences).

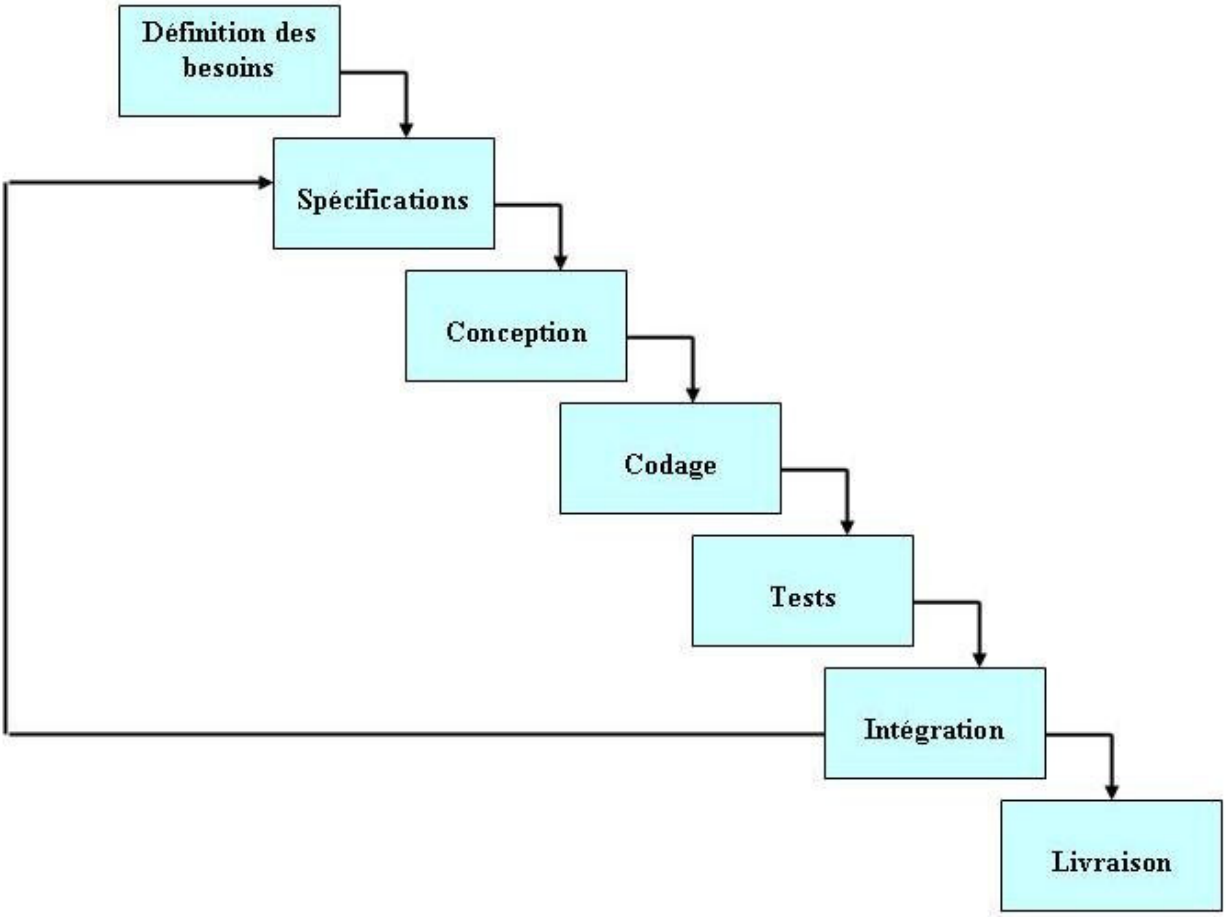


Figure 4: Cycle de vie par incréments

3.4 Choix de cycle de vie pour le projet COCoFil3

Pour le projet COCoFil3, nous allons dans un premier temps effectuer une phase d'apprentissage. Etant donné que nous partons d'un système existant, cette phase est un bon moyen de comprendre l'architecture de ce système. De plus, il permettra de mener une étude de faisabilité c'est-à-dire voir si COCoFil2 est adaptable aux besoins exprimés par le client.
En ce qui concerne le cycle de vie, nous avons choisi de conserver les avantages des deux cycles de vie présentés ci-dessus à savoir le cycle de vie en V et le cycle de vie par incréments. Les raisons de ce choix sont données ci-dessous :

- Le cycle de vie en V : l'une des forces de ce modèle est la définition des tests pour chaque étape du cycle de vie ; il nous a donc paru intéressant de garder cet avantage. Ainsi les tests d'acceptation seront spécifier à l'analyse des besoins, les tests d'intégration pendant la phase de conception…
- Cycle de vie par incréments : le développement incrémental réduit les risques d'échec du projet. Les problèmes sont découverts assez tôt, les parties importantes seront fournies en premier lieu (priorité des fonctionnalités) et le client peut à tout moment ajouter de nouvelles exigences.

Ainsi, le cycle de vie de base pour notre projet est un cycle de vie incrémental, afin de bien définir les tests, pour chaque phase du cycle de vie, une préparation des tests correspondants à la phase d'après le cycle de vie en V sera faite.

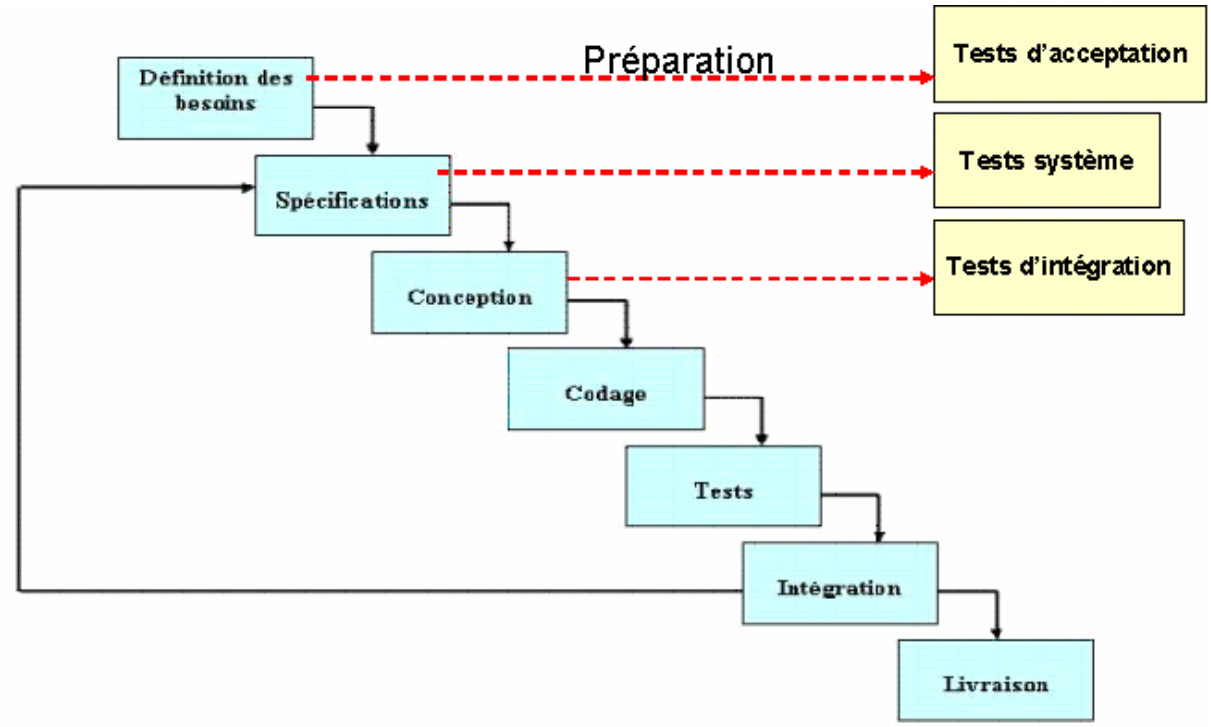


Figure 5: Cycle de vie du projet COCoFil3

3.5 Définition des incréments

Le système COCoFil3 sera réalise en 3 incréments. Les incréments seront définis par ordre de priorité croissante.
Le premier incrément consistera à mettre en place un système de recommandation de base. Il implémentera les fonctionnalités suivantes (toutes « **très importante** ») :

1. S'identifier
2. Fournir les données personnelles
3. Se déconnecter
4. Rechercher un film
5. Consulter les détails d'un film
6. Consulter ses recommandations
7. Consulter ses évaluations
8. Evaluer le film
9. Visualiser les communautés selon le critère « contenu »
10. Visualiser les communautés selon le critère « qualité »
11. Visualiser le profil d'un utilisateur

Le système existant COCoFil2 tourne sur la base de données « SmallMovieLens ». Le développement et le test du premier incrément se feront aussi avec cette même base de données. Un des objectifs du projet est de remplacer la base de données « SmallMovieLens » par la base de données « Intégration ». Ce passage de la base de données «SmallMovieLens » à la base de données «Intégration » est une étape très importante du projet. Elle se fera à la fin du premier incrément.

Le deuxième incrément ajoutera au premier les fonctionnalités liées à la gestion du carnet d'adresses et à la gestion de la liste des favoris. Les fonctionnalités suivantes seront implémentées :

1. Visualiser son profil des communautés
2. Modifier ses données personnelles
3. Consulter la liste des contacts
4. Ajouter un contact
5. Supprimer un contact
6. Bloquer un contact
7. Consulter sa liste des favoris
8. Ajouter le film à sa liste des favoris
9. Supprimer le contact de sa liste des favoris

Le développement et le test des deux premiers incréments se feront en local sur les machines « Windows » des deux membres du projet. A terme, le système devra tourner sur un des serveurs « Linux » de l'équipe appelé « Korskov1 ». La mise en place de l'application sur ce serveur se fera à la fin du deuxième incrément.

Le troisième incrément complétera les deux premiers. Les fonctionnalités sont les suivantes :

1. Recommander le film à un contact
2. Définir ses paramètres par défaut
3. Modifier ses paramètres par défaut
4. Créer un groupe de contacts
5. Supprimer un groupe de contacts

6. Ajouter un contact dans un groupe
7. Supprimer un contact d'un groupe
8. Déplacer un contact d'un groupe à l'autre
9. Commenter un film

Le passage d'un incrément à l'autre se fera sur validation du maître d'ouvrage.

4. Gestion du projet

Estimation de la durée et de l'effort

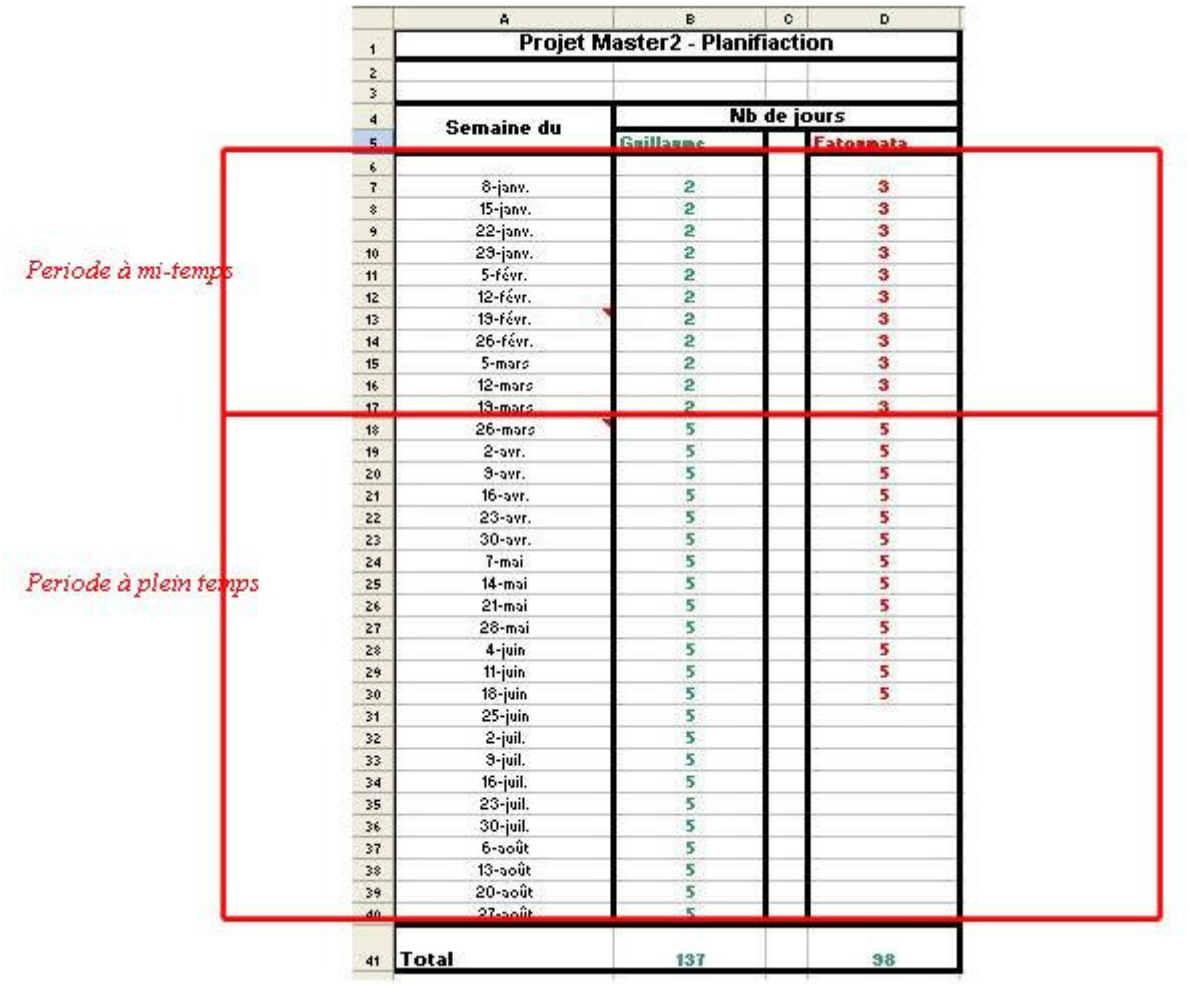
Estimation de la durée

Le projet se déroulera sur une durée totale de 8 mois (Janvier - Août) avec une période à temps partiel et une période à temps complet. Les deux membres de l'équipe en charge de la réalisation du projet n'ont les mêmes disponibilités sur toute las durée du projet. La disponibilité de chacun d'eux est donnée ci-dessous:

- ARGOUG Guillaume :
  - Durée temps partiel : 3 mois (Janvier - Mars) à 2 jours par semaine, soit 22 jours.
  - Durée temps complet : 5 mois (Avril - Août) à 5 jours par semaine, soit 115 jours.

- CAMARA Fatoumata Goundo
  - Durée temps partiel : 3 mois (Janvier - Mars) à 3 jours par semaine, soit 33 jours.
  - Durée temps complet : 3 mois (Avril - Juin) à 3 jours par semaine, soit 65 jours.

En effet, pendant les mois de juillet et août, CAMARA Fatoumata Goundo reste dans la même équipe donc, elle va être responsable qualité du projet COCoFil3.



	A	B	C	D
1	Projet Master2 - Planifiaction			
2				
3				
4	Semaine du	Nb de jours		
5		Guillaume		Fatoumata
6				
7	8-janv.	2		3
8	15-janv.	2		3
9	22-janv.	2		3
10	29-janv.	2		3
11	5-févr.	2		3
12	12-févr.	2		3
13	19-févr.	2		3
14	26-févr.	2		3
15	5-mars	2		3
16	12-mars	2		3
17	19-mars	2		3
18	26-mars	5		5
19	2-avr.	5		5
20	9-avr.	5		5
21	16-avr.	5		5
22	23-avr.	5		5
23	30-avr.	5		5
24	7-mai	5		5
25	14-mai	5		5
26	21-mai	5		5
27	28-mai	5		5
28	4-juin	5		5
29	11-juin	5		5
30	18-juin	5		5
31	25-juin	5		
32	2-juil.	5		
33	9-juil.	5		
34	16-juil.	5		
35	23-juil.	5		
36	30-juil.	5		
37	6-août	5		
38	13-août	5		
39	20-août	5		
40	27-août	5		
41	Total	137		98

Figure 6: Estimation de la durée du projet en jour

Estimation de l'effort

Le tableau ci-dessus (Fig. 6) nous donne la durée en jour par chaque membre projet. De ces données, nous déduisons l'effort total sur le projet :

137 + 98 = 235 hommes/jours

En considérant qu'un homme/mois est égal à 21 jours, l'effort total en homme/mois est d'environ **11,1**.

Application du modèle Cocomo de base

Type du projet

Notre projet est de type « ***semidetached*** » car le périmètre fonctionnel du système COCoFil3 reste déterministe bien que le nombre de tests doive être assez important.
De plus les contraintes sont liées au développement sont très faibles et précises non seulement d'un point de vue logiciel mais aussi matériel.

Mode	Effort (hmois)	Durée (mois)
Organic	$2,4\ KDSI^{1,05}$	$2,5\ Effort^{0,38}$
Semidetached	$3,0\ KDSI^{1,12}$	$2,5\ Effort^{0,35}$
Embedded	$3,6\ KDSI^{1,20}$	$2,5\ Effort^{0,32}$

Figure 7: Equation du modèle de base de Cocomo

Le type de projet ainsi défini, nous calculons la taille du logiciel. D'après le modèle Cocomo, pour un projet de type « ***semidetached*** » nous avons :

- $E = 3.0\ KDSI^{1,12}$
- $KSDI = (E/3)^{1/1,12}$

Dans le cadre de notre projet, la taille du logiciel sera alors :

$$(11,1/3)^{1/1,12} \approx 3200\ DSI$$

Les équations du modèle Cocomo permettent de calculer la durée théorique du projet à partir de l'effort, ce qui nous donne une durée théorique d'environ **5,8 mois**.

Compte tenu de l'infériorité de la durée (5,8 mois) par rapport à la durée réelle (8 mois), nous pouvons conclure que la taille du logicielle sera légèrement supérieure à 3200 DSI.

Répartition de l'effort par étape du cycle de vie

En se basant sur la répartition de l'effort selon le modèle Cocomo (Fig. 8), la répartition de l'effort pour le projet COCoFil3 est la suivante :

Etape du cycle de vie	Pourcentage	Effort (homme/mois)
Analyse des besoins	7	0,77
Spécifications	17	1,88
Conception	26	2,88
Codage et tests unitaires	35	3,88
Intégration et tests	22	2,44
Total	100	11,1

Mode	Etape	2 KDSI	8 KDSI	32 KDSI	128 KDSI	256 KDSI
Organic	Analyse de besoins	6	6	6	6	
	Spécification	16	16	16	16	
	Programmation	68	65	62	59	
	Conception	26	25	24	23	
	Codage et test unitaire	42	40	38	36	
	Intégration et test	16	19	22	25	
Semidetached	Analyse de besoins	7	7	7	7	7
	Spécification	17	17	17	17	17
	Programmation	64	61	58	55	52
	Conception	27	26	25	24	23
	Codage et test unitaire	37	35	33	31	29
	Intégration et test	19	22	25	28	31
Embedded	Analyse de besoins	8	8	8	8	8
	Spécification	18	18	18	18	18
	Programmation	60	57	54	51	48
	Conception	28	27	26	25	24
	Codage et test unitaire	32	30	28	26	24
	Intégration et test	22	25	28	31	34

Figure 8: Répartition de l'effort par étape du cycle de vie

(Modèle Cocomo de base)

Répartition de la durée par étape du cycle de vie

En se basant sur la répartition de la durée selon le modèle Cocomo (Fig. 8), la répartition de la durée pour le projet COCoFil3 est la suivante :

Etape du cycle de vie	Pourcentage	Durée (jour)
Analyse des besoins	18	42,3
Spécifications	25	58,75
Conception, codage et tests unitaires	52	122,2
Intégration et tests	23	54,05
Total	100	235

Mode	Etape	2 KDSI	8 KDSI	32 KDSI	128 KDSI	256 KDSI
Organic	Analyse de besoins	10	11	12	13	
	Spécification	19	19	19	19	
	Programmation	63	59	55	51	
	Intégration et test	18	22	26	30	
Semidetached	Analyse de besoins	16	18	20	22	24
	Spécification	24	25	26	27	28
	Programmation	56	52	48	44	40
	Intégration et test	20	23	26	39	32
Embedded	Analyse de besoins	24	28	32	36	40
	Spécification	30	32	34	36	38
	Programmation	48	44	40	36	32
	Intégration et test	22	24	26	28	30

Figure 9: Répartition de la durée par étape du cycle de vie

(Modèle Cocomo de base)

5. Planification

Planning prévisionnel global

GANTT project

Nom	Date de début	Date de fin	Durée
Analyse des besoins	08/01/07	15/03/07	48
phase d'apprentissage	27/02/07	06/04/07	28
Spécifications (1)	09/04/07	17/04/07	6
Conception (1)	18/04/07	27/04/07	7
Codage (1)	30/04/07	18/05/07	14
Tests (1)	14/05/07	18/05/07	4
Spécifications (2)	21/05/07	25/05/07	4
Conception (2)	28/05/07	01/06/07	4
Codage (2)	04/06/07	22/06/07	14
Tests (2)	18/06/07	29/06/07	9
Spécifications (3)	02/07/07	10/07/07	6
Conception (3)	11/07/07	19/07/07	6
Codage (3)	23/07/07	18/08/07	20
Tests et Intégration	20/08/07	30/08/07	8
Livraison	31/08/07	01/09/07	1

Figure 10:Planning prévisionnel global (1)

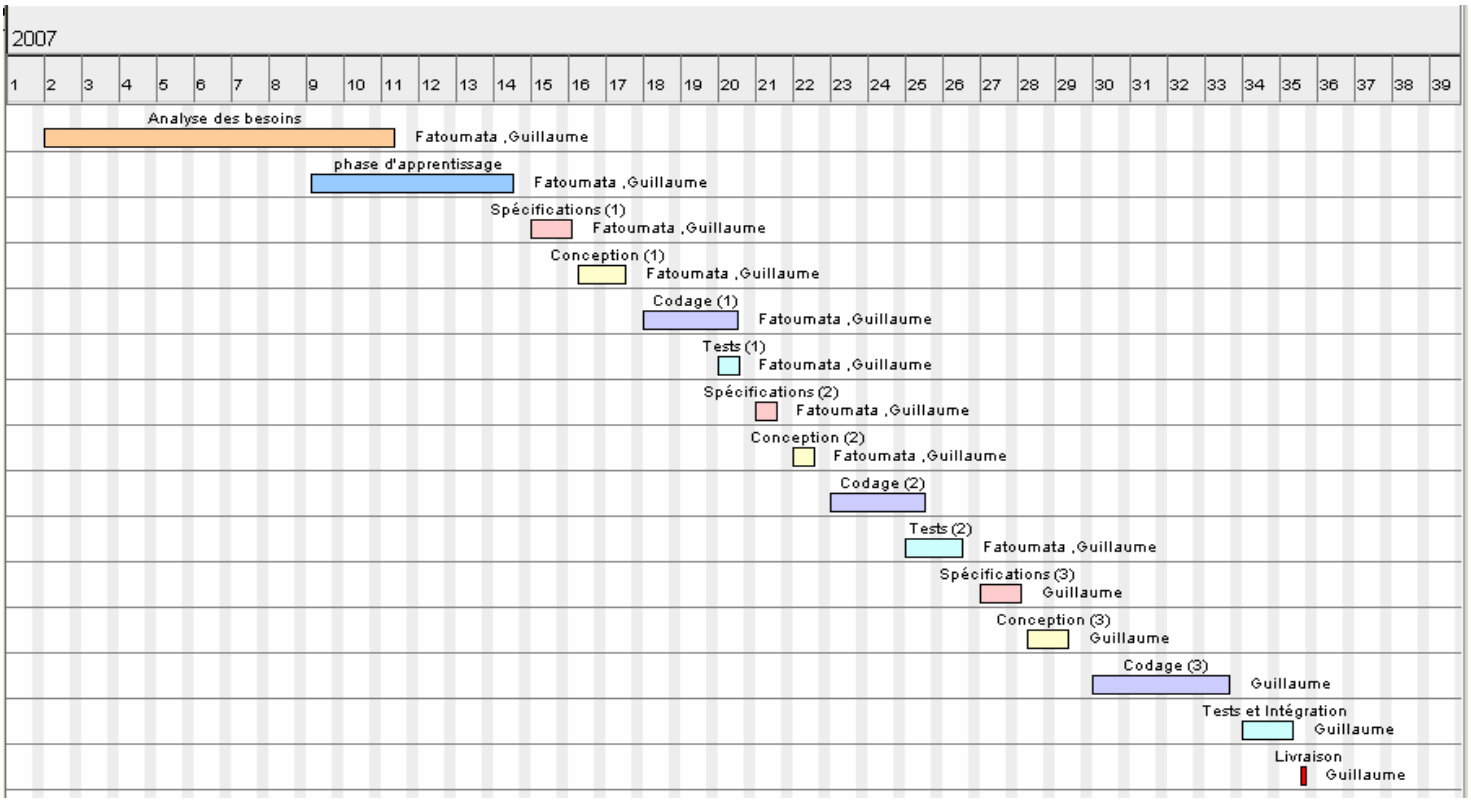
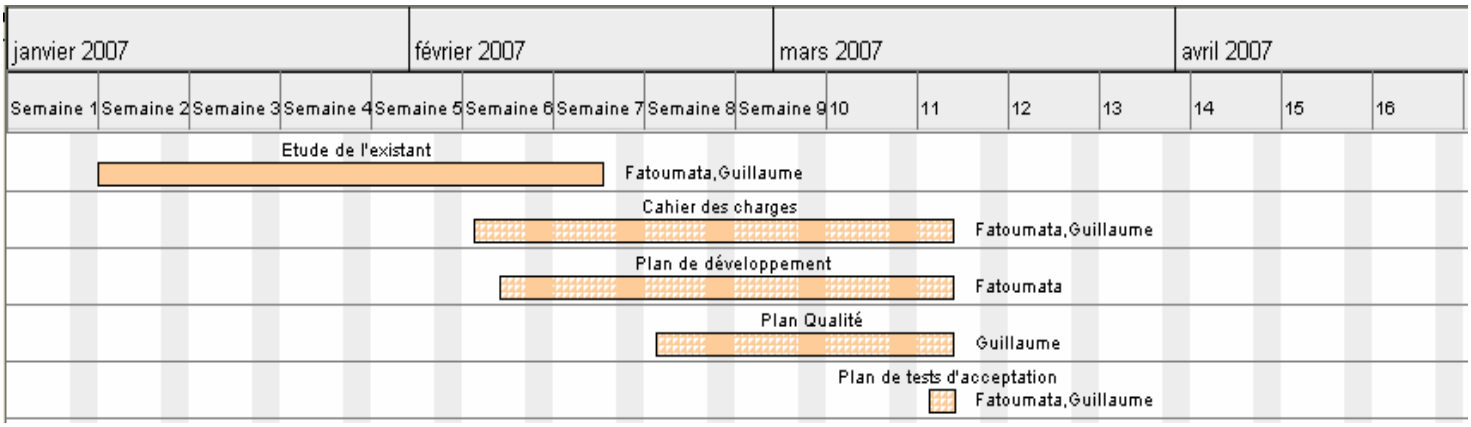


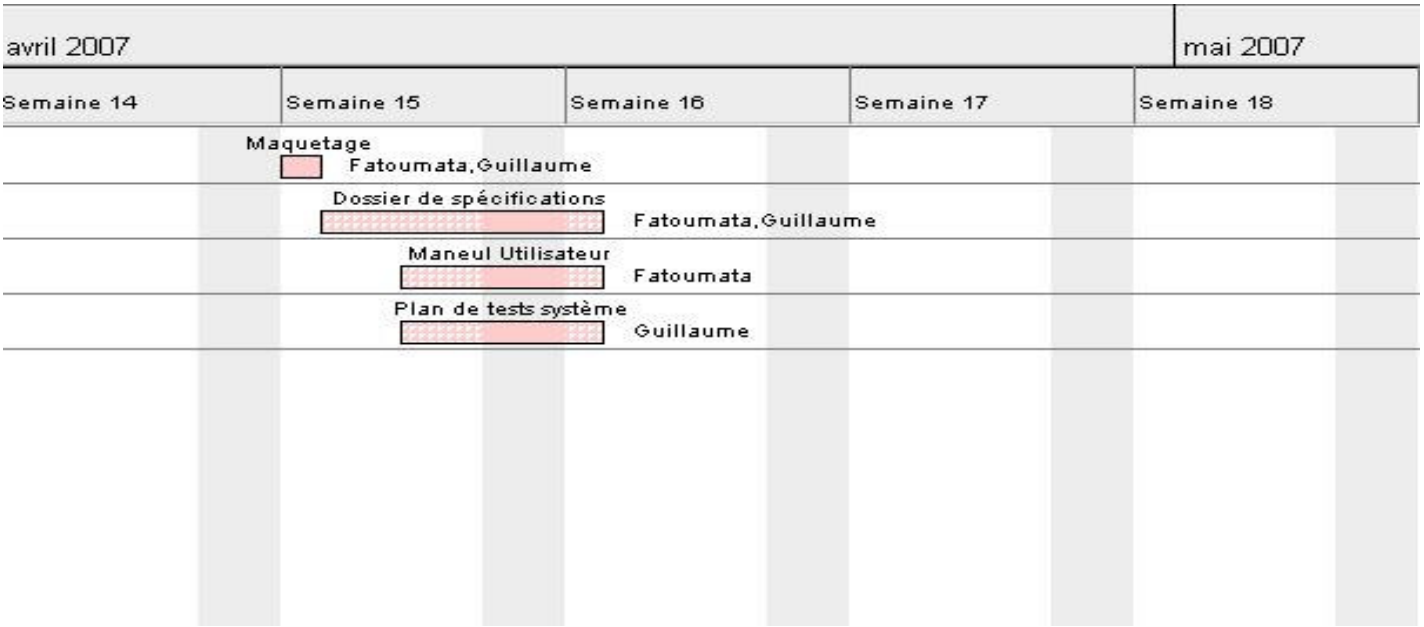
Figure 11:Planning prévisionnel global (2)

Planning prévisionnel détaillé

Analyse des besoins



Spécifications



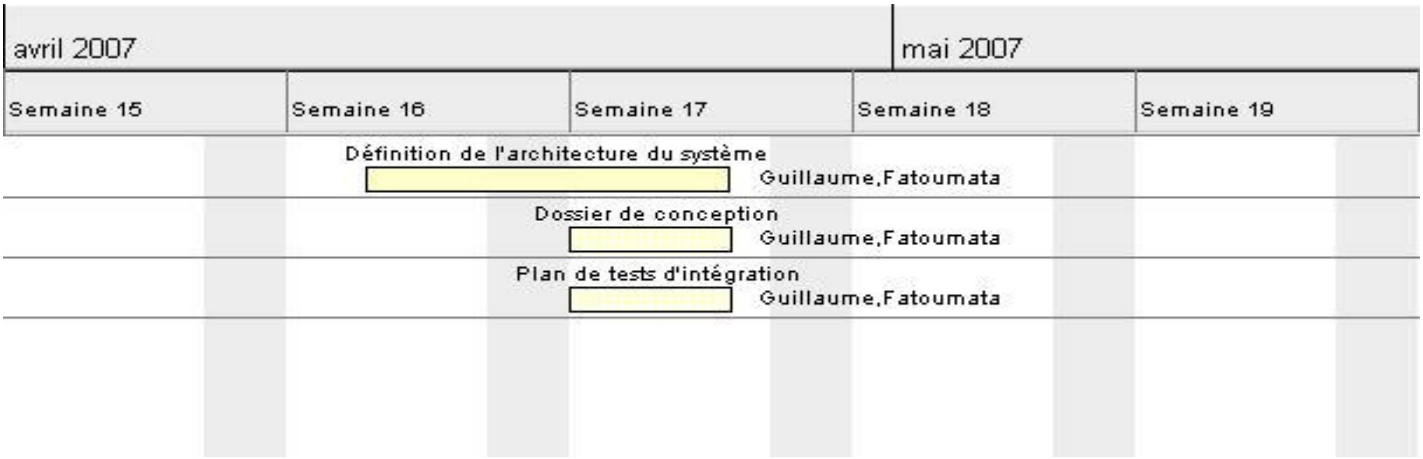
Le système COCoFil3 sera réalisé de façon incrémentale en itérant sur les étapes de spécification, conception, codage et tests. Le diagramme de Gantt ci-dessus donne le planning détaillé de l'étape de spécification pour le premier incrément.

Le diagramme ci-dessous donne de façon approximative le nombre de nécessaire à la réalisation de chaque activité de l'étape de spécification pour le deuxième et le troisième incrément.

GANTT project

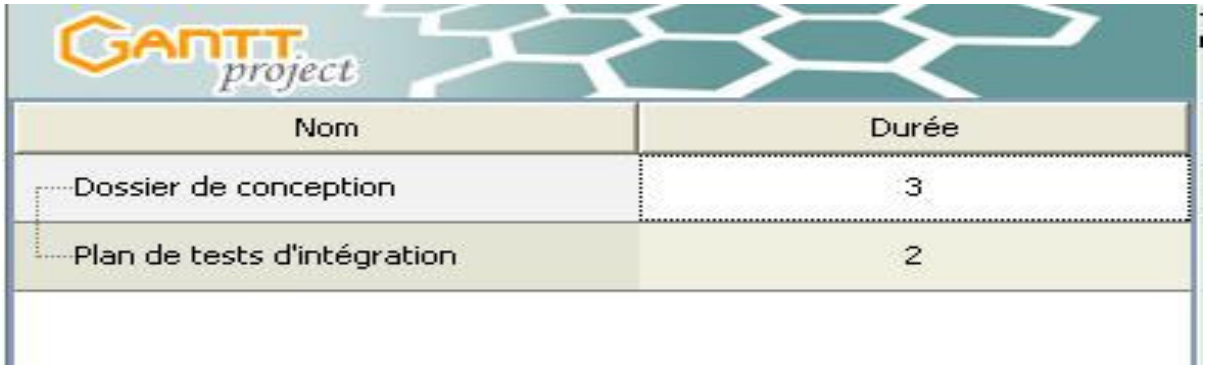
Nom	Durée
Maquettage	1
Dossier de spécifications	5
Maneul Utilisateur	2
Plan de tests système	2

Conception



Le système COCoFil3 sera réalisé de façon incrémentale en itérant sur les étapes de spécification, conception, codage et tests. Le diagramme de Gantt ci-dessus donne le planning détaillé de l'étape de conception pour le premier incrément.

Le diagramme ci-dessous donne de façon approximative le nombre de nécessaire à la réalisation de chaque activité de l'étape de conception pour le deuxième et le troisième incrément.



Codage et Tests

Les activités des étapes de codage et de test à savoir : le dossier de conception détaillée, le plan de tests unitaires et les rapports de test seront faites tout au long des phases correspondantes.

6. Gestion des risques

Risque	Solution	Probabilité
Compréhension du code du système existant (COCoFil2)	Effectuer une phase d'apprentissage afin de « jouer » avec le code.	Moyenne
Exploitation et adaptation du système existant aux besoins du client	Utilisation d'un bouchon pour simuler le comportement attendu.	Forte
Les deux membres du projet n'ont pas la même disponibilité	Définition des incréments de façon « équitable ».	Forte
Le produit final ne correspond pas aux attentes du client	Réunion hebdomadaire avec le client.	Faible

Equipe MRIM
Laboratoire LIG-IMAG
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Plan d'assurance qualité logiciel

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	14/04/2007
Version	2.0

Historique des versions

Version	Date	Modifications
1.0	20/02/2007	Début de la rédaction
1.1	21/03/2007	Corrections suite à l'audit 1 Ajout des mesures des critères de qualité
2.0	14/04/2007	Modifications suite au changement des critères de qualité du facteur maintenabilité. Ajout de la configuration des versions.

Sommaire

1. Introduction

But et portée du document

Le but de ce document est de spécifier les procédures mises en œuvre afin d'assurer la qualité du logiciel à réaliser. On y retrouve aussi les différents acteurs qui interagissent dans le cadre du projet avec leur rôle respectif.

Le projet se déroule au sein d'un laboratoire de l'IMAG : le LIG (Laboratoire d'Informatique de Grenoble) dans l'équipe MRIM (Modélisation et Recherche d'Information Multimédia). Le projet consiste en la réalisation d'une application Web autour des recommandations de films grâce à un système de filtrage collaboratif basé sur les communautés.

Sont spécifiés également, dans le présent document, les méthodes, outils, règles, normes et styles de programmation adoptés, ainsi que la liste des documentations produites.

Responsabilité

Les différents acteurs entrant en jeu dans le cadre de ce projet sont cités ci-dessous :

- Maîtrise d'ouvrage : LIG, BERRUT Catherine
  - Directrice de l'équipe MRIM
  - Rôle : Validation des documents.

- Maîtrise d'ouvrage déléguée: LIG, DENOS Nathalie dans l'équipe MRIM
  - Rôle : Encadrement du projet. Validation des documents.

- Maîtrise d' oeuvre : UFR IMA

- Maîtrise d'oeuvre déléguée: ARGOUD Guillaume et CAMARA Fatoumata Goundo
  - Etudiants de M2PGI
  - Chef de projet:
    - CAMARA Fatoumata Goundo (janvier à fin juin)
    - ARGOUD Guillaume (juillet à fin août)
  - Responsable développement logiciel :
    - CAMARA Fatoumata Goundo (janvier à fin juin)
    - ARGOUD Guillaume (juillet à fin août)
  - Responsable assurance qualité:
    - ARGOUD Guillaume (janvier à fin juin)
    - CAMARA Fatoumata Goundo (juillet à fin août)

- Consultant : CUNIN Pierre-Yves
  - Rôle : prodiguer des conseils quant à la gestion du projet, rédactions des documents, planification et organisation.

Les responsabilités de chaque personne seront détaillées dans les sections suivantes et tout au long de ce document.

2. Terminologie

Glossaire

COCFil : plateforme de filtrage collaboratif tournée vers l'innovation en matière de fonctionnalités interactives au travers de fonctionnalités orientées vers la notion de communauté et leur perception.

Abréviations et acronymes

P.Q.L	Plan Qualité Logiciel
C.d.C	Cahier des Charges
D.S.E	Dossier de Spécification externe
P.D.L	Plan de Développement Logiciel
P.T.A	Plan de Tests d'Acceptation
P.T.I	Plan de Tests d'Intégration
D.d.C	Dossier de Conception
P.T.U	Plan de Tests Unitaire
P.T.S	Plan de Tests Système
I.H.M	Interface Homme Machine

3. Organisation de la qualité

Responsabilités des participants

La maîtrise d'ouvrage

La maîtrise d'ouvrage peut être considérée comme un groupe de validation et d'approbation des choix de conception et de réalisation du projet.

La maîtrise d'ouvrage déléguée est sollicité à chaque prise de décision sur les différents modules de travail et se rassemble donc durant une réunion (hebdomadaire) avec le groupe de M2PGI.

La maîtrise d'œuvre

Le chef de projet joue le rôle de maître d'œuvre, c'est-à-dire qu'il a pour objectif de répartir et organiser les différentes tâches de travail entre les membres du groupe de M2PGI.

Le groupe de M2PGI

ARGOUD Guillaume et CAMARA Fatoumata Goundo constituent le groupe de M2PGI. Le groupe de M2PGI a pour charge de concevoir et de réaliser les différents travaux exigés par le projet ainsi que de produire les différents documents requis pour le projet.
Une personne assume le rôle de chef de projet (Maître d'œuvre).

L'établissement de la répartition des rôles semble indispensable pour instaurer une qualité dans l'organisation du développement du projet. Ceci permet de vérifier la conformité des tâches, que se soit sur la production de documents ou sur la réalisation des objectifs du projet.

Description détaillée des activités qualité

Lecture croisée

A chaque rédaction d'une nouvelle version d'un document, l'auteur demande à l'autre membre du groupe de relire son document. Ceci a pour objectif de vérifier si le contenu du document est bien conforme aux idées du groupe et d'éliminer toutes les ambiguïtés. A la fin d'une lecture, une réunion est organisée pour résoudre les éventuels problèmes relevés lors de la lecture.

Inspection externe

Des inspections impliquant d'autres personnes que les membres du groupe M2PGI seront menées sur les documents produits. En effet, le maître d'ouvrage délégué ainsi que le consultant seront sollicités afin de lire les documents produits et d'en confirmer la validité.

Audit

Les audits sont des réunions réunissant le consultant et le groupe du M2PGI. Ces réunions sont organisées sur l'initiative du chef de projet, en accord avec le consultant. Les personnes concernées par ces réunions recevront tous les documents une semaine à l'avance. Ces réunions seront au nombre de trois et auront pour but de s'assurer que les principes de base du Génie Logiciel sont correctement appliqués dans le projet.

Animation du P.A.Q.L.

Définition

Le responsable qualité du groupe de M2PGI a pour charge de définir le P.A.Q.L, en s'appuyant sur le modèle de Mc Call et l'expérience vécue de plusieurs autres projets de M2PGI.
La mise en place et le respect du P.A.Q.L sont sous la tutelle du responsable qualité, il est lui-même sous la tutelle du maître d'œuvre.
Le responsable qualité fait une inspection de tout ce qui a été produit avant chaque revue de livrable.

Rédaction

Le responsable qualité a pour travail de rédiger la version 1.0 du P.A.Q.L, en suivant les directives décidées avec le groupe de M2PGI.

La version 1.0 est distribuée à l'ensemble du groupe. Après lecture du document, le groupe décide ensemble des corrections à apporter au P.A.Q.L. Le responsable qualité doit alors modifier le document en tenant compte des corrections apportées par le groupe. Ensuite les modifications qui viendront se rajouter par la suite seront répertoriées dans le bilan qualité.

Validation

Le chef de projet et la maîtrise d'ouvrage doivent valider le document selon la procédure suivante :

- Remise du document en attente de validation au maître d'ouvrage délégué.
- Les remarques sont faites à l'oral et une trace écrite figure sur le document remit précédemment.
- Une nouvelle rédaction du document est effectuée en prenant en compte les remarques.
- La validation du document se fait après la nouvelle rédaction.
- Si le document n'est pas validé, on recommence la procédure jusqu'a ce qu'il le soit.

4. Démarche qualité

Exigences qualité

Facteurs qualité

Dans ce chapitre est décrit l'ensemble des facteurs qualité exigés par le maître d'ouvrage. Ils correspondent à la définition donnée dans le modèle de Mc Call. Ces choix ont été pris lors des premières réunions avec le maître d'ouvrage délégué et les raisons des choix sont expliquées ci-dessous.

Rappel des facteurs de qualité prioritaires présentés dans le cahier des charges :

- Utilisabilité : Qualité du logiciel à proposer une interface facile à apprendre et à utiliser.
- Maintenabilité : Capacité de pouvoir maintenir de manière cohérente et à moindre coût. Un des critères choisi pour ce facteur est la modularité (forte cohésion, faible couplage). Afin d'atteindre une forte cohésion, nous avons opté pour cohésion fonctionnelle ainsi les fonctionnalités liées aux recommandations seront regroupées, celles liées aux utilisateurs ensemble…En ce qui concerne le couplage, il sera de données, les modules vont échanger des données simples via leurs interfaces.

Critères qualité

Ce tableau résume les associations entre les facteurs qualité demandés à notre logiciel et les critères qui doivent être respectés.

Facteur	Critère	Mesure
Utilisabilité	Facilité d'apprentissage et d'utilisation	Les mesures seront faites à partir de questionnaires soumis aux utilisateurs après leur utilisation du système.
	Bonne compréhension des entrées/sorties (communicativeness)	
		Le temps de réponse moyen

	Performance : temps de réponse	à une requête utilisateur doit être de 10 secondes au maximum.
Maintenabilité	Modularité : faible couplage	L'outil Metrics sera utilisé pour mesurer le degré de couplage entre modules.
	Simplicité	Une classe doit compter au maximum 500 lignes. Une fonction doit compter au maximum 150.

On donne au facteur utilisabilité une note d'importance de 9.5 sur 10 et une note de 9 sur 10 pour la maintenabilité.

Autres facteurs qualité

Parmi les exigences non fonctionnelles citées dans le cahier des charges, des aspects ergonomiques sont à prendre en compte dans la qualité de l'application. Ces aspects ergonomiques visent à détailler les critères à prendre en compte afin de produire une application dont le critère « facilité d'apprentissage et d'utilisation » du facteur utilisabilité soit respecté. Ces critères seront donc fortement pris en compte dans l'élaboration du questionnaire soumis aux utilisateurs afin de s'assurer qu'ils ont été respectés. Par exemple, il sera demandé en combien de clics l'utilisateur a réalisé une opération afin de s'assurer que le guidage est suffisamment important pour garantir un nombre minimum de clics.

Evaluation de la qualité

Assurance qualité

L'assurance qualité est une structure regroupant un ensemble de dispositions permettant d'atteindre les objectifs. Elle assure la surveillance des facteurs et des critères de qualité. Le responsable qualité a la responsabilité de la mise en place de l'assurance qualité.

Qualité de la réalisation

La qualité dépend directement des critères jugés important. Le maître d'œuvre délégué et le responsable qualité doivent contrôler la qualité en évaluant les critères. Si des critères sont évalués en faute, le groupe de M2PGI doit tout mettre en œuvre pour remédier à ce défaut.

Estimation

Si le groupe du projet se retrouve dans une situation où il s'avère impossible de réaliser une mesure alors il procède à une estimation (ex : coût de la mesure trop important). En aucun cas cette estimation peut être considérée comme équivalente à une mesure, elle peut être au plus prise comme une supposition sur la qualité obtenue.
L'estimation des critères sera réalisée par le maître d'œuvre et le responsable qualité, ils donneront un indice de 1 à 10 pour noter la validité du critère.

Mesures

Ce chapitre définit les mesures pour chaque critère. Toute mesure est une note sur une échelle de 10.

Mesure de la facilité d'apprentissage et d'utilisation et de la communicativité

Nous choisirons un ensemble d'utilisateurs représentatifs pour utiliser l'application. Nous leur ferons remplir un questionnaire (cf. annexe) après leur utilisation de l'application. Ces questionnaires nous permettrons, après une analyse des réponses, d'évaluer le critère de communicativité et le critère de facilité d'apprentissage et d'utilisation grâce à un barème fixé préalablement. Chaque questionnaire aura une note sur 10 reflétant la communicativité et la facilité d'apprentissage et d'utilisation. La moyenne des notes de chaque critère donne la donne global pour ce critère.

Le questionnaire est composé de plusieurs petits scénarios couvrant la totalité des fonctionnalités. L'utilisateur exécute les scénarios est note ces résultats et observations.

Mesure de l'efficacité d'exécution

Les temps de réponse des systèmes interactifs, c'est-à-dire les délais entre les actions de l'utilisateur et le retour d'information des systèmes, sont des événements « stressants ». Des délais supérieurs à 10 secondes entraînent une perte d'intérêt. Les délais de réponse auraient de plus des effets sur la satisfaction et sur les performances des utilisateurs (productivité, etc.).

Le temps de réponse à une tâche d'un utilisateur doit donc être minimal. Néanmoins certains processus demande un temps d'exécution de plusieurs minutes. Le temps de réponse à une tâche « simple » de l'utilisateur (consultation du profil, du carnet d'adresses, …) devra être inférieur à 5 secondes. Une tâche pouvant entraîner l'exécution de processus long (consultation des cartes de communautés, des recommandations, …) devra avoir un temps de réponse inférieur à 10 secondes. De plus, les processus les plus long, comme le calcul de distance entre utilisateurs, devra se faire en temps « masqué » avant et non au moment de la demande de l'utilisateur.

Mesure de la modularité

La modularité est l'aptitude d'un logiciel à être composée de modules indépendants. Comme le langage de programmation utilisé dans ce projet est Java qui est un langage objet, la modularité sera calculé en fonction du couplage des packages.

Le couplage correspond au nombre de classes hors d'un package qui dépendent d'une classe dans le package.

Le couplage des packages permet de déterminer si le projet est bien conçu de manière modulaire. En effet plus la mesure de couplage des packages est faible, plus les packages sont indépendants.

On pose alors :

PrédicatCouplage = Couplage < 5

On calcul ensuite la note de modularité :

$$NoteModularité = 10 * \frac{NombreDePackagesVérifiantPrédicatCouplage}{NombreDePackagesTotal}$$

Pour valider ce critère, la note sur 10 obtenue devra être supérieur à 8.5.

Mesure de la simplicité

La simplicité est l'aptitude d'un logiciel à avoir un fonctionnement interne compréhensible facilement. Pour faciliter la compréhension, les normes de programmation devront être respectées. De plus la taille maximum d'une fonction est fixée à 150 lignes et la taille d'une classe à moins de 500 lignes. Au-delà, le logiciel devient trop complexe.

On pose alors :

$$PrédicatClasse = TailleClasse < 500$$

et

$$PrédicatMéthode = \begin{array}{c} NormesDeProgrammationRespectées \\ \& \\ TailleMéthode < 150 \end{array}$$

On calcul ensuite les deux notes suivantes :

$$NoteMéthodes = 10 * \frac{NombreDeMéthodesVérifiantPrédicatMéthode}{NombreDeMéthodesTotal}$$

$$NoteClasses = 10 * \frac{NombreDeClassesVérifiantPrédicatClasse}{NombreDeClassesTotal}$$

On peut enfin calculer la note d'utilisabilité qui vaut

$$NoteUtilisabilité = (NoteMéthodes + NoteClasses) / 2$$

Pour valider ce critère, la note sur 10 obtenue devra être supérieur à 8.

Détermination des métriques des facteurs

Lorsque tous les critères déterminants pour un facteur ont été mesurés ou estimés, le groupe du projet peut alors donner une valeur pour le facteur de qualité correspondant. Cette valeur correspond à la moyenne des valeurs des critères correspondant à un facteur donné.

Interprétation

Lorsque la valeur de ce facteur est obtenue on peut, grâce à la note d'importance du facteur et la formule suivante, obtenir un coefficient de satisfaction :

$$CoefficientSatisfaisabilité = 100 * \frac{MesureObtenue}{NoteImportance}$$

Exemple : avec une mesure obtenue de 7,5 et une note d'importance de 8, les objectifs sont remplis à 100 * (7,5 / 8) = 93%.

En fonction de la valeur du cœfficient obtenu, le chef de projet et le responsable qualité prennent des directives pour améliorer le coefficient de satisfaction. Pour cela, ils s'appuient sur le tableau suivant :

Coefficient de satisfaction	Bilan
Inférieur à 75%	Objectif non atteint.
Entre 75% et 90%	Objectif partiellement atteint. Un travail supplémentaire pourrait améliorer ce taux
Supérieur à 90%	Objectif atteint.

Disposition techniques

Nous décrivons ici certains détails de protocole de développement qui pourront nous aider à atteindre les objectifs des facteurs qualité.

Qualité de production

Règles de programmation

Il est indispensable de produire des programmes sources lisibles et compréhensibles par chacun des membres du groupe. Pour cela nous établissons des normes de codage, les règles suivantes sont à respecter :

- Les identificateurs, les variables, les commentaires doivent être explicites.
- Le nom des variables commence par une minuscule
- Le nom d'une méthode devra toujours commencer par une minuscule. Si ce nom est composé de plusieurs mots, chacun devra commencer par une majuscule (excepté le premier).
- Le nom d'une classe commence par une majuscule.

Contrôle

Le respect de ces normes est vérifié par le responsable qualité sur l'ensemble du code produit. Il peut, s'il juge que les normes n'ont pas été respectées, contraindre l'auteur à reprendre son code pour le mettre à niveau. Le contrôle portera sur le respect de la documentation concernant les commentaires du code, la signification des noms des variables employées, l'indentation du code.

Qualité des tests et essais

Un document est rédigé à la fin des tests d'un module. Ce document rapporte les observations faites lors de cette phase. Ce document est joint au dossier de test concerné, ce dossier fera l'objet d'une inspection.
Ces inspections réuniront le maître d'ouvrage délégué pour discuter des observations faites et de leurs influences sur la suite du projet.
La décision de poursuivre le projet ou de corriger les anomalies est prise par le chef de projet et le responsable qualité.

Qualité logistique

Le responsable qualité a pour rôle :

- D'établir une liste des tâches liées à la logistique à effectuer
- De s'assurer que toute les tâches de la liste ont bien été effectuées avant le commencement d'une activité

Qualité de la documentation

Les documents produits devront respecter certaines règles, ils devront entre autre comporter au moins les éléments suivants :

- Une page d'entête type
- Un historique des versions du document
- Une table des matières
- L'objet du document et le rappel du contexte
- Un corps de document

La numérotation des chapitres se fait de façon incrémentales : 1, 1.1, 1.1.1, etc.
Pour chaque titre ou type de texte est attribué une taille, police, couleur et style.
Chaque page du document (hormis la page de garde, l'historique des versions et la table des matières) possède une entête (rappelant le nom du document) et un pied de page (contenant le numéro de la page).

Documentation du code

La documentation du code devra faire apparaître les informations suivantes :

- Description de la classe ou du module
- Description des méthodes
- Précision des paramètres de sorties et d'entrées
- Explication des algorithmes utilisés
- Commentaires dans le code décrivant l'enchaînement des actions

Rappel du cycle de développement

Le cycle de vie du logiciel est décrit dans le plan de développement. Ceci est un rappel du cycle de vie que nous avons choisi :

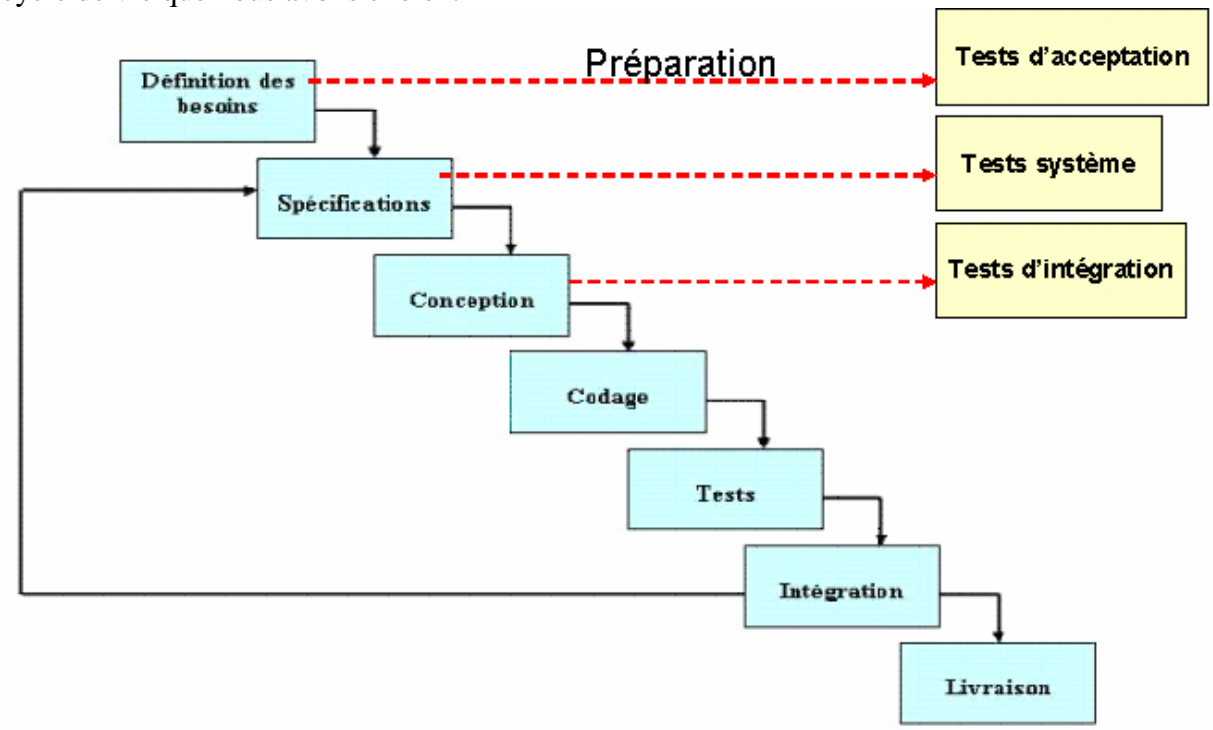


Figure 1 : Cycle de vie du projet COCoFil3

Le cycle de vie de base pour notre projet est un cycle de vie incrémental, afin de bien définir les tests, pour chaque phase du cycle de vie, une préparation des tests correspondants à la phase d'après le cycle de vie en V sera faite.

5. Activités qualité par phase

L'application des dispositions du P.A.Q.L. est contrôlée de façon permanente par le responsable qualité du projet. Chaque transition d'une phase à une autre est précédée par une revue de fin de phase entre le maître d'ouvrage déléguée et le groupe de M2PGI. Une fois l'accord du maître d'ouvrage obtenu, le chef de projet décide du passage à la phase suivante. Un compte-rendu est rédigé à la suite de cette réunion.

Les chapitres suivants décrivent les dispositions définies dans chaque phase.

Analyse des besoins

Activités d'environnement

- Etude de l'existant et état de l'art
- Prise en main et préparations des machines
- Préparation du premier audit

Produits nécessaire

- Sujet du projet

Activité de la phase

- Analyse des besoins et des exigences du maître d'ouvrage par des réunions.
- Rédaction du C.d.C, du P.D.L, du P.A.Q.L et du P.T.A.

Produits

- Rédaction du C.d.C, du P.D.L, du P.A.Q.L et du P.T.A

Conditions de passage à la phase suivante

- Inspection : pour le C.d.C un compte rendu écrit fournit les défauts relevés par le maître d'ouvrage délégué permettant ainsi au groupe de M2PGI d'apporter les corrections nécessaires.
- Audit : pour le C.d.C, le P.D.L, le P.A.Q.L et le P .T.A, une réunion entre le consultant et le groupe de M2PGI permet d'écarter les éventuelles erreurs.
- Revue de vérification et validation : pour le C.d.C, le P.D.L, le P.A.Q.L et le P .T.A, une approbation du maître d'ouvrage sera suivit d'une validation par le chef de projet.

Spécifications externes

Activités d'environnement

- Présentation des maquettes lors des réunions
- Etude des applications concurrentes
- Mise à jour du planning

Produits nécessaire

- C.d.C et P.A.Q.L

Activité de la phase

- Réalisation de maquettes
- Rédaction du D.S.E et du P.T.S

Produits

- D.S.E et P.T.S
- Maquettes

Conditions de passage à la phase suivante

- Vérification et validation des maquettes, du D.S.E, du P.T.S et du manuel utilisateur.
- Lorsque la phase de spécifications externes de chaque module finit et validé, on passe à l'analyse des besoins du module suivant. Une fois l'ensemble terminé on passe à la conception globale.

Conception globale

Activités d'environnement

- Choix et installation des environnements de programmation et des logiciels utilisés
- Mise à jour du planning
- Déterminer la procédure d'intégration des différents modules

Produits nécessaire

- Le D.S.E, le P.Q.L, le C.d.C, et le P.A.Q.L

Activité de la phase

- Rédaction du D.C.G et du P.T.I
- Mise à jour du Manuel Utilisateur

Produits

- D.C.G et P.T.I

Conditions de passage à la phase suivante

- Audit : validation du D.S.E, du D.d.C et du P.T.I par le consultant et le groupe de M2PGI

Conception détaillée, codage et tests unitaire

Activités d'environnement

- Mise à jour du planning

Produits nécessaire

- P.Q.L, C.D.C, D.S.E et D.C.G

Activité de la phase

- Conception du P.T.U
- Codage
- Exécution du P.T.U
- Corrections des erreurs rencontrées
- Rédaction du Manuel Utilisateur

Produits

- D.C.D, Manuel Utilisateur et P.T.U

Conditions de passage à la phase suivante

- Succès des tests unitaires
- Vérification de la conformité entre les spécifications et les implémentations

Test d'intégration

Activités d'environnement

- Mise à jour du planning
- Création d'éventuels bouchons et lanceurs

Produits nécessaire

- P.T.I

Activité de la phase

- Exécution des tests prévus par le P.T.I

Produits

- Compte rendu des tests

Conditions de passage à la phase suivante

- Réussite des tests d'intégration

Test système, d'acceptation et recette

Activités d'environnement

- Mise à jour du planning

Produits nécessaire

- D.S.E, P.T.S et P.T.A

Activité de la phase

- Exécution des plans de tests

Produits

- Compte rendu des tests
- Rapport d'évaluation de l'I.H.M et de la qualité
- Document de bilan.

Conditions de passage à la phase suivante

- Tests Système et d'acceptation terminés et positifs
- Recette validée par le maître d'œuvre délégué

Procédures de rattrapage

Si une phase de développement ne respecte pas le planning qui lui a été affecté alors le groupe du M2PGI doit prendre les mesures nécessaires pour prévoir une période de rattrapage afin de terminer au plus tôt, sans altérer la qualité.

6. Documentation

Documents de gestion de projet

Plan Assurance Qualité Logiciel

Le contenu de ce document livrable doit au moins traiter les points suivants :

- Responsabilités
- Organisations
- Démarche développement
- Documentation
- Gestion de la configuration
- Gestions des modifications
- Méthodes outils et règles
- Critères de qualité
- Suivi de l'application du plan d'assurance qualité logiciel

La charge de la rédaction et de la vérification de l'application de ce document est attribué au responsable qualité.

Plan de développement du Logiciel

Ce document livrable est constitué du planning prévisionnel du projet. Il doit contenir la description des aspects suivants :

- Durées et dates du projet
- Décomposition du temps par tâche (avec estimation d
- Planning général et détaillé

Le document de bilan

Ce document est un document livrable, on y trouve toutes les conclusions qui ont été tirées sur le développement du logiciel une fois le temps imparti, au projet, terminé. Les comptes rendus de réunion serviront également de support à l'élaboration de ce document.
Ce document abordera les quatre points importants suivants :

- Mesure de l'application des méthodes au cours du développement
- Bilan du suivi qualité
- Bilan du plan de développement
- Les modifications demandées qui n'ont pu être réalisées dans le temps imparti au projet.

Le responsable qualité et le chef de projet ont à charge ce document.

Documents techniques de réalisation

Cahier des Charges

Ce document livrable a pour but de décrire de manière claire et précise quels sont les besoins de la maîtrise d'œuvre. La description des points suivants doit faire parti du document :

- Présentation de l'existant et objectifs du projet
- Acteurs et cas d'utilisations du système

- Exigences fonctionnelles
- Exigences non fonctionnelles (facteurs et critères de qualité)

Tout le groupe est responsable de ce document.

Dossier de Spécifications Externes

Ce document a pour but de formaliser les besoins de la maîtrise d'ouvrage qui ont été exprimées dans le cahier des charges et de conclure la phase de spécification externe. Il doit contenir les points suivants :

- Cas d'utilisation
- Données en entrée et en sortie
- Interface utilisateur

Tout le groupe est responsable de la rédaction de ce document. Le chef de projet doit le faire respecter.

Dossier de conception globale

Ce document a pour but de décrire l'architecture des incréments à un niveau d'abstraction élevé. Il doit aborder tous les points suivants :

- Décomposition fonctionnelle en sous systèmes.
- Architecture fonctionnelle en sous systèmes.
- Diagramme de classe.

Tout le groupe est responsable de la rédaction de ce document. Le chef de projet doit le faire respecter.

Dossier de conception détaillé

Ce document a pour but de décrire et détailler chaque élément de la structure présenté dans le dossier de conception globale. Il doit aborder tous les points suivants :

- Spécifications fonctionnelles détaillées pour chacun des sous systèmes.
- Architecture fonctionnelle détaillée pour chacun des sous systèmes.

Tout le groupe est responsable de la rédaction de ce document. Le chef de projet doit le faire respecter.

Plan de tests d'Acceptation du Logiciel

Ce document décrit les scénarios, les critères et les moyens à mettre en œuvre pour éprouver le fonctionnement du logiciel. Il est livré au même moment que le dossier de spécifications externes. Il doit contenir au minimum :

- Spécification des scénarios de test.
- Moyens et critères de vérification.
- Critères de réussite.

Le groupe a la responsabilité de ce document.

Réalisation de Tests d'Acceptation du Logiciel

Ce document regroupe les résultats de l'application du P.T.A, il est sous la responsabilité de la maîtrise d'œuvre déléguée.

Plan de Tests d'Intégration du Logiciel

Ce document est livrable en même temps que le D.C.G et doit au minimum contenir :

- La spécification des scénarios de test
- Les moyens et critères d'intégration
- Les critères de réussite

Le groupe a la responsabilité de ce document.

Réalisation de Tests d'Intégration du Logiciel

Le responsable qualité doit rédiger ce document livrable. Il contient les résultats de l'application du P.I.L.

Plan de Tests Unitaire

Ce document est livrable en même temps que le D.C.D et doit contenir au minimum :

- Les unités à tester
- Les spécifications des scénarios de test
- Les critères de réussites

Le groupe a la responsabilité de ce document.

Réalisation de Tests Unitaire

Le responsable qualité doit rédiger ce document livrable. Il contient les résultats de l'application du Plan de tests unitaires.

Manuel Utilisateur

Ce document livrable regroupe la description des modalités d'utilisation du logiciel. Il doit expliquer les différentes fonctionnalités à l'utilisateur.

Le groupe est responsable de ce document.

Dossier d'installation

Ce document livrable regroupe la description des modalités d'installation du logiciel. Il doit expliciter les points suivants

- Installation de l'application
- La mise en œuvre
- La maintenance du système
- Arrêt du système

7. Gestion de configuration

Gestion des documents

Style commun

Tous les documents produits possèderons un style en commun :

- Une page de garde : comportant le nom du projet, l'adresse du laboratoire, les logos, le nom des membres de l'équipe et les responsables du projets, le nom du document, la date de dernière modification et le numéro de version du document.
- Une en-tête : comportant le logo de l'équipe MRIM et le nom du document.
- Un pied de page : comportant le numéro de la page (sauf pour la page de garde)
- Une table des matières : une mise en place d'une hiérarchie de titres sous le logiciel de traitement de texte permet de générer la table automatiquement.

Numéro de version

Les noms des documents sont de la forme Nom_VX.Y.doc où Nom correspond au nom du document et X et Y permettent de définir le numéro de version du document. Le numéro Y est un incrément comptabilisant les modifications minimes du document. Le numéro désigne lui un incrément comptabilisant le passage à une version supérieur du à des modifications majeures (suite à une réunion avec le responsable ou suite à un audit par exemple). Le numéro Y est réinitialisé lors de l'incrémentation de X.

Sauvegarde

Lorsque l'un des deux membres du projet produit une nouvelle version d'un document, il l'envoie à l'autre membre du projet et le dépose sur la page de suivi du projet :
http://www-clips.imag.fr/mrim/Prototypes/Cocofil/suivi/

Gestion du code source

La configuration est l'ensemble des éléments suivants : code source exécutable, outils de développement et de test utilisés, documents et données.

L'outil utilisé pour gérer le travail collaboratif sous environnement Eclipse est CVS (Concurrent Versions System).

Cet outil permet de partager le code source d'un logiciel et d'intégrer facilement les modifications de fichiers d'un développeur avant de les mettre à disposition des autres utilisateurs. Cet outil permet également de garder un historique de toutes les modifications et les différentes versions du projet. Il permet ainsi de revenir à une version précédente en cas de problème.
Le serveur CVS utilisé est celui de l'équipe MRIM.

8. Méthode et outils

Méthodes

Dans les différents plannings, nous utiliserons la méthode CoCoMo et celle de GANTT afin d'estimer les ressources nécessaires au projet (pour une définition plus précise, se référer au

plan de développement). Le modèle CoCoMo permet de fixer les efforts pour les différentes phases de développement.

Outils

Outils de rédaction de documents

La documentation du projet est réalisée grâce à :

- Microsoft Office 2003
- Gantt project 2.0 (pour le planning)

Outils de développement

L'IDE Eclipse sera utilisé pour le développement Java. Des plug-ins pourrons venir s'ajouter afin de facilité et rendre plus confortable la phase de codage. Nous préciserons ultérieurement leur nom et leur nature dans les documents correspondants.

Configuration matérielle

Pour mener le projet à bien, les ressources suivantes sont mises à notre disposition :

- Deux machines Windows XP Pro
- Un serveur Korsakov1 pour l'exécution et tests
- Messagerie électronique
- Espace de stockage réseau

9. Bilan qualité

En fin de projet un document faisant le bilan de l'activité qualité sera rédigé. Ce document regroupera tous les documents de suivi et de contrôle qualité, ainsi que les conclusions sur l'application des méthodes utilisées.

Equipe MRIM
Laboratoire CLIPS-IMAG
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Plan de Tests Système

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	11/04/2007
Version	1.0

Historique des versions

Version	Date	Modifications
1.0	11/04/2007	Début de la rédaction

Sommaire

1. Introduction

But du document

Ce document présente le plan de tests système du projet de réalisation d'une application Web pour un système de recommandation de films grâce au filtrage collaboratif basé sur les communautés (COCoFil3).

Les tests système entrent dans le cadre de l'organisation et de la gestion du développement logiciel. Ils ont pour objectif de fixer de façon détaillée les actions et moyens qui permettront de valider le fonctionnement du logiciel créé.

Portée du document

Ce document s'adresse :

- Au responsable de stage : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

Documentation de référence

Le présent document fait référence au Cahier des Charges du projet et est rédigé en fonction des clauses qualités définies dans le Plan d'Assurance Qualité Logicielle.

Il est directement issu du document de spécifications externes puisqu'il permettra de tester cette phase du cycle de vie logiciel.

2. Utilisation des tests système

Les tests système permettent de tester les fonctions offertes à l'utilisateur et de vérifier la cohérence entre le logiciel et sa définition qui décrit les fonctions accessibles à l'utilisateur.

Les tests système ne pourront se faire que lorsque les tests d'intégration seront validés.

Ces tests sont spécifiés à partir des spécifications externes. Il s'agira de vérifier l'existence de critères graphiques ainsi que d'effectuer des tests sur le déroulement des fonctionnalités.

L'équipe de développement en entier est responsable de l'exécution des tests système. Le dossier de spécifications externes et le plan de tests système serviront de support aux tests.

Les tests système sont validés lorsque les résultats obtenus sont conformes aux résultats attendus.

3. Tests système des interfaces

Objectifs

Ce test permet de vérifier que les modules de base de l'interface utilisateur ont été implantés suivant les contraintes définies dans le dossier de spécifications externes.

Aspect de l'interface principale

Le testeur doit vérifier l'existence et le contenu des modules suivant :

- La zone d'identification / inscription qui comprend les éléments suivants :
  - Une zone d'identification et un lien pour s'inscrire si l'utilisateur n'est pas identifié.
  - Un lien vers la gestion de son profil et un lien de déconnexion si l'utilisateur est identifié
- La zone recherche qui permet de trouver des films suivants un mot-clé.
- La zone « Menu » qui permet d'effectuer les différentes tâches, c'est à dire :
  - Afficher ses recommandations
  - Afficher ses favoris
  - Afficher ses films évalués
  - Accéder à la consultation / gestion de ses favoris
- La zone de travail où sera effectué l'affichage du résultat d'une recherche, le formulaire d'inscription, l'affichage d'une liste de films, l'affichage d'une description de film, la gestion du profil et la gestion du carnet d'adresses.

Aspect de l'interface de gestion de profil

Le testeur doit vérifier que l'utilisateur à la possibilité (une fois identifié) d'accéder à ses données personnelles fournit lors de l'inscription. Il doit vérifier que l'on a la possibilité de modifier certaines de ses données (le nom d'utilisateur ne peut pas être modifier par exemple) et que les modifications sont bien enregistrées.

Aspect de l'interface de gestion du carnet d'adresse

Le testeur doit vérifier l'existence des fonctionnalités suivantes :

- Ajout d'un contact au carnet d'adresse
- Suppression d'un contact du carnet d'adresse
- Création d'un groupe de contact
- Suppression d'un groupe de contact
- Ajout d'un contact dans un groupe
- Suppression d'un contact d'un groupe

4. Tests système des fonctionnalités

Objectifs et rappels des fonctionnalités

L'objectif est de vérifier que toutes les fonctionnalités du système ont été implémentées et qu'elles fonctionnent correctement. La suite décrit toutes les fonctionnalités que le système doit vérifier.

Un utilisateur ***quelconque*** peut rechercher un film.

Un utilisateur ***quelconque*** peut s'inscrire. L'inscription se passe en deux étapes :

1. Fournir ses données personnelles :
    - nom d'utilisateur
    - mot de passe
    - adresse
    - profession,
    - âge

Tous ses champs sont obligatoires, ils permettent de calculer les premières recommandations pour l'utilisateur.

2. Définir les paramètres par défaut concernant :
    - L'anonymat
    - Le carnet d'adresses
    - La confidentialité

Cette étape de l'inscription est optionnelle. Elle peut être sautée par l'utilisateur ; dans ce cas les paramètres par défaut du système sont pris en compte.

Tout utilisateur membre connecté au système peut :

- s'identifier en fournissant son nom d'utilisateur et son mot de passe.

Une fois identifié, il accède à l'ensemble des fonctionnalités du système jusqu'à ce qu'il se déconnecte.

Un utilisateur ***quelconque*** peut consulter une liste de films suite à une recherche.
Si l'utilisateur est ***membre et connecté***, il peut en plus :

- Consulter ses recommandations
- Consulter ses évaluations
- Consulter sa liste des favoris

Un utilisateur ***quelconque*** peut agir sur un film suite à une recherche en consultant sa fiche.
Si l'utilisateur est ***membre et connecté***, il peut en plus :

- Evaluer un film
- Commenter un film
- Recommander un film à un contact
- Ajouter un film à sa liste des favoris
- Consulter la fiche d'un film
-

Tout utilisateur ***membre connecté*** au système peut :

- Visualiser son profil selon les critères « contenu » et « qualité » sur une carte 2D. Il peut ensuite agir sur la carte en :
- Visualisant le profil d'un autre utilisateur.

Il peut prendre connaissance des informations personnelles de cet utilisateur (à condition qu'il ne soit pas anonyme), visualiser la liste des films qu'il a évalués, la liste des films qui lui ont été recommandés mais aussi l'ajouter à son carnet d'adresses s'il le désire.

- Modifier ses données personnelles.

Tout utilisateur ***membre connecté*** au système peut :

- Ajouter un contact dans son carnet d'adresses (ce contact ne doit pas déjà exister dans le carnet d'adresses).
- Supprimer un contact du carnet d'adresses (ce contact doit exister).
- Bloquer un contact (ce contact doit exister).
- Créer un groupe de contacts (ce groupe de contacts ne doit pas déjà exister dans le carnet d'adresses).
- Supprimer un groupe de contact (ce groupe de contact doit exister)
- Ajouter un contact dans un groupe de contacts
- Supprimer un contact d'un groupe de contacts

Scénarios nominaux

Les tests suivants permettent de vérifier que les fonctionnalités générales de l'application telles que l'évaluation d'un film, la consultation des recommandations, la recherche d'un film, etc., respectent les spécifications du dossier de spécifications externes.

Le test suivant se présente comme sous la forme d'un scénario expliquant les étapes nécessaires pour réaliser le cas d'utilisation principal demandé par le client.

Scénario : Rechercher un film

Acteur : Bob, utilisateur quelconque.
Contexte : il utilise un navigateur web.

Bob arrive sur le site COCoFil3. Il lance une recherche par mots-clés. Il obtient une liste de films correspondants à la requête. Il trouve le film désiré parmi les différents résultats et consulte sa fiche.

Scénario : S'inscrire

Acteur : Bob, utilisateur internaute.
Contexte : il utilise un navigateur web.

Bob arrive sur COCoFil3 pour la première. Il souhaite s'inscrire afin d'accéder à toutes les fonctionnalités du système.
Il commence par fournir ses données personnelles (nom d'utilisateur, mot de passe, adresse, profession, âge) ; tous ses champs sont obligatoires.
Ensuite, on lui propose de configurer le système, mais il décide de sauter cette étape. Le système prend donc en compte les paramètres par défaut. Bob peut désormais accéder à toutes les fonctionnalités.

Scénario : Consulter ses recommandations

Acteur : Bob, utilisateur membre.
Contexte : Il utilise un navigateur web.

Bob est connecté sur le site COCoFil3 en tant que membre. Il consulte les recommandations qui lui ont été faites depuis sa dernière utilisation du système (nouvelles recommandations). Parmi ses recommandations, il se trouve un film que Bob a déjà vu, tout d'abord il note le film, ensuite il décide d'y mettre un commentaire pour donner ses impressions, le

recommande à son ami Baba présent dans son carnet d'adresses, et enfin il l'ajoute à sa liste des favoris. Bob quitte le système en se déconnectant.

Scénario : Consulter ses évaluations

Acteur : Bob, utilisateur membre.
Contexte : Il utilise un navigateur web.

Bob connecté en tant que membre en consulte ses évaluations. Il voit un film qu'il avait assez bien noté mais décide de le supprimer de ses évaluations car il a changé d'impression sur le film après l'avoir revu la veille.

Scénario : Visualiser les communautés

Acteur : Bob, utilisateur membre.
Contexte : Il utilise un navigateur web.

Bob étant identifié auprès du système, choisit de visualiser son profil selon le profil « contenu », on lui affiche une carte 2D sur laquelle sa position lui est clairement indiquée. Ensuite, il choisit de visualiser son profil selon le critère « qualité », là aussi, on lui affiche une carte 2D sur laquelle sa position lui est indiquée. Il décide alors d'agir sur la carte en visualisant le profil d'un autre utilisateur. Il consulte les informations de cet utilisateur avant de se déconnecter.

Scénario : Gérer son profil

Acteur : Bob, utilisateur internaute.
Contexte : Il utilise un navigateur web.

Bob est connecté sur le site en tant que membre. Il souhaite modifier son adresse, il va dans la gestion du profil et effectue les modifications nécessaires. Ensuite il va consulter son vecteur de positionnement, se renseigne sur ses différentes communautés avant de quitter le site.

Scénario : Agir sur un film

Acteur : Bob, utilisateur internaute.
Contexte : Il utilise un navigateur web.

Bob s'identifie et Il consulte sa liste de favoris. Dans cette liste, figure deux films qu'il a vu il y a peu et qu'il n'avait pas vu quand il les avait ajouter aux favoris. Il note donc le premier film et le recommande à un contact. Il note ensuite le deuxième film et, comme il ne l'a pas aimé, il le supprime de ses favoris.

Scénario : Gérer son carnet d'adresses

Acteur : Bob, utilisateur membre.
Contexte : Il utilise un navigateur web.

Bob entrain de consulter ses recommandations, aperçoit un film qu'un contact lui avait recommandé mais qu'il n'avait pas aimé. Il décide donc de supprimer ce contact. Après avoir ouvert son carnet d'adresse, il sélectionne le contact et le supprime. Le contact est supprimé

seulement après confirmation de Bob. Avant de se déconnecter, Bob se souvient de son ami Rémi il désire l'ajouter dans son carnet d'adresses ; il crée un groupe nommé « amis » puis ajoute Rémi dans ce groupe.

Scénarios d'exceptions

Scénario : Formulaire mal rempli

Acteur : Bob, utilisateur internaute.
Contexte : il utilise un navigateur web.

Bob arrive sur COCoFil3 pour la première. Il souhaite s'inscrire afin d'accéder à toutes les fonctionnalités du système. Pour commencer, il fournit ses données personnelles et les soumet au système. Il se trouve que Bob a oublié de fournir une des données obligatoires. Il est avertit par un message d'erreur avertit Bob de l'oubli. Il fournit la donnée manquante et valide le formulaire.

Scénario : Erreurs à l'identification

Acteur : Bob, utilisateur internaute.
Contexte : Il utilise un navigateur web.

Bob arrive sur le site COCoFil3, entre son nom d'utilisateur mais oublie d'entrer son mot de passe, un message d'erreur indique à Bob que le champ « Mot de passe » est obligatoire. Il entre donc son mot de passe mais ce dernier est erroné, un message le lui indique. Enfin, Bob entre un mot de passe correct et accède à toutes les fonctionnalités accessibles au membre.

Scénario : Recommandation d'un film non évalué

Acteur : Bob, utilisateur membre connecté.
Contexte : Il utilise un navigateur web.

Bob étant identifié auprès du système, choisit de consulter ses nouvelles recommandations, il consulte la fiche d'un film ; ce film pourrait bien plaire à son ami Baba. Il décide de le lui recommander. Pour cela, il note d'abord le film puis le recommande à Baba mais Baba a déjà noté le film, Bob ne peut donc pas le lui recommander.
Comme il n'a pas pu le recommander à Baba, Bob décide de le recommander à Jane elle aussi présente dans son carnet d'adresses. Mais malheureusement, ce film a déjà été recommandé à Jane. De nouveau Bob est avertit par un message d'erreur et le film n'est pas recommandé à Jane non plus. Bob abandonne l'idée de recommander le film à un contact et quitte le système en se déconnectant.

Scénario : Contact déjà existant

Acteur : Bob, utilisateur membre connecté.
Contexte : Il utilise un navigateur web.

Bob étant identifié auprès du système, choisit de visualiser son profil selon le profil « contenu », on lui affiche une carte 2D sur laquelle sa position lui est clairement indiquée. Il décide alors d'agir sur la carte en visualisant le profil d'un autre utilisateur. Il consulte les informations de cet utilisateur qu'il intéressant et l'ajoute dans son carnet d'adresses. Mais ce

profil est déjà dans son carnet d'adresses. Bob est avertit par un message d'erreur, l'utilisateur n'est pas ajouté de nouveau aux contacts de Bob.

5. Exécution des tests

Stratégie de tests et Production des jeux de données

Une fiche dressant la liste des fonctionnalités (avec la priorité de la fonctionnalité et un champ qui indique si elle a été validée ou pas) définies dans le cahier des charges sera établie.
Ensuite, scénario par scénario, on procède à la validation des fonctionnalités.
Pour chaque scénario, les jeux de données requis seront disponibles. Ils seront crées manuellement par l'équipe de développement.
Le déroulement sera le même pour chacun des scénarios :

- Lancer le navigateur
- Suivre le scénario
- Analyser les résultats.

Analyse des résultats

Les résultats sont examinés pour voir s'ils sont conformes aux attentes. Une fiche est créée pour recenser les erreurs, leurs origines, et une estimation du temps qu'il faut pour réaliser les modifications. Ses modifications seront réalisées si le temps requis pour n'est pas très coûteux. Dans le cas échéant, un plan de résolution sera rédigé pour la future équipe de développement.

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Plan de tests unitaires

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	08/08/2007
Version	1.2

Historique des versions

Version	Date	Modifications
1.0	21/05/2007	Début de la rédaction
1.1	13/06/2007	Ajout des tests pour le deuxième incrément
1.2	08/08/2007	Ajout des tests pour le troisième incrément

Sommaire

1. Introduction

Ce document présente le plan de tests unitaires du projet de réalisation d'une application Web pour un système de recommandation de films grâce au filtrage collaboratif basé sur les communautés (COCoFil3).
Ce document est destiné :

- à notre client : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

2. Le module GIU

2.1 Classe : UserBean.java

Cette classe a les attributs suivants :

- userName : nom d'utilisateur
- userAge: âge
- userPassword: mot de passe
- userPasswordBis: confirmation du mot de passe
- userProfession : profession
- userTown : ville
- userZipCode : code postal
- userRegion : region
- isConnected : vrai si l'utilisateur est connecté, faux sinon

Méthode	Fonctionnalité	Jeux de test	Résultat attendu
connectUser()	Gère la connexion d'un utilisateur.	Entrez un nom d'utilisateur et un mot de passe vides c'est-à-dire : `userName` = "" et `userPassword` =""	Erreur : nom d'utilisateur vide.
		Entrez un nom d'utilisateur incorrect	Erreur : nom d'utilisateur incorrect
		Entrez un mot de passe incorrect	Erreur : mot de passe incorrect
		Entrez un nom d'utilisateur et un mot de passe corrects	Ok : l'attribut `isConnected` est à vrai utilisateur connecté
disconnectUser()	Gère la déconnexion d'un utilisateur	Appel simple de la méthode	Toutes les informations de l'utilisateur sont perdues
Validate()	Vérifie le formulaire	Entrez un formulaire mal rempli par exemple : `userRegion`= "", `userPassword` et `userPasswordBis` différents,	Erreur : Formulaire mal rempli avec le message d'erreur associé

	d'inscription de l'utilisateur et l'enregistre	…	
		Entrez un formulaire bien rempli	Ok : inscription bien passée, les données personnelles sont disponibles

2.2 Classe : ProfilBean.java

Cette classe contient les attributs suivants :

- userAge : le nouvel âge
- userOldPassword : le mot de passe actuel
- userPassword : le nouveau mot de passe
- userPasswordBis: la confirmation du nouveau mot de passe
- userProfession: la nouvelle profession de l'utilisateur
- userTown: la nouvelle ville de l'utilisateur
- userRegion : la nouvelle région de l'utilisateur
- userZipCode : le nouveau code postal
- vecteur : le vecteur de communautés de l'utilisateur

Méthode	Fonctionnalité	Jeux de test	Résultat attendu
modifyUser()	Gère les modifications des données personnelles de l'utilisateur	L'attribut `isConnected` de la classe UserBean.java doit être à vrai. Attribuer une valeur aux attributs suivants `userAge` et `userProfession`	Erreur : le formulaire est incomplet
		L'attribut `isConnected` de la classe UserBean.java doit être à vrai. Attribuer une valeur aux attributs suivants `userAge`, `userOldPassword`, `userPassword`, `userPasswordBis`, `userProfession`, `userTown`, `userRegion` et `userRegion`	Ok les modifications sont prises en compte
getV_positionnement()	Gère les informations sur les 5 communautés de l'utilisateur	L'attribut `isConnected` de la classe UserBean.java doit être à vrai	L'attribut `vecteur` contient les informations sur les 5 communautés

2.3 Classe : AddressBook.java

Méthode	Fonctionnalité	Jeux de test	Résultat attendu
addContact()	Ajoute un utilisateur au carnet d'adresses	Un nom d'utilisateur n'appartenant pas encore au carnet d'adresses de l'utilisateur	Un tuple est ajouté à la table contacts contenant le bon utilisateur et étant non bloqué
		Un nom d'utilisateur appartenant déjà au carnet d'adresses de l'utilisateur et se trouvant dans un groupe et étant non bloqué	Un tuple est ajouté à la table contacts contenant le bon utilisateur et étant non bloqué
		Un nom d'utilisateur appartenant déjà au carnet d'adresses de l'utilisateur et se trouvant dans un groupe et étant bloqué	Un tuple est ajouté à la table contacts contenant le bon utilisateur et étant bloqué
addContactFromId()	Ajoute un utilisateur au carnet d'adresses	Un identifiant d'utilisateur n'appartenant pas encore au carnet d'adresses de l'utilisateur	Un tuple est ajouté à la table contacts contenant le bon utilisateur et étant non bloqué
		Un identifiant d'utilisateur appartenant déjà au carnet d'adresses de l'utilisateur et se trouvant dans un groupe et étant non bloqué	Un tuple est ajouté à la table contacts contenant le bon utilisateur et étant non bloqué
		Un nom d'utilisateur appartenant déjà au carnet d'adresses de l'utilisateur et se trouvant dans un groupe et étant bloqué	Un tuple est ajouté à la table contacts contenant le bon utilisateur et étant bloqué
addGroup()	Ajoute un groupe au carnet d'adresses	Un nom de groupe déjà présent dans le carnet d'adresses de l'utilisateur	Il ne se passe rien
		Un nom de groupe absent du carnet d'adresse de l'utilisateur	Le tuple correspondant est ajouté à la table groups.
bloquerContact()	Bloque le contact	Un nom d'utilisateur appartenant au carnet d'adresses et étant débloqué.	Le tuple correspondant est modifié
debloquerContact()	Débloque le contact	Un nom d'utilisateur appartenant au carnet d'adresses et étant bloqué.	Le tuple correspondant est modifié
deleteContact()	Supprime un utilisateur du carnet d'adresses	Un nom d'utilisateur n'appartenant pas au carnet d'adresses de l'utilisateur	Il ne se passe rien
		Un nom d'utilisateur appartenant au carnet d'adresse de l'utilisateur et ne se trouvant pas dans un groupe	Le tuple correspondant est supprimé de la table contacts
		Un nom d'utilisateur appartenant au carnet d'adresses de l'utilisateur et se trouvant dans un groupe	Les tuples correspondants sont supprimés de la table contacts
	Supprime un utilisateur	Un nom d'utilisateur	Il ne se passe rien

deleteContactFromGroup()	d'un groupe du carnet d'adresses	n'appartenant pas au carnet d'adresse de l'utilisateur	
		Un nom de groupe et un nom d'utilisateur appartenant au carnet d'adresses de l'utilisateur et ne se trouvant pas dans ce groupe	Il ne se passe rien
		Un nom de groupe et un nom d'utilisateur appartenant au carnet d'adresses de l'utilisateur et se trouvant dans ce groupe	Le tuple correspondant est supprimé de la table groupe et est mis à jour dans la table contacts
deleteGroup()	Supprime un groupe du carnet d'adresses	Un nom de groupe n'appartenant pas au carnet d'adresses de l'utilisateur	Il ne se passe rien
		Un nom de groupe appartenant au carnet d'adresse de l'utilisateur et ne contenant aucun contact	Le tuple correspondant est supprimé de la table groups
		Un nom de groupe appartenant au carnet d'adresse de l'utilisateur et contenant des contacts	Le tuple correspondant est supprimé de la table groupe et la table contacts est mise à jour pour modifier le groupe des utilisateurs qui se trouvaient dans le groupe supprimé.

3. Le module GRE

Le module GRE gère les recommandations et les évaluations de l'utilisateur. Les calculs de recommandation sont faits sur cinq critères effectués respectivement sur 5 méthodes java. De nombreux points doivent être testés comme ne pas recommander à l'utilisateur un film qu'il a déjà dans sa liste des recommandations ou des évaluations.
Afin de tester les cinq algorithmes de recommandation, cinq classe Junit ont été crées (cf. annexes).

4. Le module GF

4.1Classe : Movie.java

Méthode	Fonctionnalité	Jeux de test	Résultat attendu
getMovieActors()	Renvoie les acteurs d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les acteurs correspondant au film d'identifiant `movieId`
		Une valeur de `movieId` dont aucun film n'a pour identifiant	`movieActors` est null.

getMovieColors()	Renvoie les couleurs d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les couleurs correspondant au film d'identifiant `movieId`
		Une valeur de `movieId` dont aucun film n'a pour identifiant	`movieActors` est null.
getMovieDirectors()	Renvoie les réalisateurs d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les directeurs correspondant au film d'identifiant `movieId`
		Une valeur de `movieId` dont aucun film n'a pour identifiant	`movieActors` est null.
getMovieGenres()	Renvoie les genres d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les genres correspondant au film d'identifiant `movieId`
		Une valeur de `movieId` dont aucun film n'a pour identifiant	`movieActors` est null.
getMovieKeyWords()	Renvoie les mots clès d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les mots clès correspondant au film d'identifiant `movieId`
		Une valeur de `movieId` dont aucun film n'a pour identifiant	`movieActors` est null.
getMovieProducers()	Renvoie les prducteurs d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les producteurs correspondant au film d'identifiant `movieId`
		Une valeur de `movieId` dont aucun film n'a pour identifiant	`movieActors` est null.
getMovieProc ducersCompany()	Renvoie les compagnies de production d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les compagnies de production correspondant au film d'identifiant `movieId`
		Une valeur de `movieId` dont aucun film n'a pour identifiant	`movieActors` est null.
getMovieWriters()	Renvoie les scénaristes d'un film dont l'identifiant est celui de l'attribut `movieId`	Une valeur de `movieId` correspondant à un film de la base	`movieActors` contient les scénaristes correspondant au film d'identifiant `movieId`
		Une valeur de `movieId`	`movieActors`

		dont aucun film n'a pour identifiant	est null.
getUserRate()	Renvoie la note de l'utilisateur d'identifiant `user.userId` du film dont l'identifiant est celui de l'attribut `movieId`	`movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui a vue ce film	`movieActors` contient les scénaristes correspondant au film d'identifiant `movieId`
		`movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui n'a pas vue ce film	`userRate` vaut 0.
		Une valeur de `movieId` dont aucun film n'a pour identifiant et `user.userId` quelconque	`userRate` vaut 0.
movieToEvaluation()	Ajoute le film dont l'identifiant est celui de l'attribut `movieId` aux évaluations de l'utilisateur d'identifiant `user.userId`	Une valeur de `movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui n'a pas encore évaluer ce film	Le tuple correspondant est ajouté à la table rating et les recommandations de `movieId` pour `user.userId` sont supprimées
		Une valeur de `movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui a déjà évalué ce film et `userRate` différent de 0	Le tuple correspondant est mis à jour dans la table rating
		Une valeur de `movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui a déjà évalué ce film et `userRate` égal à 0	Le tuple correspondant est supprimé de la table rating
getMovieId	Renvoie l'identifiant du film dont le titre est celui de l'attribut `title`	Une valeur de `title` correspondant à un film de la base	`movieId` contient les scénaristes correspondant au film de titre `title`
		Une valeur de `title` dont aucun film n'a pour titre	`movieId` vaut 0.

4.2 Classe : Search.java

Méthode	Fonctionnalité	Jeux de test	Résultat attendu
seek()	Recherche tous les films contenant la chaîne `searchQuery`.	Une valeur de `searchQuery` contenu dans l'un des titres des films de la base	`searchResult` est un tableau remplie avec les films correspondant à la recherche
		Une valeur de `searchQuery` contenu dans aucun des titres des films de la base	`searchResult` est un tableau Le tableau `searchQuery`

			est vide
getNbResults()	Retourne le nombre de film dans le tableau `searchQuery`	Le tableau `searchQuery` est vide	0
		Le tableau `searchQuery` n'est pas vide	le nombre de film dans `searchQuery`
movieToEvaluation()	Ajoute le film dont l'identifiant est celui de l'attribut `movieId` aux évaluations de l'utilisateur d'identifiant `user.userId`	Une valeur de `movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui n'a pas encore évaluer ce film	Le tuple correspondant est ajouté à la table rating et les recommandations de `movieId` pour `user.userId` sont supprimées
		Une valeur de `movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui a déjà évalué ce film et `userRate` différent de 0	Le tuple correspondant est mis à jour dans la table rating
		Une valeur de `movieId` correspondant à un film de la base et `user.userId` à un utilisateur qui a déjà évalué ce film et `userRate` égal à 0	Le tuple correspondant est supprimé de la table rating

5. Le module CVC

5.1Classe : UserProfile.java

Méthode	Fonctionnalité	Jeux de test	Résultat attendu
getProfilContent()	Renvoie l'appréciation par genre de film pour les utilisateurs dont les identifiants sont ceux des attributs `uid1` et `uid2`	Une valeur de `uid1` ou de `uid2` ne correspondant pas à un identifiant d'utilisateur	`profileContent` contient le profil suivant le critère contenu pour les utilisateurs d'identifiant `uid1` et `uid2` avec des valeurs nulles pour l'identifiant ne correspondant pas à un utilisateur
		Une valeur de `uid1` et de `uid2` correspondant à un identifiant d'utilisateur	`profileContent` contient le profil suivant le critère contenu pour les utilisateurs d'identifiant `uid1` et `uid2`
getProfilQuality()	Renvoie les évaluations communes pour les utilisateurs dont les identifiants sont ceux des attributs `uid1` et `uid2`	Une valeur de `uid1` ou de `uid2` ne correspondant pas à un identifiant d'utilisateur	`profileContent` a la valeur null
		Une valeur de `uid1` et de `uid2` correspondant à un identifiant d'utilisateur mais n'ayant pas d'évaluation commune	`profileContent` a la valeur null

		Une valeur de `uid1` et de `uid2` correspondant à un identifiant d'utilisateur et ayant des évaluations communes	`profileContent` contient ces évaluations

5.2 Classe : CommunityProfile.java

Méthode	Fonctionnalité	Jeux de test	Résultat attendu
getCommunityStat()	Renvoie les films les plus vus et la moyenne des notes pour les utilisateurs de la communauté dont l'identifiant est l'attribut `cid` de la classe	Une valeur de `cid` ne correspondant pas à un identifiant de communauté	`communityStat` contient la valeur null
		Une valeur de `cid` correspondant à un identifiant de communauté	`communityStat` contient le profil de la communauté dont l'identifiant est `cid`

6. Annexes

6.1 Test_age.java : tests du calcul de recommandation selon le critère âge

```
package Tests;

import COMMUNITY.Lib;
import GRE.RECOMMENDATIONS.getRecommendations_a1Optimise;
import junit.framework.TestCase;

public class test_age extends TestCase {

	public test_age(String name) {
		super(name);
	}

	protected void setUp() throws Exception {
		super.setUp();
	}

	protected void tearDown() throws Exception {
		super.tearDown();
	}

	 public void test1() {
		int id=1;
			Lib lib = new Lib();
			java.util.Date strt, end;
			int     hS, mS, sS, hE, mE, sE;
			strt = new java.util.Date();
			hS   = strt.getHours();
			mS   = strt.getMinutes();
			sS   = strt.getSeconds();
			System.out.println("** Calcul de recommandation selon le critère âge
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
			getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
			end = new java.util.Date();
			hE  = end.getHours();
			mE  = end.getMinutes();
			sE  = end.getSeconds();
			System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
			System.out.println(" ");
		}
```

```
	public void test2() {
		int id=2;
			Lib lib = new Lib();
			java.util.Date strt, end;
			int     hS, mS, sS, hE, mE, sE;
			strt = new java.util.Date();
			hS   = strt.getHours();
			mS   = strt.getMinutes();
			sS   = strt.getSeconds();
			System.out.println("** Calcul de recommandation selon le critère âge
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
			getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
			end = new java.util.Date();
			hE  = end.getHours();
			mE  = end.getMinutes();
			sE  = end.getSeconds();
			System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
			System.out.println(" ");
		}

	public void test3() {
		int id=10;
			Lib lib = new Lib();
			java.util.Date strt, end;
			int     hS, mS, sS, hE, mE, sE;
			strt = new java.util.Date();
			hS   = strt.getHours();
			mS   = strt.getMinutes();
			sS   = strt.getSeconds();
			System.out.println("** Calcul de recommandation selon le critère age
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
			getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
			end = new java.util.Date();
			hE  = end.getHours();
			mE  = end.getMinutes();
			sE  = end.getSeconds();
			System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
			System.out.println(" ");
		}

	public void test4() {
		int id=11;
			Lib lib = new Lib();
			java.util.Date strt, end;
```

```
                        int     hS, mS, sS, hE, mE, sE;
                        strt = new java.util.Date();
                        hS   = strt.getHours();
                        mS   = strt.getMinutes();
                        sS   = strt.getSeconds();
                        System.out.println("** Calcul de recommandation selon le critère age
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                        getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
                        end = new java.util.Date();
                        hE  = end.getHours();
                        mE  = end.getMinutes();
                        sE  = end.getSeconds();
                        System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
                        System.out.println(" ");
                }

                public void test5() {
                        int id=12;
                                Lib lib = new Lib();
                                java.util.Date strt, end;
                                int     hS, mS, sS, hE, mE, sE;
                                strt = new java.util.Date();
                                hS   = strt.getHours();
                                mS   = strt.getMinutes();
                                sS   = strt.getSeconds();
                                System.out.println("** Calcul de recommandation selon le critére age
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                                getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
                                end = new java.util.Date();
                                hE  = end.getHours();
                                mE  = end.getMinutes();
                                sE  = end.getSeconds();
                                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
                                System.out.println(" ");
                 }

                public void test6() {
                        int id=20;
                                Lib lib = new Lib();
                                java.util.Date strt, end;
                                int     hS, mS, sS, hE, mE, sE;
                                strt = new java.util.Date();
                                hS   = strt.getHours();
                                mS   = strt.getMinutes();
```

```
                        sS   = strt.getSeconds();
                        System.out.println("** Calcul de recommandation selon le critère age
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                        getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
                        end = new java.util.Date();
                        hE  = end.getHours();
                        mE  = end.getMinutes();
                        sE  = end.getSeconds();
                        System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
                        System.out.println(" ");
                }

                public void test7() {
                        int id=997;
                        Lib lib = new Lib();
                        java.util.Date strt, end;
                        int     hS, mS, sS, hE, mE, sE;
                        strt = new java.util.Date();
                        hS   = strt.getHours();
                        mS   = strt.getMinutes();
                        sS   = strt.getSeconds();
                        System.out.println("** Calcul de recommandation selon le critère âge
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                        getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
                        end = new java.util.Date();
                        hE  = end.getHours();
                        mE  = end.getMinutes();
                        sE  = end.getSeconds();
                        System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
                        System.out.println(" ");
                }

                public void test8() {
                        int id=1963;
                        Lib lib = new Lib();
                        java.util.Date strt, end;
                        int     hS, mS, sS, hE, mE, sE;
                        strt = new java.util.Date();
                        hS   = strt.getHours();
                        mS   = strt.getMinutes();
                        sS   = strt.getSeconds();
                        System.out.println("** Calcul de recommandation selon le critère âge
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
```

```
                        getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
                        end = new java.util.Date();
                        hE  = end.getHours();
                        mE  = end.getMinutes();
                        sE  = end.getSeconds();
                        System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
                        System.out.println(" ");
                }

                public void test9() {
                    int id=2698;
                        Lib lib = new Lib();
                        java.util.Date strt, end;
                        int     hS, mS, sS, hE, mE, sE;
                        strt = new java.util.Date();
                        hS   = strt.getHours();
                        mS   = strt.getMinutes();
                        sS   = strt.getSeconds();
                        System.out.println("** Calcul de recommandation selon le critère âge
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                        getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
                        end = new java.util.Date();
                        hE  = end.getHours();
                        mE  = end.getMinutes();
                        sE  = end.getSeconds();
                        System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
                        System.out.println(" ");
                }

                public void test10() {
                    int id=3005;
                        Lib lib = new Lib();
                        java.util.Date strt, end;
                        int     hS, mS, sS, hE, mE, sE;
                        strt = new java.util.Date();
                        hS   = strt.getHours();
                        mS   = strt.getMinutes();
                        sS   = strt.getSeconds();
                        System.out.println("** Calcul de recommandation selon le critère âge
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                        getRecommendations_a1Optimise a1 = new
getRecommendations_a1Optimise(id);
                        end = new java.util.Date();
                        hE  = end.getHours();
```

```
                        mE  = end.getMinutes();
                        sE  = end.getSeconds();
                        System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS,
sS, hE, mE, sE));
                        System.out.println(" ");
                }

        }
```

6.2 Test_profession.java : tests du calcul de recommandation selon le critère profession

```
package Tests;

import COMMUNITY.Lib;
import GRE.RECOMMENDATIONS.getRecommendations_a2Optimise;
import junit.framework.TestCase;

public class test_profession extends TestCase {

        public test_profession(String name) {
                super(name);
        }

        protected void setUp() throws Exception {
                super.setUp();
        }

        protected void tearDown() throws Exception {
                super.tearDown();
        }

        public void test1() {
        int id=1;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
```

```
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test2() {
        int id=2;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
     }

    public void test3() {
        int id=10;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
```

```
    }

    public void test4() {
        int id=11;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test5() {
        int id=12;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
     }

    public void test6() {
        int id=20;
```

```
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test7() {
            int id=997;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test8() {
            int id=1963;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
```

```
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test9() {
            int id=2698;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test10() {
            int id=3005;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
```

```
                System.out.println("** Calcul de recommandation selon le critère profession
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a2Optimise a2 = new
getRecommendations_a2Optimise(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }


}
```

6.3 Test_localisation.java : tests du calcul de recommandation selon le critère localisation géographique

```
package Tests;

import COMMUNITY.Lib;
import GRE.RECOMMENDATIONS.getRecommendations_a3;
import junit.framework.TestCase;

public class test_localisation extends TestCase {

        public test_localisation(String name) {
                super(name);
        }

        protected void setUp() throws Exception {
                super.setUp();
        }

        protected void tearDown() throws Exception {
                super.tearDown();
        }

        public void test1() {
        int id=1;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
```

```
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test2() {
        int id=2;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
     }

    public void test3() {
        int id=10;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
```

```
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test4() {
        int id=11;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test5() {
        int id=12;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }
```

```
    public void test6() {
        int id=20;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
      }

    public void test7() {
        int id=997;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
      }

    public void test8() {
        int id=1963;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
```

```
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test9() {
            int id=2698;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test10() {
            int id=3005;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère localisation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                getRecommendations_a3 a3 = new getRecommendations_a3(id);
```

```
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }


}
```

6.4 Test_genre.java : tests du calcul de recommandation selon le critère genre

```
package Tests;

import junit.framework.TestCase;
import COMMUNITY.Lib;
import GRE.RECOMMENDATIONS_GENRE.dataPreparation;
import GRE.RECOMMENDATIONS_GENRE.getRecommendations_a5;


public class test_genre extends TestCase {


    private java.util.List emptyList;


    /**
     * Sets up the test fixture.
     * (Called before every test case method.)
     */
    protected void setUp() {
        emptyList = new java.util.ArrayList();
    }


    /**
     * Tears down the test fixture.
     * (Called after every test case method.)
     */
    protected void tearDown() {
        emptyList = null;
    }


    public void test1() {
```

```
        int id=509;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test2() {
        int id=555;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
```

```
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test3() {
        int id=1014;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test4() {
        int id=2205;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
```

```
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test5() {
        int id=3006;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
     }

    public void test6() {
        int id=4016;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
```

```
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test7() {
        int id=5020;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test8() {
        int id=5021;
                Lib lib = new Lib();
```

```
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test9() {
        int id=5022;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
```

```
                System.out.println(" ");
    }

    public void test10() {
        int id=5023;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère genre pour
l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                dataPreparation d = new dataPreparation();
                getRecommendations_a5 r = new getRecommendations_a5();
                d.upadateStats(id);
                d.createCNT_Profiles_And_MatrixPF(id, 19, 5, 5, 5, "identity", "genre",
"stats", "a", "b");
                r.getCommunityForId(id);
                r.getRecommendationsForId(id);
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

}
```

6.5 Test_evaluation.java : tests du calcul de recommandation selon le critère évaluation

```
package Tests;
import COMMUNITY.Lib;
import GRE.RECOMMENDATIONS_EVALUATION.getRecommendations_a6;
import junit.framework.TestCase;


public class test_evaluation extends TestCase {


    private java.util.List emptyList;


    /**
     * Sets up the test fixture.
     * (Called before every test case method.)
     */
    protected void setUp() {
        emptyList = new java.util.ArrayList();
    }


    /**
     * Tears down the test fixture.
     * (Called after every test case method.)
     */
    protected void tearDown() {
        emptyList = null;
    }


    public void test1() {
          int id=1;
                    Lib lib = new Lib();
                    java.util.Date strt, end;
                    int     hS, mS, sS, hE, mE, sE;
                    strt = new java.util.Date();
                    hS   = strt.getHours();
                    mS   = strt.getMinutes();
                    sS   = strt.getSeconds();
                    System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                    double  delta      = 0.10;
                    double  minSupport = 0.40;
                    int  top_N         = 20;
                    getRecommendations_a6 a6 = new getRecommendations_a6();
```

```
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test2() {
        int id=2;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta     = 0.10;
                double  minSupport = 0.40;
                int  top_N         = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
     }

    public void test3() {
        int id=10;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
```

```
                double  delta     = 0.10;
                double  minSupport = 0.40;
                int  top_N        = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test4() {
        int id=11;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta     = 0.10;
                double  minSupport = 0.40;
                int  top_N        = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
    }

    public void test5() {
        int id=12;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
```

```
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta     = 0.10;
                double  minSupport = 0.40;
                int  top_N         = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test6() {
        int id=20;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta     = 0.10;
                double  minSupport = 0.40;
                int  top_N         = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test7() {
        int id=997;
                Lib lib = new Lib();
```

```
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta      = 0.10;
                double  minSupport = 0.40;
                int  top_N         = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

    public void test8() {
            int id=1963;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta      = 0.10;
                double  minSupport = 0.40;
                int  top_N         = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }
```

```
    public void test9() {
        int id=2698;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta     = 0.10;
                double  minSupport = 0.40;
                int  top_N         = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
      }

    public void test10() {
        int id=3005;
                Lib lib = new Lib();
                java.util.Date strt, end;
                int     hS, mS, sS, hE, mE, sE;
                strt = new java.util.Date();
                hS   = strt.getHours();
                mS   = strt.getMinutes();
                sS   = strt.getSeconds();
                System.out.println("** Calcul de recommandation selon le critère évaluation
pour l'utilisateur n°"+ id+" (" + hS + ":" + mS + ":" + sS + ")");
                double  delta     = 0.10;
                double  minSupport = 0.40;
                int  top_N         = 20;
                getRecommendations_a6 a6 = new getRecommendations_a6();
                a6.qualityCF(id,"198804", "a6", minSupport, delta, top_N, "refuser_partition",
"refuser_partition", "rating", "evaluation");
                end = new java.util.Date();
                hE  = end.getHours();
                mE  = end.getMinutes();
                sE  = end.getSeconds();
```

```
                System.out.println("   ==> Durée du calcul= " + lib.durationInt(hS, mS, sS, hE,
mE, sE));
                System.out.println(" ");
        }

}
```

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Dossier de conception globale

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	13/07/2007
Version	1.1

Historique des versions

Version	Date	Modifications
1.0	23/04/2007	Début de la rédaction
1.1	13/07/2007	Modification suite aux remarques faites lors du 3ème audit.

Sommaire

1. Introduction

But et portée du document

Ce document a pour but de présenter l'architecture et les choix de conception à adopter pour la réalisation du projet COCoFil3. Nous rappelons que le projet est une application Web pour un système de recommandation de films grâce au filtrage collaboratif basé sur les communautés.

Ce document fait suite à la phase de spécifications externes et a pour but d'exposer les choix d'architecture retenus. Les aspects suivants sont traités :

- Vue logique : représentation sous forme de modules et interactions de l'application.
- Vue physique : montre l'allocation des ressources matérielles pour les éléments de la vue logique.
- Vue dynamique : montre l'enchaînement des interactions entre les différents modules.
- Vue de développement : montre l'organisation des différentes sources (code, librairies, package, …).

Le document présente également les choix de technologies mises en jeu, nous spécifions donc les langages et framework de programmation utilisés et enfin la modélisation.

Ce document s'adresse :

- Au responsable de stage : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

Le présent document fait référence au Dossier de Spécifications Externes du projet et est rédigé en fonction des clauses qualités définies dans le Plan d'Assurance Qualité Logicielle.

A propos du projet

Le projet COCoFil3 consiste à développer une application web autour d'un système de recommandations basé sur les films. Au cours de précédents travaux, le noyau d'un système de recommandation orienté vers les communautés, nommé COCoFil2 a été développé par An-Te Nguyen. COCoFil3 vise à rendre ce système interactif et dynamique.

2. Vues architecturales

Contexte

Cette section présente le contexte d'utilisation de l'application à développer, c'est-à-dire les acteurs et les interactions avec le système.

Conformément aux spécifications externes et au cahier des charges, les deux acteurs qui interagissent avec COCoFil3 sont l'Internaute et le Membre.

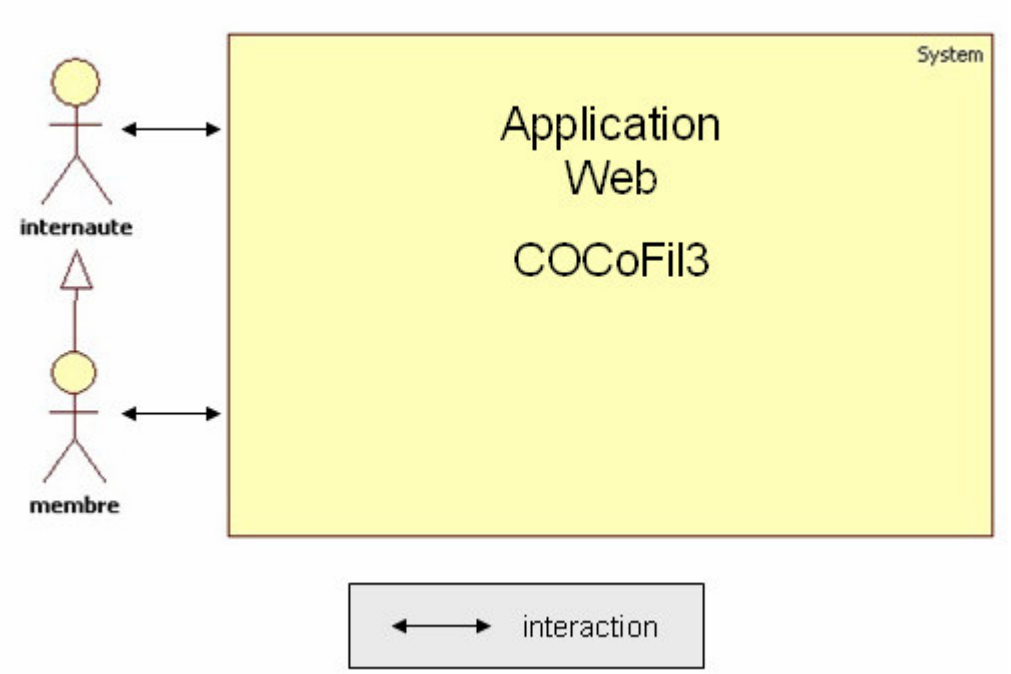


Figure 1: Diagramme de contexte de l'application COCoFil3

L'application Web propose à un Internaute un accès limité aux fonctionnalités du système. Seules les fonctionnalités de recherche, de consultation de détails sur les films et d'inscription lui sont accessibles. L'inscription transforme son statut en membre lui permettant ainsi d'accéder, après s'être identifié, à la totalité des fonctionnalités du système.

Vue logique

Cette vue présente la structuration du système en unités de développement que l'on nomme modules.
L'application COCoFil3 sera basée sur une architecture à trois niveaux (3-tier). L'architecture logique du système sera donc divisée en trois couches (cf. Figure2) :

- Couche présentation : Cette couche est représentée par le module IHM
- Couche métier : Cette couche est représentée par le module LOGIQUE METIER
- Couche accès aux données : Cette couche est représentée par le module BD

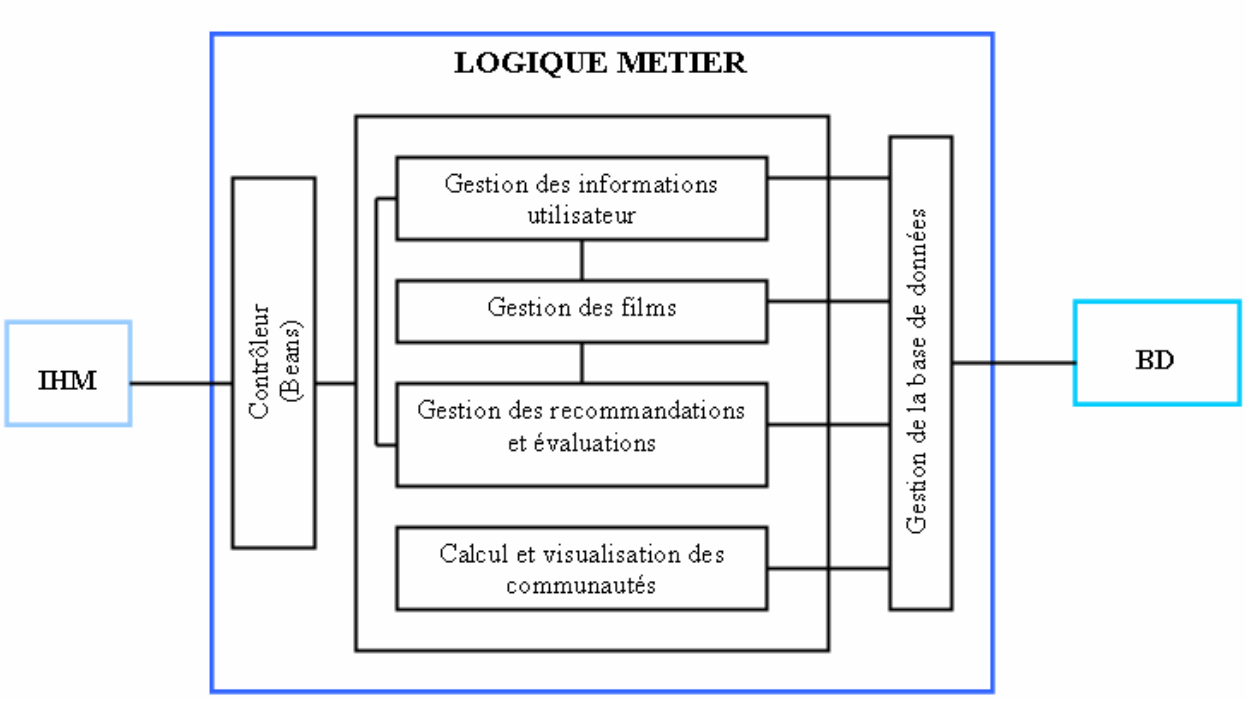


Figure 2: Vue logique de l'application COCoFil3

Description des modules

Le module IHM

Le module IHM correspond à la Vue de notre architecture MVC (Modèle – Vue - Contrôleur). En effet, c'est dans ce module que l'on va trouver les pages JSF de l'application. JSF (voir section 3) facilite l'écriture d'interface à partir d'une bibliothèque de contrôles et gère automatiquement l'état http entre client et serveur (en cas de Postback par exemple). De plus, JSF fournit un modèle simple pour la gestion des évènements côté client et côté serveur et autorise les créations ou l'enrichissement de composants graphiques utilisateur (custom).

Ayant choisi l'utilisabilité comme facteur principal de qualité, la conception de ce module requiert donc une attention particulière.

La conception devra donc prendre en compte différents critères :

- Garder une homogénéité dans l'ensemble des pages en utilisant un modèle de page (template) dans lequel s'insère le contenu.
- Simplifier au maximum l'enchaînement des tâches afin d'obtenir une charge de travail minimum pour l'utilisateur.
- Présenter les sorties de manière claire et concise afin de respecter le critère de communicativité. De même, les entrées demandées devront se restreindre au minimum nécessaire.

Le module LOGIQUE METIER

A. Contrôleur

Ce module « contrôleur » prend en charge le rôle du « controller » dans le modèle MVC (Model – View – Controller) c'est-à-dire la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle. Il n'effectue aucun traitement, ne modifie aucune donnée, il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande.
Il fait le lien entre vue (module « IHM ») et le modèle (module « LOGIQUE METIER »). En d'autres termes, il analyse les requêtes en provenance du module « IHM », demande au modèle approprié d'effectuer les traitements puis actualise la vue.
Compte tenu du framework utilisé (JSF, cf. 3.4), ce module ne contiendra que des java-beans.

B. Gestion de la base de données (GBD)

Le module « Gestion de la base de données » permet de réaliser les accès au module « «BD » représentant la base de données. Il aura aussi pour but de réaliser la connexion et la déconnexion à la base de données mais aussi l'exécution des requêtes envoyées par les modules suivants : « Gestion des informations utilisateur », « Gestion des films », « Gestion des recommandations et des évaluations », « Calcul et visualisation des communautés ».

C. Gestion des informations utilisateur (GIU)

Le module « Gestion des données utilisateur » gère les informations relatives aux utilisateurs. Ce module implémentera les fonctionnalités suivantes:

- L'identification d'un utilisateur
- La déconnexion d'un utilisateur
- L'inscription d'un nouvel utilisateur

- La modification des données personnelles et des paramètres par défaut
- La gestion du carnet d'adresses (ajout de contact, suppression de contact, création de groupes de contacts…)
- La gestion de la liste des favoris (ajout et suppression de films à la liste des favoris)

Il utilise le module « BD » via le module « Gestion de la base données » pour y enregistrer des données (par exemple, des informations utilisateur à l'inscription : nom d'utilisateur, mot de passe….) ou pour y extraire des données (par exemple, la liste des favoris).

D. Gestion des films (GF)

Le module « Gestion des films » implémente les fonctionnalités suivantes :

- Recherche de films par mots clés
- Lecture d'informations sur les films.

Ce module permet donc d'obtenir une liste de films à partir d'un mot-clé et d'obtenir la fiche descriptive d'un film.

Même si la recherche peut être vue comme un élément négligeable du fait que l'application est là pour recommander les films à l'utilisateur, elle reste capitale pour maintenir l'intérêt de l'utilisateur en lui permettant de s'informer sur les films. De plus, la recherche permet à l'utilisateur de trouver rapidement les films afin de les évaluer, notamment après une inscription où le profil de l'utilisateur est pauvre.

E. Gestion des recommandations et évaluations (GRE)

Le rôle de ce module est de gérer toutes les recommandations faites à l'utilisateur et toutes ses évaluations. Il comprendra les unités de calcul des recommandations selon les critères âge, profession, localisation géographique, genre et évaluation.

Il va également utiliser le module « BD » via le module « Gestion de la base de données » pour l'enregistrement de données telles que les recommandations produites après le calcul ou une évaluation faite par l'utilisateur. Il va aussi extraire des données (liste des films recommandés à un utilisateur par exemple).

F. Calcul et visualisation des communautés (CVC)

Ce module contient le calcul des cartes suivant deux critères : le contenu ou la qualité. Une fois que la position de chaque utilisateur a été calculée et que les utilisateurs ont été répartis en communautés, le module permet de créer une carte au format SVG pour permettre à l'utilisateur de visualiser le résultat.

Le module BD

Ce module représente la base de données. On y accède grâce au module « Gestion de la base de données ». Les tables de la base de données seront décrites dans le dossier de conception détaillée.

Vue physique

Cette vue montre comment les éléments de la vue physique sont alloués à des plates-formes d'exécution. On détaille donc :

- Les contraintes des éléments logiciels
- Les caractéristiques des éléments hardware

En phase de codage et de test, le client et le serveur s'exécutent sur la même machine. Le client exécute l'application sur un PC « standard », c'est-à-dire dans ce cas équipé pour faire fonctionner un navigateur Web dans de bonnes conditions.

A terme, le serveur doit s'exécuter sur une machine linux (korsakov1, Pentium 4, 3.2 GHz, Ram 2Go, disque 0.6 To, Fedora Core 4 32 bits). Il est prévu de faire la migration au passage de l'incrément 2 à l'incrément 3, dès lors que des problèmes de performances (suite au passage à la base de données plus importante) risquent d'avoir un impact sur l'utilisation du système.

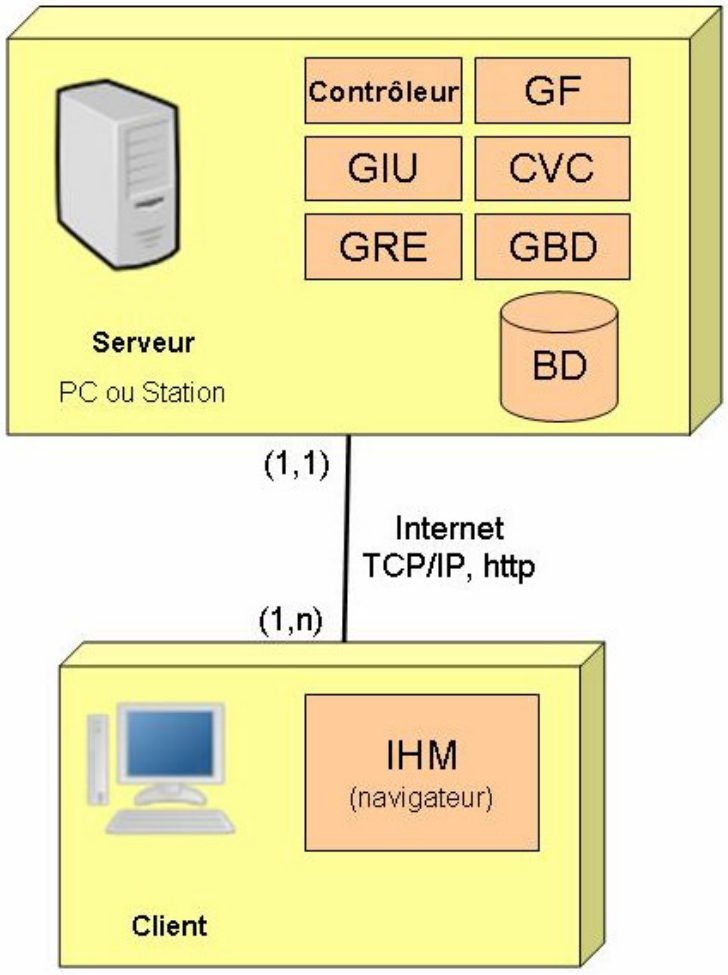


Figure 3: Vue physique de l'application COCoFil3

Vue dynamique

Les diagrammes de séquence suivants représentent la vue dynamique de l'application COCoFil3. Ils schématisent les interactions qui peuvent survenir entre les différents modules. Nous allons représenter la dynamique de l'application par les deux scénarios suivants :

1. Identification de l'utilisateur

Le diagramme de séquence ci-dessous illustre les interactions entre les différents modules de la vue logique (voir section 2.2) lorsqu'un utilisateur souhaite se connecter. Le module « IHM » envoie la demande d'identification vers le contrôleur qui achemine la requête vers le module concerné « GIU ». Le module « GIU » via le module « GBD » interroge la base de données (module « BD ») pour savoir si l'utilisateur existe. La base de données répond et la vue est rafraîchie.

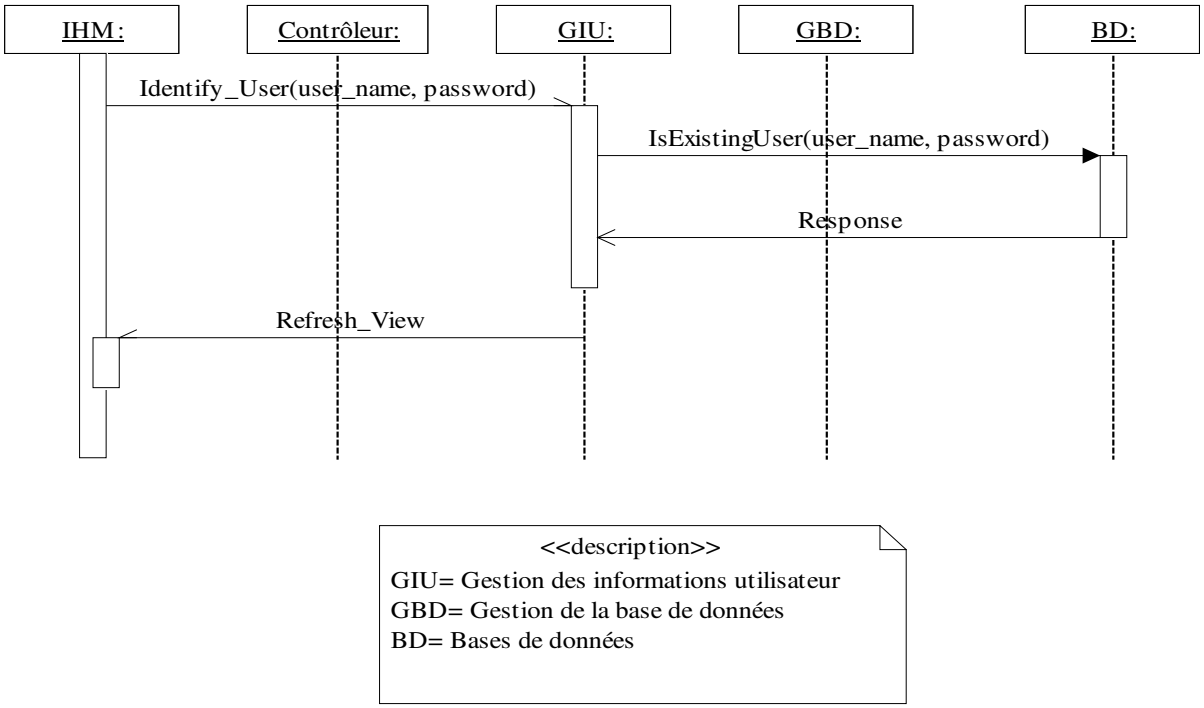


Figure 4: Vue dynamique (pour l'identification)

2. Consultation des recommandations

Le processus de communication entre les modules est identique à celui utilisé lors d'une identification sauf que l'on s'adresse ici au module GRE (gestion des recommandations et des évaluations) au lieu de s'adresser au module GIU.

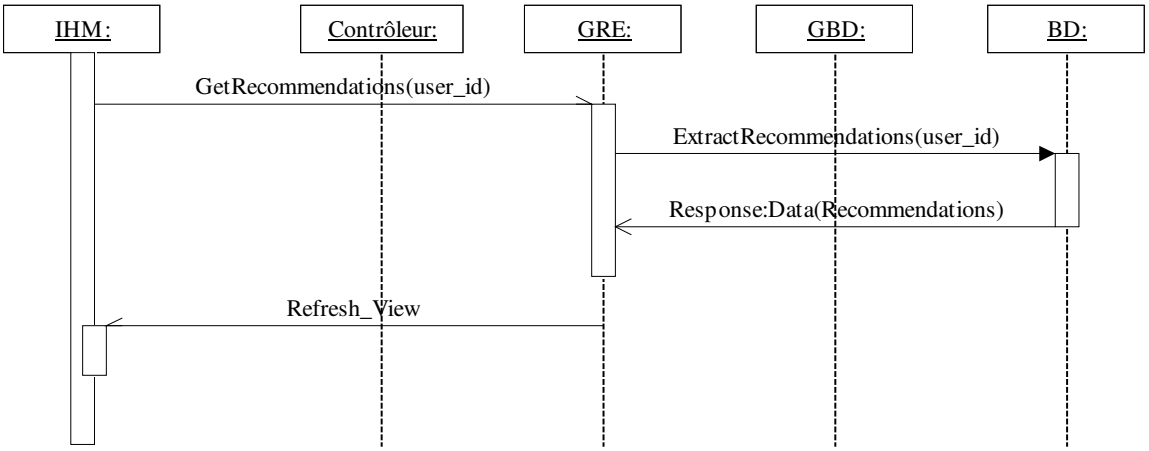


Figure 5: Vue dynamique (pour la consultation de recommandations)

Vue de développement

Cette vue montre la projection des éléments de la vue module sur une infrastructure de développement. Elle présente donc la hiérarchie des répertoires ainsi que leur utilité.

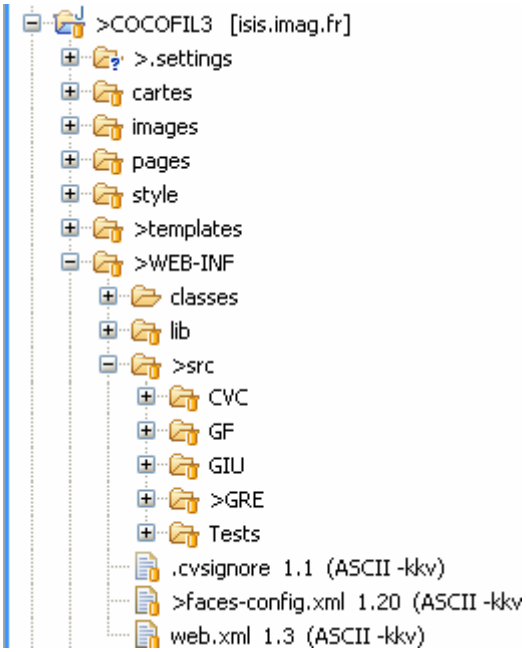


Figure 6: Vue de développement de l'application COCoFil3

Le répertoire « WEB-INF » du dossier « COCoFil3 » (répertoire racine) contient les dossiers :

- « src » : contient les sources de l'application (reparties en plusieurs package).
- « classes » : contient les versions compilées des sources.
- « lib »: les librairies nécessaires à l'application Web (librairies de JSF, AJAX4JSF, …).

Ainsi que les fichiers :

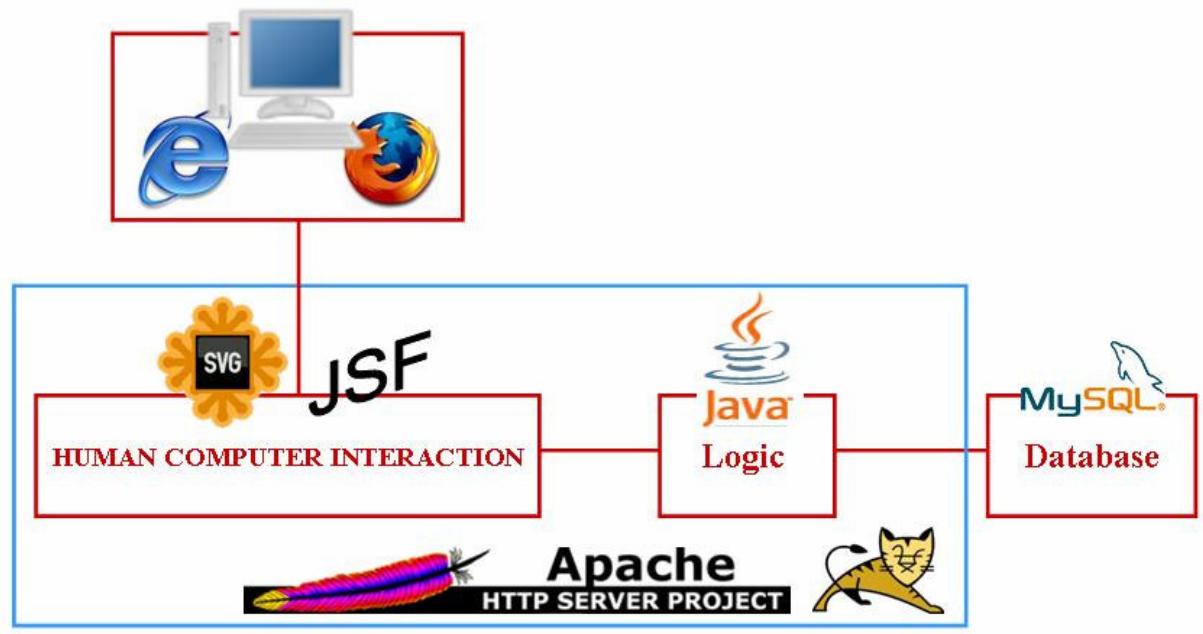
- « web.xml » : descripteur de déploiement de l'application web contenant les caractéristiques et paramètres de l'application (paramètres d'initialisation, de configuration, etc…).
- « faces-config.xml » : paramétrage de JSF permettant de fournir des valeurs d'initialisation pour les ressources nécessaires à l'application, la déclaration des beans managés, les règles de navigation, etc.

Le dossier « cartes » contient les cartes créées par le système, le dossier « pages » contient les pages JSF et le dossier « images » contient les images utilisées dans ces pages. Le dossier « style » contient les fichiers CSS correspondant aux styles utilisés dans les pages JSF et dans les cartes SVG. Enfin le fichier « common.xhtml » se trouve dans le dossier « templates » et il sert de modèle de présentation à toutes les pages.
Le dossier « src » se compose de plusieurs package :

- GIU : pour la gestion des utilisateurs.
- GF : pour la gestion des films
- GRE : pour la gestions des recommandations et des évaluations
- GVC : pour la gestion de visualisation des cartes
- GBD : pour la gestion de la base de données

3. Technologies utilisées

La figure ci-dessous montre les principales technologies misent en jeu dans le fonctionnement



de l'application. Le choix des technologies est ensuite brièvement détaillé.

Langage de programmation

Le système COCoFil2 ayant été implémenté en Java, les modules de COCoFil3 qui compléteront le système existant seront implémentés également en Java1.5. En effet, le

langage Java à pour qualité d'être robuste, simple, rapide et surtout portable. De plus, le langage Java se présente comme un candidat idéal pour la conception d'applications Web, en offrant notamment :

- des bibliothèques conçues spécialement pour les besoins du web (gestion des requêtes HTTP, des cookies, des sessions, etc.)
- un langage de script (Java Server Pages et taglibs JSTL) simple et efficace
- des frameworks nombreux et réutilisables (dont JSF présenté ci-dessous)
- des moteurs d'exécution (serveur Web) libres ou propriétaires pour répondre à tous les besoins

Nous présenterons ci-dessous le framework JSF ainsi que le serveur Web Tomcat.

Les outils offerts pour le développement en Java assurent la production d'application Web robustes et performantes.

Serveur Web

Le serveur Web utilisé est Apache Tomcat (5.5.23) qui est issu du projet Jakarta. Tomcat est un conteneur de servlet J2EE supportant JSP. Tomcat a été écrit en langage Java, il peut donc s'exécuter via la JVM (machine virtuelle java) sur n'importe quel système d'exploitation.

Base de données

MySQL est un serveur de bases de données relationnelles SQL. L'accès à la base dans le code se fait via JDBC connector 5.4. MySQL et JDBC connector sont tous deux des logiciels libres.
Pour effectuer les opérations d'administration de la base comme l'exécution des scripts de création et remplissage des tables ou encore la suppression d'un tuple suite à un test, on utilise l'interface graphique proposé par phpMyAdmin (inclut dans EasyPHP). phpMyAdmin offre la possibilité d'administrer n'importe quel serveur MySQL et permet d'effectuer la plupart des tâches courantes.

Le framework JSF

JavaServer Faces (JSF) est un framework Java, pour le développement d'application Web, utilisant l'architecture J2EE. Le but de JSF est de fournir une interface de programmation permettant de manipuler l'interface Web sans avoir à recourir à du code HTML ou JavaScript. Les spécifications de la version 1.2 sont définies dans la JSR 252 et requièrent les bibliothèques suivantes :

- Servlets version 2.5
- JSP version 2.1
- Java version 1.5 (J2SE 5.0)

Le format SVG

Scalable Vector Graphics (SVG) est une spécification du W3C. C'est un format de fichier basé sur XML permettant de décrire des ensembles de graphiques vectoriels. Les images SVG seront utilisées pour représenter les cartes des communautés. SVG peut être visualisé nativement avec certains navigateurs Web mais pour d'autres, l'installation d'un plug-in sur le poste de l'utilisateur est nécessaire. En effet pour Internet Explorer, il est nécessaire d'installer le « viewer » d'Adobe disponible gratuitement (environ 2Mo) à cette adresse : http://www.adobe.com/svg/main.html

4. Bases de données

Pour réaliser l'application, nous avons à notre disposition une plateforme de test mis en place par le projet APMD, dont le projet COCoFil3 fait partie. Cette plateforme inclut quatre bases de données relationnelles contenant des informations à propos des films, nommées :

- **MovieLens** : Base de données relationnelle contenant des évaluations d'utilisateur à propos des films. C'est un jeu de données important (plus d'un million d'évaluations) fourni par MovieLens.
- **SmallMovieLens** : C'est une version antérieure de MovieLens contenant un jeu de données plus restreint (100 000 évaluations).
- **IMDb** : Une base de données relationnelle qui fournit les informations sur 858961 films parmi lesquels on retrouve les films évalués dans la base « Movielens ».
- **Integration** : Une base de données relationnelle intégrant les données de MovieLens et IMDb (intégrant par intersection). En d'autres termes, elle intègre les informations sur les films à la base de données « Movielens » qui ne contient que les évaluations des utilisateurs sur les films.

Pour chaque base de données, la plateforme de test nous fournit :

- un fichier ZIP contenant toutes les tables au format texte.
- un fichier ZIP contenant les scripts pour créer les tables et les charger dans une base de données Oracle.
- un fichier ZIP contenant une base de données Microsoft Access.

Afin d'utiliser les données avec mySQL, les scripts Oracle fournis doivent être modifiés, en particulier pour quelques variations de type entre les deux langages ou encore quelques changements de syntaxe.

La base utilisée pour le projet sera la base **Integration**, elle nous fournit les informations sur les utilisateurs, les évaluations faites par les utilisateurs et les informations sur les films.
De plus, des tables nécessaires aux calculs des recommandations ou à la production des cartes seront ajoutées. Certaines tables seront également modifiées pour intégrer des données supplémentaires, comme par exemple le nom d'utilisateur d'un utilisateur.
Toutes les données sur les films fournies par IMDb ne seront pas utilisées. Seules les données jugées les plus importantes et pertinentes, comme les acteurs ou le réalisateur, seront utilisées. Les données de moindre importance, comme les détails sur les décors ou les costumes, ne seront pas forcements exploitées.

Le détail complet (attributs, types, clefs, …) des tables se trouve dans le dossier de conception détaillé.

5. Modélisation

Nous présentons ci-dessous le diagramme de classe de l'application COCoFil3. Il montre les objets manipulés dans l'application ainsi que les relations entres ces objets.

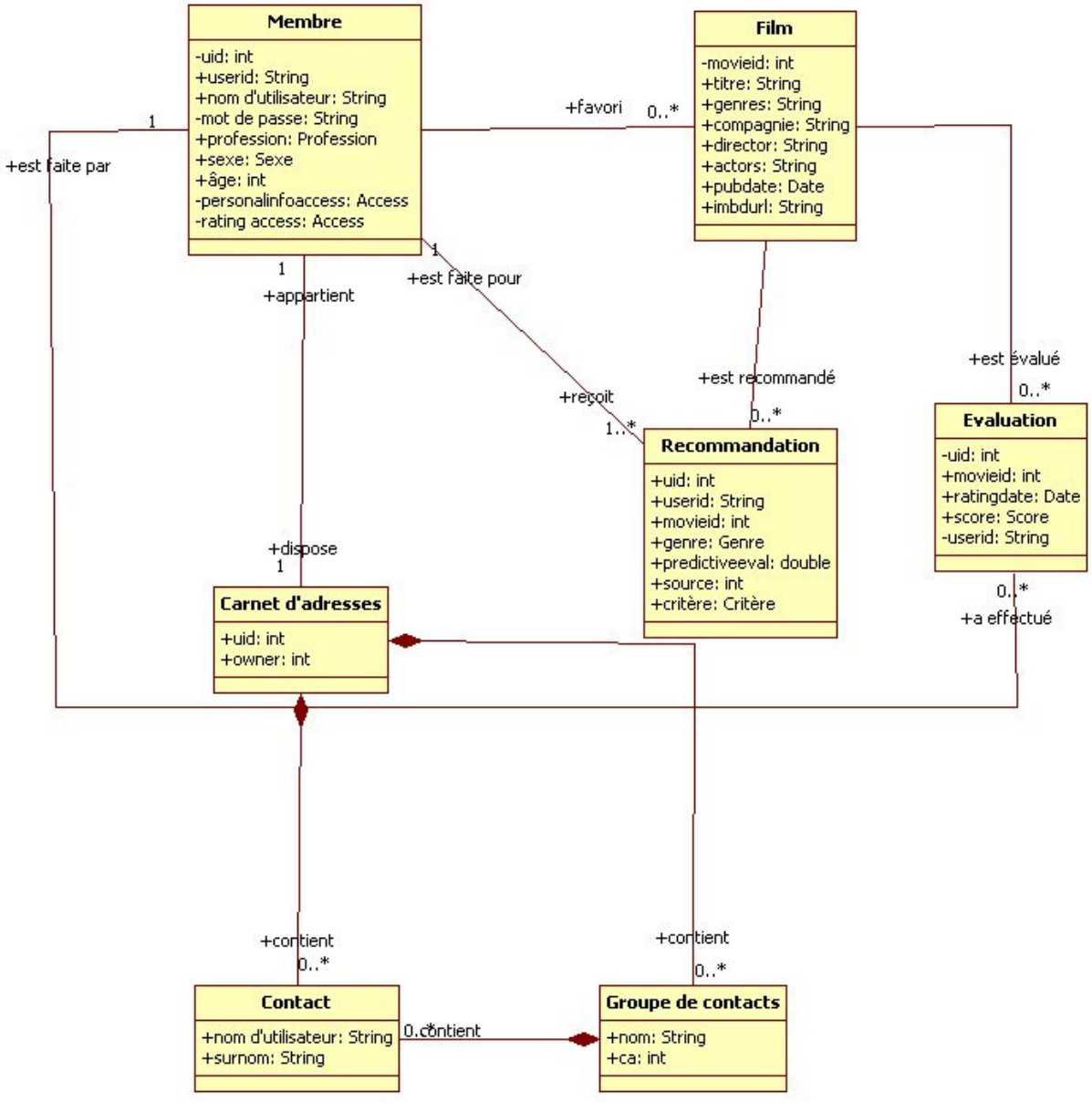


Figure 7: Diagramme de classe de l'application COCoFil3

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Dossier de conception détaillée

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	06/08/2007
Version	1.2

Historique des versions

Version	Date	Modifications
1.0	14/05/2007	Début de la rédaction
1.1	09/07/2007	Ajout suite à la conception détaillée de l'incrément 2
1.2	06/08/2007	Ajout suite à la conception détaillée de l'incrément 3

Sommaire

1. Introduction

1.1 But et portée du document

Ce document représente la conception détaillée du projet COCoFil3. Nous présentons de façon plus détaillée les modules de notre application (cf. Dossier de conception globale) dans le but d'exposer l'organisation du développement de l'application COCoFil3.

Ce document s'adresse :

- Au responsable de stage : DENOS Nathalie
- à l'équipe MRIM
- au consultant : CUNIN Pierre-Yves
- à l'équipe du projet : ARGOUD Guillaume et CAMARA Fatoumata Goundo

1.2 A propos du projet

Le projet COCoFil3 consiste à développer une application Web autour d'un système de recommandations basé sur les films. Au cours de précédents travaux, le noyau d'un système de recommandation orienté vers les communautés, nommé COCoFil2 a été développé par An-Te Nguyen. COCoFil3 vise à rendre ce système interactif et dynamique.

2. Rappel de l'architecture

Ici, nous rappelons l'architecture de notre application

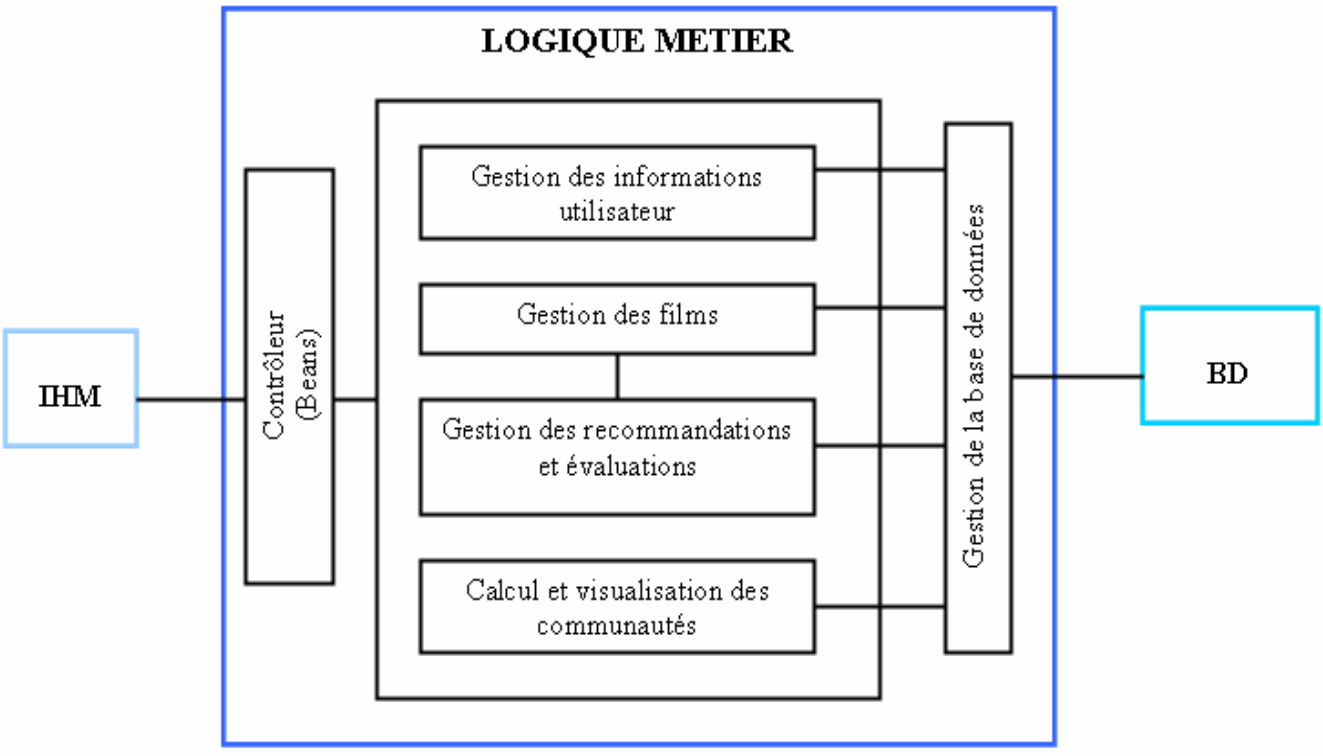
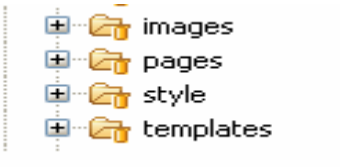


Figure 1: Architecture de l'application COCoFil3

3. Description détaillée des modules

3.1 Le module IHM



Les pages sont toutes basées sur la même structure décrite par le fichier **common.xhtml**. Le titre et le contenu de la page sont spécifiés à l'aide des facelets composition, insert et define. Les pages se trouvent dans le dossier **pages**. Le dossier **style** contient les fichiers css utilisés pour la mise en forme des pages. Le dossier carte contient les cartes des communautés au format SVG.

Nom de la page	Description
404	Affichée dans le cas où la page demandée n'est pas trouvée (par exemple suite à une erreur de saisie de l'utilisateur dans l'URL)
allrecommandations	Affiche tous les films recommandés à l'utilisateur depuis son inscription excepté les films celles qui lui ont été faites depuis sa dernière connexion.
carnetAdresses	Affiche tous les contacts de l'utilisateur classés par groupe. Propose

	également toutes les actions permettant d'agir sur le carnet d'adresse. Utilise le bean GIU.AddressBook.
carteContenu	Affiche la carte représentant les utilisateurs repartis dans des communautés suivant le critère contenu. Utilise le bean CVC.Cartes.
carteQualite	Affiche la carte représentant les utilisateurs repartis dans des communautés suivant le critère qualité. Utilise le bean CVC.Cartes.
evaluations	Affiche toutes les évaluations que l'utilisateur a fait depuis son inscription. Utilise le bean GRE.GREBean.
favoris	Affiche tous les favoris de l'utilisateur. Utilise le bean GIU.Favori
fiche	Page de description d'un film. Utilise le bean GF.Movie.
InfoCommunaute	Page affichant le profil de la communauté de l'utilisateur courrant.
inscription	Page de saisie des informations demandées lors de l'inscription. Utilise le bean GIU.UserBean.
inscriptionSucces	Page de confirmation suite à la validation de l'inscription d'un utilisateur.
modifierDonnees	Page de modification des données personnelles de l'utilisateur. Utilise le bean GIU.ProfilBean.
modifierDonneesSucces	Page de confirmation suite à la validation des modifications des données personnelles.
newrecommandations	Affiche les recommandations ajoutées entre la dernière connexion et la connexion courante de l'utilisateur. Utilise le bean GRE.GREBean.
presentation	Page décrivant le système COCoFil3 : son objectif et ses fonctionnalités.
profil	Page permettant à l'utilisateur courant de choisir les différentes options pour gérer son profil.
profilCommunaute	Page permettant à l'utilisateur courant de comparer son profil à celui d'une communauté (choisie à travers les cartes). Utilise le bean CVC.CommunityProfile.
profilUtilisateur	Page permettant à l'utilisateur courant de comparer son profil à celui d'un autre utilisateur (choisie à travers les cartes). Utilise le bean CVC.UserProfile.
recherche	Affiche le résultat d'une recherche de film par mots-clés sur le titre. Utilise le bean GF.Search.
Vposition	Page permettant à l'utilisateur courrant d'accéder à la visualisation de sa communauté suivant différents critères.

Le dossier **images** contient toutes les images nécessaires à l'affichage des pages (logos, bannière, etc.). Le dossier images/etoiles contient les images utiliser pour représenter la note de l'utilisateur seul ou la note de l'utilisateur et celle d'un autre. Le dossier images/contacts contient les images utilisées pour l'affichage et la gestion du carnet d'adresses. Le dossier

images/fleches contient les images utilisées pour l'affichage des listes de films permettant de passer d'une page à une autre (première, dernière, précédentes et suivantes).

Le fichier de nom **etoileX.png** signifie que l'utilisateur a donné une note de X sur 5 à un film. Tous les utilisateurs devront avoir le même « barème » pour noter un film afin que les notes soient homogènes d'un utilisateur à un autre.

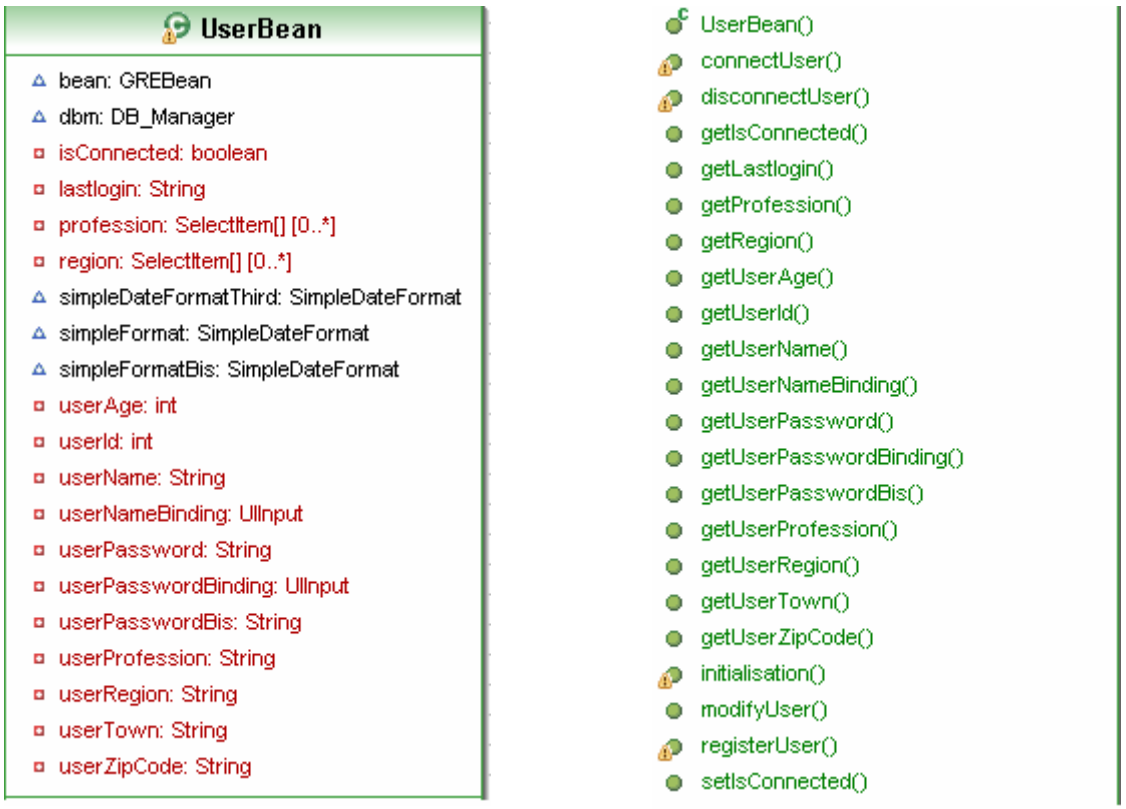
Le fichier de nom **etoileXY.png** est utilisé pour comparer la note X de l'utilisateur courant avec la note Y de l'utilisateur sélectionné.
Par exemple, l'image  est utilisée pour un film qui a été évalué à 3 sur 5 par l'utilisateur actuel et à 4 sur 5 par un autre utilisateur.

3.2 Le module Contrôleur

Nous avons choisi d'utiliser le framework JSF (Java Server Faces). JSF est basé sur le modèle MVC (Modele Vue Controller). Dans notre architecture, le module « contrôleur » prend en charge le rôle du « controller ».
Dans notre application, le contrôleur est représenté par des managed beans. Ces beans sont des java beans classiques. Chaque bean correspond à une classe java.

- ***UserBean.java***

Ce bean se trouve dans le package « GIU ». Lorsqu'un utilisateur souhaite s'inscrire, se connecter, se déconnecter ou modifier ses données personnelles, les requêtes correspondantes sont envoyés à ce bean.
Pendant toute la durée de la session de l'utilisateur, ce bean stocke l'ensemble de ses informations personnelles : nom d'utilisateur, age, adresse et la date de sa dernière connexion.
Le détail du contenu du bean est donné ci-après.


UserBean
bean: GREBean
dbm: DB_Manager
isConnected: boolean
lastlogin: String
profession: SelectItem[] [0..*]
region: SelectItem[] [0..*]
simpleDateFormatThird: SimpleDateFormat
simpleFormat: SimpleDateFormat
simpleFormatBis: SimpleDateFormat
userAge: int
userId: int
userName: String
userNameBinding: UIInput
userPassword: String
userPasswordBinding: UIInput
userPasswordBis: String
userProfession: String
userRegion: String
userTown: String
userZipCode: String
UserBean()
connectUser()
disconnectUser()
getIsConnected()
getLastlogin()
getProfession()
getRegion()
getUserAge()
getUserId()
getUserName()
getUserNameBinding()
getUserPassword()
getUserPasswordBinding()
getUserPasswordBis()
getUserProfession()
getUserRegion()
getUserTown()
getUserZipCode()
initialisation()
modifyUser()
registerUser()
setIsConnected()

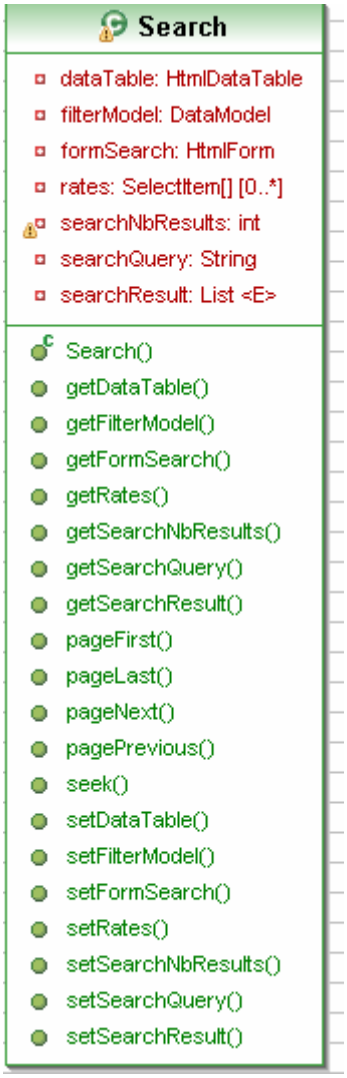

setLastlogin()
setProfession()
setRegion()
setUserAge()
setUserId()
setUserName()
setUserNameBinding()
setUserPassword()
setUserPasswordBinding()
setUserPasswordBis()
setUserProfession()
setUserRegion()
setUserTown()
setUserZipCode()
validate()

- ***GREBean.java***

Ce bean se trouve dans le package « GRE ». Lorsqu'un utilisateur souhaite consulter ses nouvelles recommandations, toutes ses recommandations ou ses évaluations, les requêtes correspondantes sont envoyées à ce bean.
Pendant toute la durée de la session de l'utilisateur, ce bean stocke l'ensemble des recommandations et évaluations de l'utilisateur.
Le détail du contenu du bean est donné ci-après.

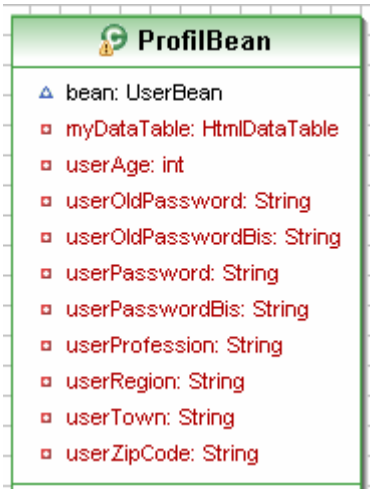
- ***Search.java***

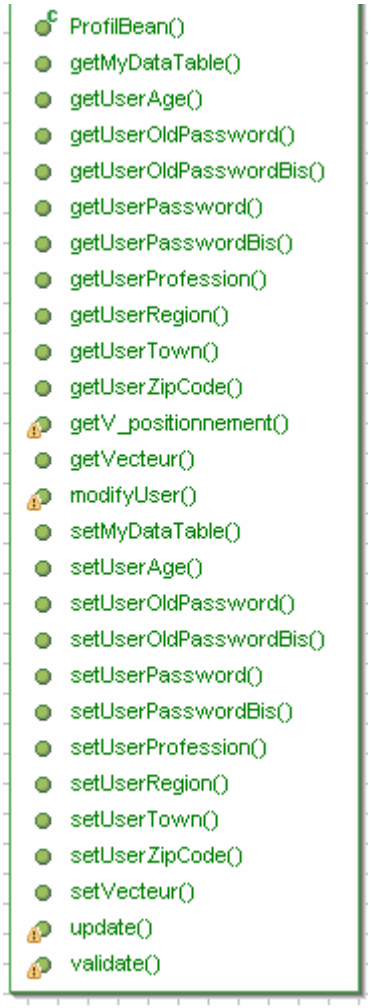
Ce bean se trouve dans le package « GF ». Lorsqu'un utilisateur souhaite effectuer une recherche, la requête correspondante est envoyée à ce bean. Le détail du contenu du bean est donné ci-après.



- ***ProfilBean.java***

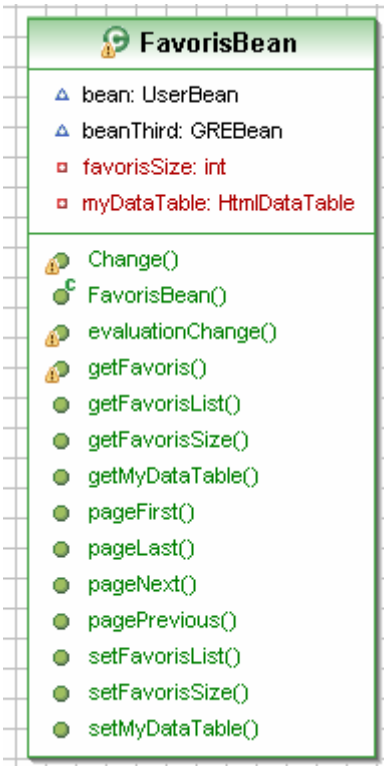
Ce bean se trouve dans le package « GIU ». Lorsqu'un utilisateur souhaite visualiser son vecteur de positionnement dans les communautés, modifier ses données personnelles ou modifier ses paramètres par défaut, les requêtes correspondantes sont envoyées à ce bean. Le détail du contenu du bean est donné ci-après.





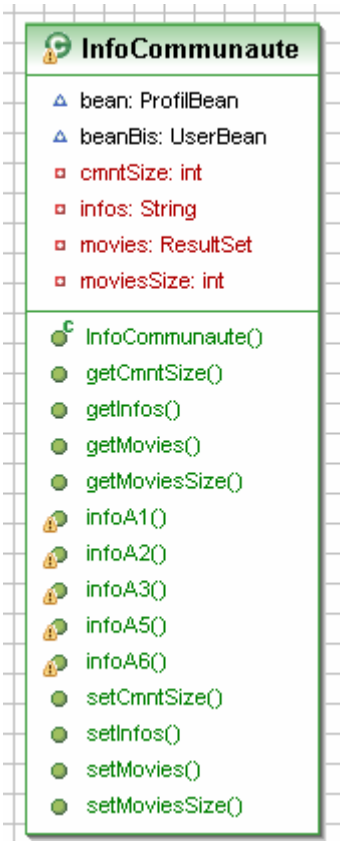
- ***FavorisBean.java***

Ce bean se trouve dans le package « GIU ». Lorsqu'un utilisateur souhaite consulter ses favoris, ajouter un film à ses favoris ou supprimer un film des favoris, les requêtes correspondantes sont envoyées à ce bean.
Pendant toute la durée de la session de l'utilisateur, ce bean stocke l'ensemble des favoris de l'utilisateur. Le détail du contenu du bean est donné ci-après.



- ***InfoCommunaute.java***

Lorsqu'un utilisateur souhaite connaître des informations sur sa communauté selon les critères âge, profession, localisation, genre ou évaluation, la requête correspondante est envoyée à ce bean. Le détail du contenu du bean est donné ci-après.

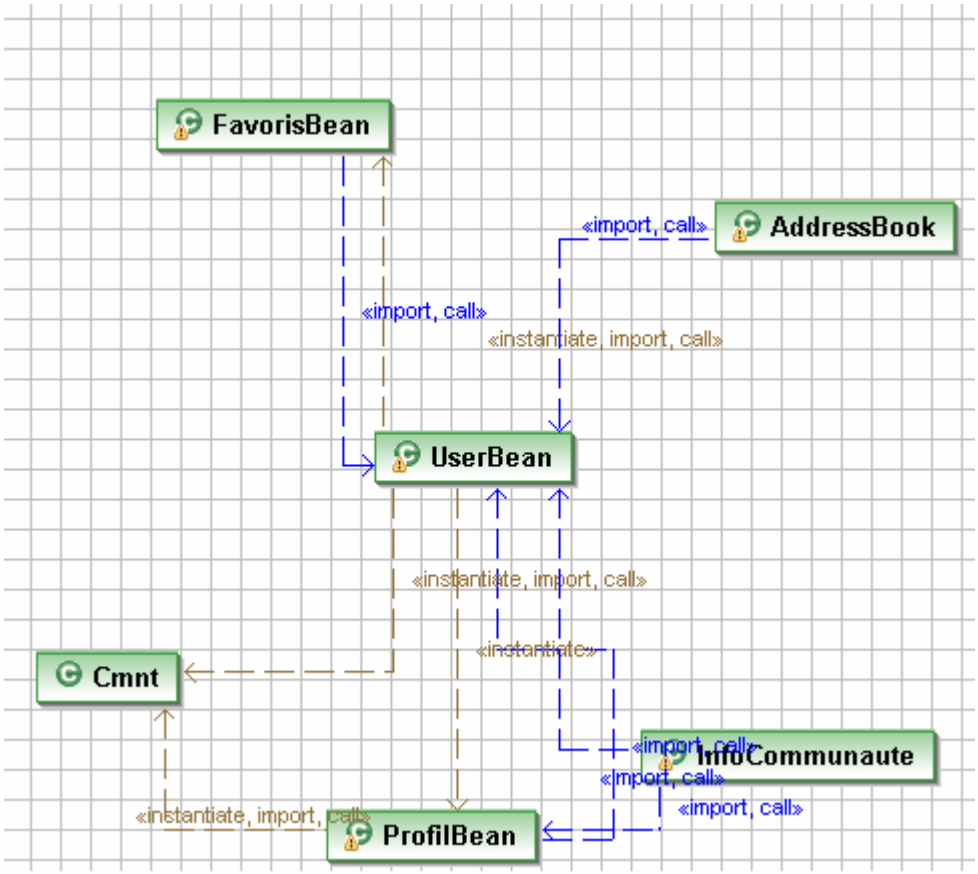


- ***AddressBook.java***

Lorsqu'un utilisateur consulte son carnet d'adresses, ajoute ou supprime un contact ou un groupe, bloque ou débloque un contact, la requête correspondante est envoyée à ce bean. Le détail du contenu du bean est donné ci-après.

3.3 Le module Gestion des informations utilisateurs (GIU)

Le module « Gestion des données utilisateur » gère les informations relatives aux utilisateurs. Il contient six classes :



- ***Userbean.java*** (cf. section 3.2)

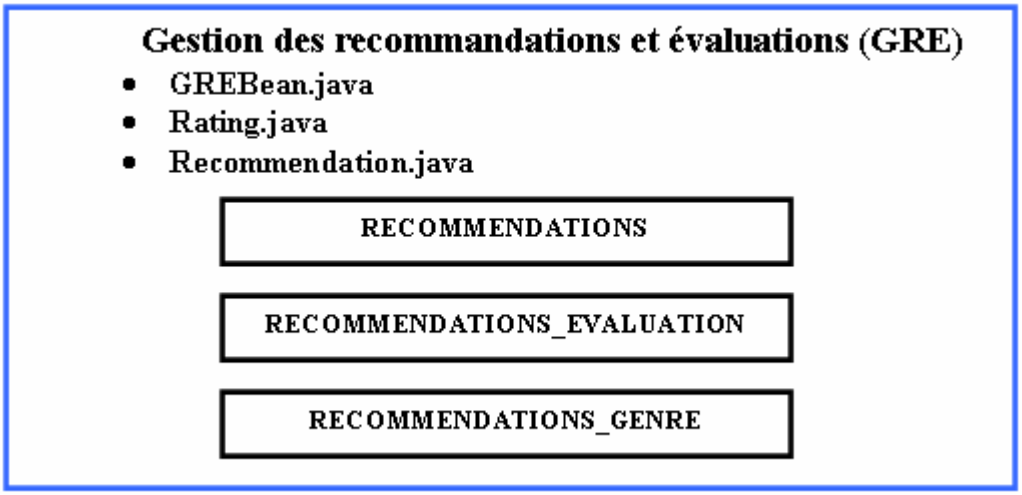
- ***Cmnt.java*** : cette classe est utilisée par le bean « UserBean.java » pour effectuer le calcul de communautés selon les critères âge, profession et localisation géographique lorsqu'un utilisateur s'inscrit. Elle est également utilisée par la classe « ProfilBean .java» pour calculer les communautés selon les critères age, profession et localisation lorsqu'un utilisateur modifie ses données personnelles. Le détail du contenu de la classe est donné ci-dessous.

- ***ProfilBean.java*** (cf. section 3.2)

- ***FavorisBean.java*** (cf. section 3.2)

- ***InfoCommunaute.java*** (cf. section 3.2)
- ***AddresBook.java*** (cf. section 3.2)

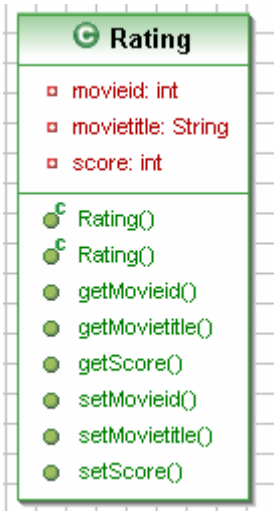
3.4 Le module Gestion des recommandations et évaluations (GRE)



Le rôle de ce module est de gérer toutes les recommandations faites à l'utilisateur et toutes ses évaluations.
Ce module contient :

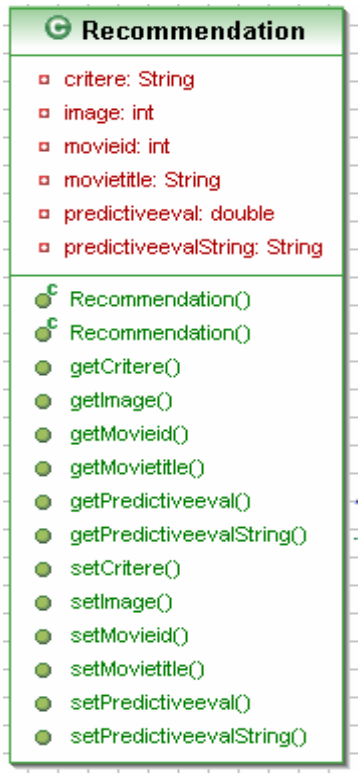
- ***GREBean.java*** (cf. section 3.2)

- ***Rating.java*** : Cette classe représente un objet évaluation qui est caractérise par l'identifiant de l'utilisateur, l'identifiant, du film évalué et la noté attribuée au film par l'utilisateur. Le détail du contenu de cette classe est donné ci-dessous.



- ***Recommendation.java*** : Cette classe représente un objet recommandation qui est caractérisé par l'identifiant de l'utilisateur auquel la recommandation est destinée, l'identifiant du film recommandé, le critère sur lequel la recommandation repose,

la note prédite et d'autres attributs utilisés pour l'affichage des recommandations. Le détail du contenu de la classe est donné ci-dessous.



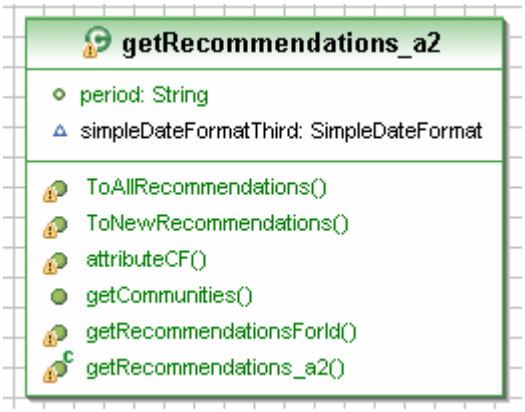
En plus des classes citées ci-dessus, le module « GRE » contient 3 sous modules : « RECOMMENDATIONS »,« RECOMMENDATIONS_GENRE »,« RECOMMENDATIONS_EVALUSATION » tous destinés à réaliser les calculs de recommandation.
Lorsqu'un utilisateur souhaite consulter ses recommandations, la requête est dirigée vers le bean « GREBean.java » (cf. section 3.2) qui appelle les classes nécessaires dans les trois modules cités ci-dessus afin d'effectuer les calculs de recommandation.

3.4.1 Le module RECOMMENDATIONS

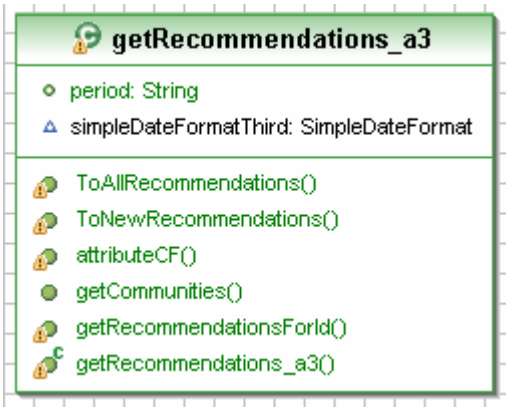
Le module « RECOMMENDATIONS » sert à effectuer les calculs de recommandation selon les critères âge, profession et localisation géographique. Il contient trois classes indépendantes : getRecommendations_a1.java, getRecommendations_a2.java et getRecommendations_a1.java correspondantes respectivement aux calculs de recommandation selon les critères âge, profession et localisation géographique.

- ***getRecommendations_a1.java :*** Pour un utilisateur donné, cette classe calcule les recommandations selon le critère âge. Le détail du contenu de la classe est donné ci-dessous.

- ***getRecommendations_a2.java*** : Pour un utilisateur donné, cette classe calcule les recommandations selon le critère profession. Le détail du contenu de la classe est donné ci-dessous.

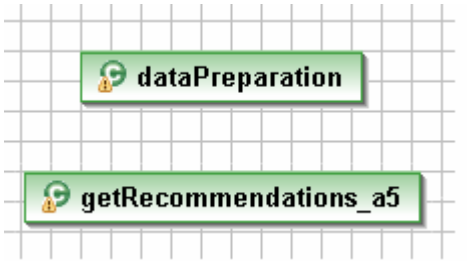


- ***getRecommendations_a3.java*** : Pour un utilisateur donné, cette classe calcule les recommandations selon le critère localisation géographique. Le détail du contenu de la classe est donné ci-dessous.

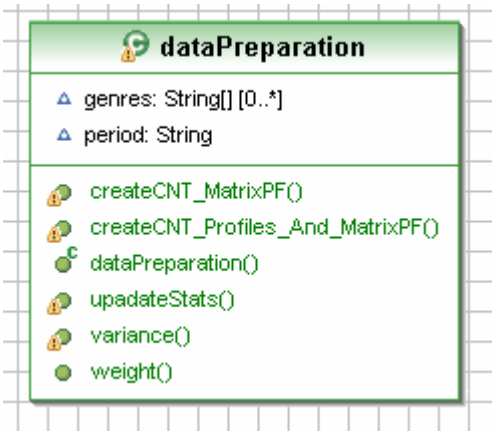


3.4.2 Le module RECOMMENDATIONS_GENRE

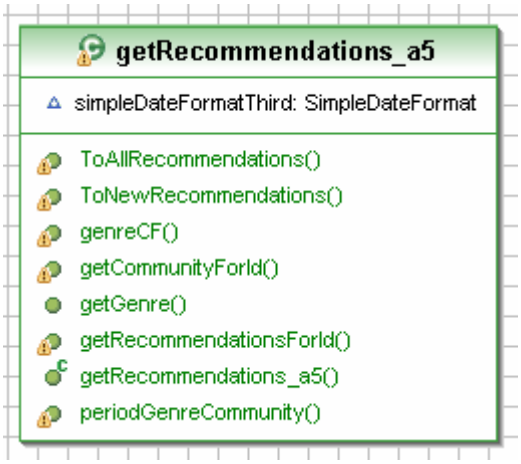
Ce module permet d'effectuer les calculs de recommandation selon le critère genre. Il contient deux classes :



- ***dataPreparation.java*** : Cette classe réalise la première étape du calcul de recommandation selon le critère genre : la préparation de données. Il s'agit de calculs statistiques permettant de calcul le poids par genre de film pour un utilisateur donné. Le genre qui a le plus grand poids est considéré comme le genre préféré de l'utilisateur. Le détail du contenu de cette classe est donné ci-dessous.



- ***getRecommendations_a5.java*** : Cette classe réalise la deuxième étape du calcul de recommandation selon le critère genre : la production de recommandation. Elle sert des résultats des calculs effectués dans l'étape de préparation des données. Le détail du contenu de cette classe est donné ci-dessous.

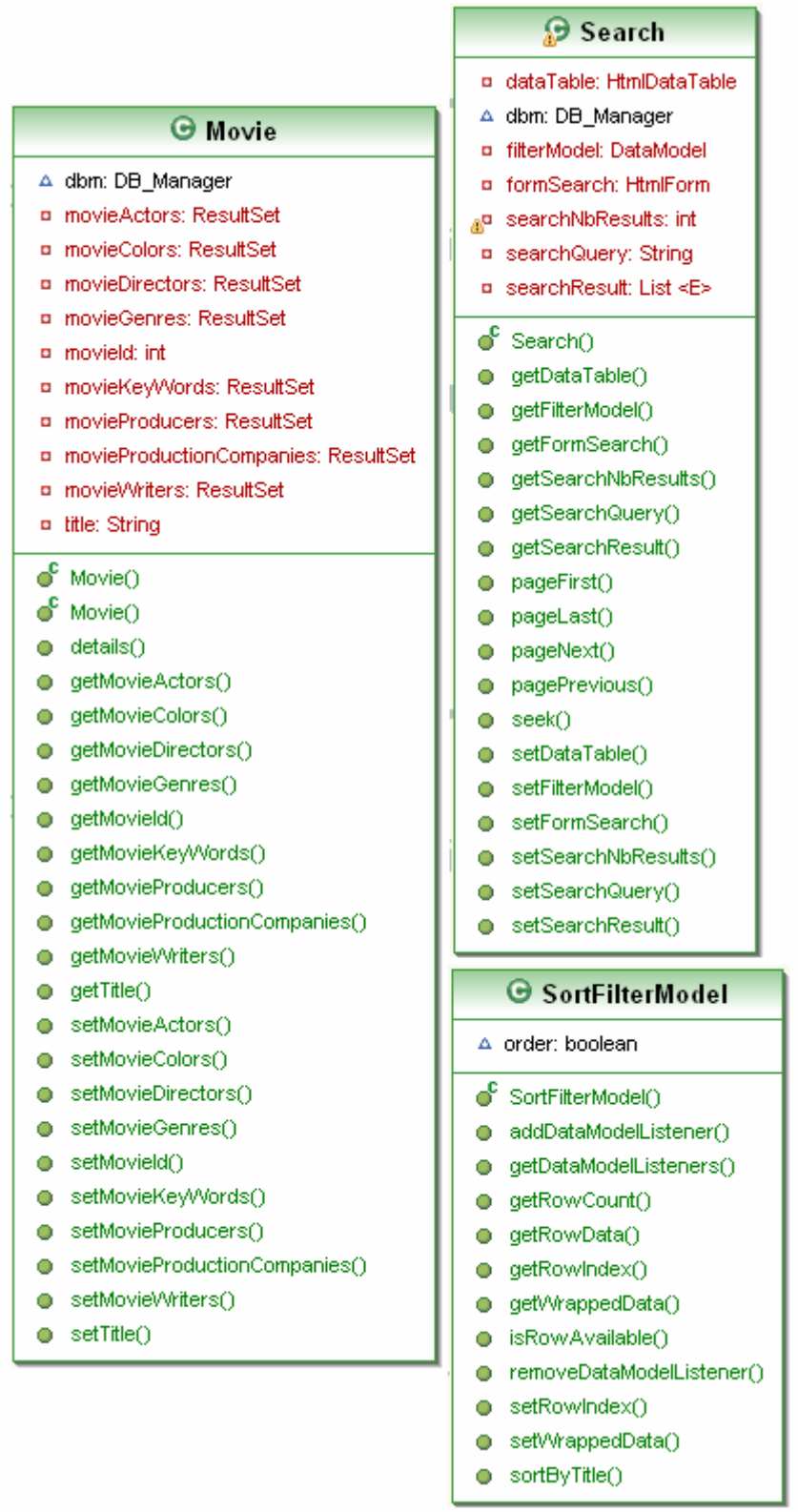


3.4.3 Le module RECOMMENDATIONS_EVALUATION

Ce module contient une seule classe (getRecommendations_a6.java) servant à effectuer les calculs de recommandation selon le critère évaluation. Le détail du contenu de cette classe est donné ci-dessous.

3.5 Le module Gestion des films (GF)

Ce bean se trouve dans le package « GF ». Lorsqu'un utilisateur souhaite rechercher un film ou consulter les détails d'un film les requêtes correspondantes sont envoyées à ce bean.

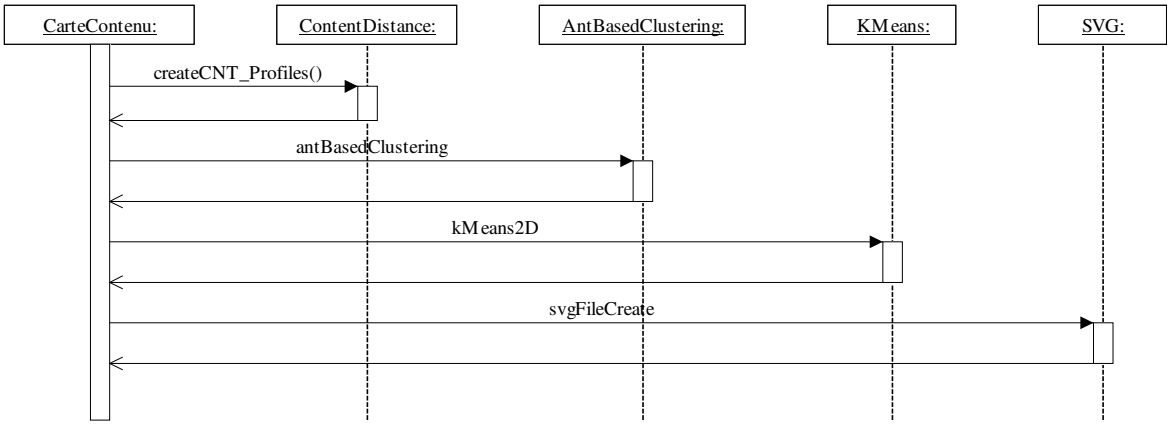


- ***Movie.java*** : cette classe permet d'extraire de la base de données les informations relatives à un film : acteurs, producteurs, scénaristes, directeurs, couleurs, genres, mots-clés, compagnies de productions.

- ***Search.java*** : permet de faire une recherche de films suivant un ou plusieurs mots-clés portant sur le titre du film.
- ***SortFilterModel.java*** : permet de trier un tableau suivant l'une de ses colonnes (par exemple suivant l'ordre alphabétique sur les titres des films).

3.6 Le module calcul et visualisations des communautés (CVC)

Ce module contient le calcul des cartes suivant deux critères : le contenu ou la qualité. Une fois que la position de chaque utilisateur a été calculée et que les utilisateurs ont été répartis en communautés, le module permet de créer une carte au format SVG pour permettre à l'utilisateur de visualiser le résultat.



Le diagramme de séquence ci-dessus montre l'enchaînement des taches permettant de créer une carte pour le critère contenu, la démarche étant analogue pour le critère qualité.

La classe ContentDistance (respectivement QualityDistance) calcul les distances entre utilisateurs. Avec le jeu de données SmallMovieLens, il faut environ 1 min 30 s pour préparer les données suivant le critère contenu et 30 minutes suivant le critères qualité. En effet, pour le calcul des distances selon le critère contenu, on utilise les évaluations d'un utilisateur pour produire des poids pour chaque genre de film (représentant l'attirance de l'utilisateur pour ce style). Le système fait donc un simple calcul de moyenne des évaluations en fonction des genres **pour chaque** utilisateur. Le poids ainsi trouvé permet d'établir les distances entre les utilisateurs. En revanche pour les distances selon le critère qualité, les évaluations de chaque utilisateur sont comparées à celle de **tous les autres** utilisateurs afin d'établir la distance entre chacun d'eux.

Le calcul de distance suivant le critère qualité est donc beaucoup plus long que pour le critère contenu mais le résultat est meilleur car on s'intéresse non pas simplement au genre. Les comparaisons entre évaluations d'un même film permettent d'établir des similarités plus finement (néanmoins il faut un nombre d'évaluations conséquent et en commun entre les utilisateurs). Le passage au jeu de données MovieLens multiplie le nombre d'utilisateurs par

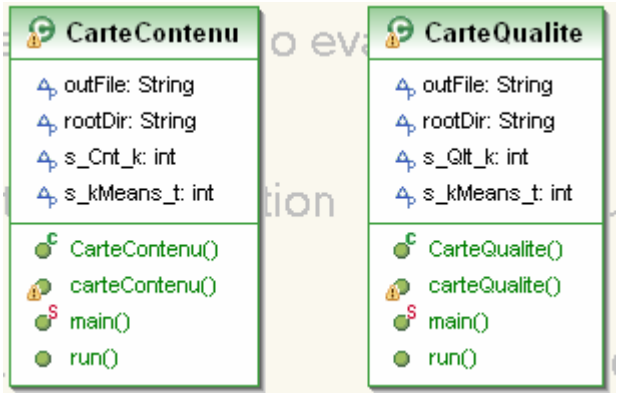
six et le nombre d'évaluations par dix provoquant une croissance exponentiellement des temps de calcul.

Une fois le calcul des distances terminé, on procède à une classification (sur un plan 2D) des utilisateurs à l'aide de l'algorithme des fourmis artificielles (**AntBasedClustering**). A partir de cette classification, l'algorithme des k-moyenne (**KMeans**) nous produit une nouvelle classification des utilisateurs suivant k communautés. Pour visualiser ces communautés, on génère une carte au format SVG à l'aide de la méthode **svgFileCreate** de la classe **SVG.java**.

Nous allons maintenant détailler le contenu de chacun de ces modules.

3.6.1 Les modules CarteContenu et CarteQualite

Les classes **CarteContenu** et **CarteQualite** héritent de la classe Thread du package java.lang. Les processus de calcul des cartes étant des processus longs, il est indispensable qu'ils ne monopolisent pas les ressources et qu'ils s'exécutent parallèlement au reste des taches de l'application. Toutes les variables servant aux algorithmes sont regroupées dans ces classes pour pouvoir les paramétrer facilement.



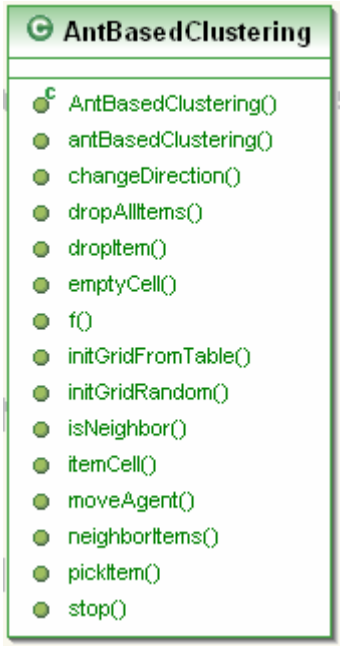
3.6.2 Les Modules ContentDistance et QualityDistance

ContentDistance et QualityDistance calculent les matrices de distance entre chaque utilisateur suivant le critère contenu et le critère qualité.

La persistance de ces matrices se fait dans les tables Cnt_Distance et Cnt_Quality.
Pour 943 utilisateurs (SmallMovieLens) : 444.153 tuples par tables (943*942/2)
Pour 6040 utilisateurs (MovieLens) : 18.237.780 tuples par tables (6040*6039/2)

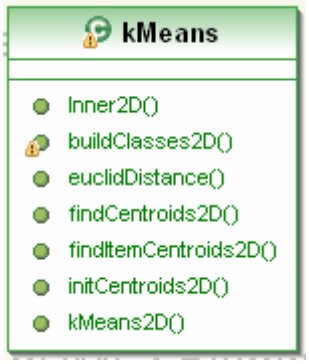
3.6.3 Le module AntBasedClustering

Suite au calcul des distances entre utilisateurs, l'algorithme des fourmis va positionner les utilisateurs sur un plan 2D en simulant le comportement réel des fourmis.



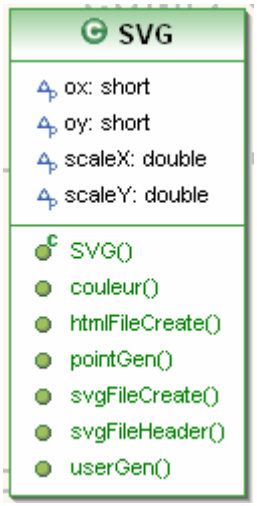
3.6.4 Le module KMeans

Après avoir obtenu la position de chaque utilisateur avec l'algorithme des fourmis artificielle, l'algorithme des K-moyennes permet de classifier les utilisateurs en K classes. Pour cela l'algorithme prend K centre de gravité au hasard et rattache chaque utilisateurs au centre le plus proche de lui. On recalcul alors les K nouveaux centres pour chaque classes précédement créées. On répète le processus un nombre i d'itérations données ou jusqu'à ce que les K centres se stabilisent suffisamment.



3.6.5 Le module SVG

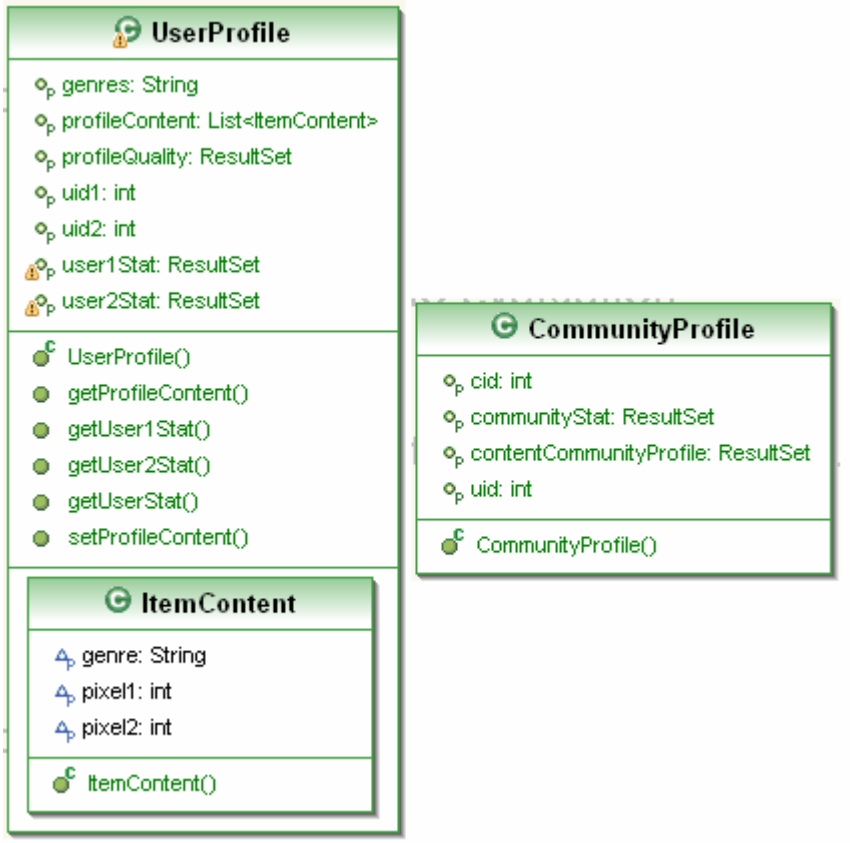
Le module SVG crée un fichier au format SVG représentant les communautés précédemment



calculées.

3.6.6 Les modules UserProfile et CommunityProfile

Ces modules permettent à l'utilisateur de comparer son profil à celui d'un autre utilisateur ou au prifl d'une communauté.

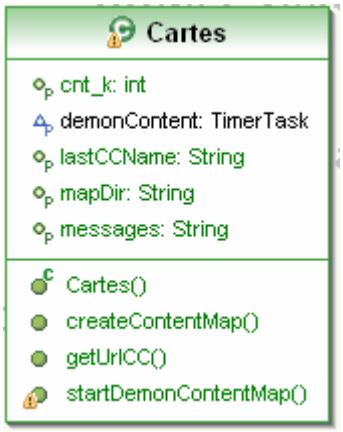


Pour comparer son profil à un autre utilisateur, les moyennes des évaluations par genre de films ainsi que les évaluations en communs des deux utilisateurs sont affichées.

Pour observer le profil d'une communauté, on calcul les films les plus vus par la communauté ainsi que la moyenne des évaluations pour chacun de ces films par les membres de cette communauté.

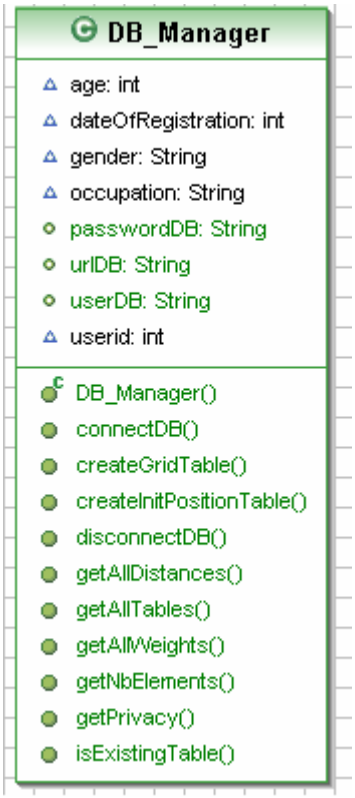
3.6.7 Les modules Cartes, DemonContent et DemonQuality

Les classes **DemonContent** et **DemonQuality** héritent de la classe `TimerTask` du package java.util. Les processus de calcul des cartes étant des processus longs, il est envisagé de les exécuter à des instants précis (par exemple tous les jours à minuit). La classe **Cartes** sert à initialiser les classes DemonContent et DemonQuality.



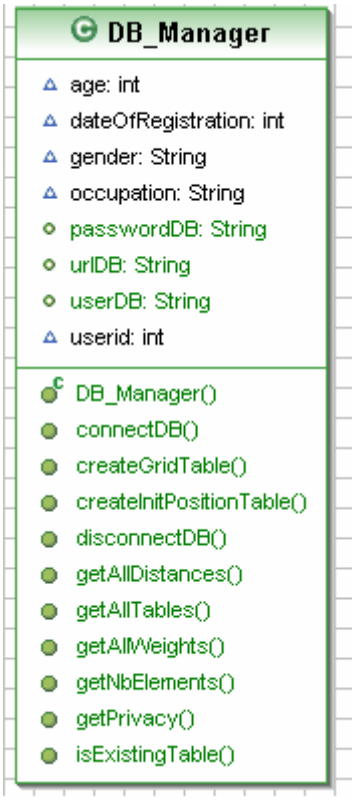
3.7 Le module Gestion de la base de données (GBD)

Le module « Gestion de la base de données » contient une seule classe (DB_Manager.java) qui permet de réaliser les accès au module « «BD » représentant la base de données. Il permet ainsi de se connecter/déconnecter de la base de données, de consulter toutes les tables de la base, de connaître le nombre d'éléments d'une table, ou encore de savoir si une table donnée existe ou non. . Le détail du contenu de cette classe est donné ci-dessous.



3.8 Le module Gestion de la base de données (GBD)

Le module « Gestion de la base de données » contient une seule classe (DB_Manager.java) qui permet de réaliser les accès au module « «BD » représentant la base de données. Il permet ainsi de se connecter/déconnecter de la base de données, de consulter toutes les tables de la base, de connaître le nombre d'éléments d'une table, ou encore de savoir si une table donnée existe ou non. . Le détail du contenu de cette classe est donné ci-dessous.



3.9 Le module BD

Ce module représente la base de données. Comme nous l'avons décrit dans le dossier de conception globale, cette base de données sur la base **Intégration** à laquelle nous ajouterons les tables nécessaires aux calculs de recommandation, la mémorisation de la dernière connexion de l'utilisateur, des informations sur les communautés, etc.…
Ainsi la base de données utilisée par l'application COCoFil3 est composée de 27 tables.

3.9.1 Description des tables

1. La table « allrecommendations »

Cette table stocke toutes les recommandations faites aux utilisateurs exceptées celles qui leur ont été faites après leur dernière connexion et celles qu'ils ont déjà évaluées. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ uid : smallint(4) ▪ movieid : smallint(4) ▪ critere : varchar(200) ▪ predictiveeval : double (11,8) ▪ date : datetime Clé primaire : uid, movieid, critere	Néant

2. La table « cmnt_age »

La table « cmnt_age » fournit les informations sur les communautés disponibles selon le critère âge. Au total, nous avons sept communautés :

- Ado (1 à 15 ans)
- Jeune (16 à 24 ans)
- Adulte (25 à 44 ans)
- Mûr (45 à 59 ans)
- Ancien (60 à 90 ans)

Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ min_age : smallint(4) ▪ max_age : smallint(4) ▪ cmnt_id : smallint(4) ▪ cmnt_name : varchar(100) Clé primaire : néant	7

3. La table « cmnt_localisation »

La table « cmnt_localisation » fournit les informations sur les communautés disponibles selon le critère localisation. Au total, nous avons 22 communautés correspondantes aux 22 régions de France. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ cmnt_id : smallint(4) ▪ cmnt_name : varchar(100) Clé primaire : cmnt_id, cmnt_name	22

4. La table « cmnt_profession »

La table « cmnt_profession » fournit pour chaque profession, la communauté à laquelle elle est affectée. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ cmnt_id : smallint(4) ▪ profession : varchar(100) Clé primaire : cmnt_id, profession	23

5. La table « cnt_genrecommunity »

Cette table fournit » enregistre les vecteurs des poids qui représentent les préférences des utilisateurs sur 19 genres de films. En plus de ces informations, nous y trouvons la communauté de l'utilisateur selon le critère genre et le genre correspondant. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ period : varchar(6) ▪ uid : smallint(4) ▪ cmnt : smallint(4) ▪ genre : varchar(20) ▪ w1 : double (10,8) ▪ ……. ▪ w19 : double (10,8) ▪ userid : varchar(4) Clé primaire : period, uid	Nombre d'utilisateurs

6. La table « cnt_matrixpf »

La table « cnt_matrixpf » enregistre les vecteurs des poids qui représentent les préférences des utilisateurs sur les 19 genres de films.

Description	Tuples
Attributs : ▪ uid : smallint (4) ▪ cmnt : smallint (4) ▪ genre : varchar (20) ▪ w1 : double (10,8) ▪ ……. ▪ w19 : double (10,8) ▪ userid : varchar (4) Clé primaire : uid	Nombre d'utilisateurs

7. la table «contacts »

La table « contacts » enregistre les contacts des utilisateurs en indiquant le groupe de contacts correspondant et le statu du contact (si il est bloqué ou non). Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ ownerID : smallint(2) ▪ contactID : smallint(2) ▪ groupID : smallint(2) ▪ bloque : tinyint(1) Clé primaire : ownerID, contactID, groupID	Néant

8. La table « favoris»

La table « favoris » enregistre les favoris des utilisateurs.

Description	Tuples
Attributs : ▪ uid : smallint (4) ▪ movieid : smallint (4) Clé primaire : uid, movieid	Néant

9. La table « genre »

La table « genre » contient les 19 genres de films disponibles.

Description	Tuples
Attributs : ▪ genre : varchar (40) Clé primaire : genre	19

10. la table «groups »

La table « groups » enregistre les groupes de contacts de chaque utilisateurs. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ ownerID : smallint(2) ▪ groupID : smallint(2) ▪ groupName : varchar(50) Clé primaire : ownerID, groupID	Néant

11. La table « i_movieactors »

La table « i_movieactors » enregistre les acteurs de chaque film.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ actor : varchar(75) ▪ playinfo : varchar(110) ▪ role : text ▪ casting : int(3) Clé primaire : movieid, actor	110545

12. La table «i_movieactresses »

La table « i_movieactors » enregistre les actrices de chaque film.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ actress : varchar(75) ▪ playinfo : varchar(110) ▪ role : text ▪ casting : int(3)	48423

Clé primaire : movieid, actress	

13. La table «i_moviecolors »

La table « i_moviecolors » enregistre les couleurs de chaque film c'est-à-dire que pour chaque film elle indique si le film est en noir et blanc ou en couleur.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ colors : varchar(20) ▪ colorinfo : varchar(50) Clé primaire : movieid, color	3898

14. La table «i_moviedirectors »

La table « i_moviedirectors » enregistre le réalisateur de chaque film.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ director : varchar(40) ▪ directoralias : varchar(40) ▪ dierctorinfo : varchar(70) Clé primaire : movieid, director	4141

15. La table «i_moviekeywords »

La table « i_moviekeywords » enregistre les mots clés de chaque film.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ keyword : varchar(60) Clé primaire : movieid, keyword	105126

16. La table «i_movieproducers »

La table « i_movieproducers » enregistre le producteur de chaque film.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ producer : varchar(60)	16749

▪ producerinfo : varchar(90) Clé primaire : movieid, producer	

17. La table «i_movieproductioncompanies »

La table « i_movieproductioncompanies » enregistre la compagnie de production de chaque film.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ companyname : varchar(120) ▪ countrycode : varchar(15) ▪ companyinfo : varchar(190) Clé primaire : movieid, companyname, countrycode	9517

18. La table « i_moviewriters»

La table «i_moviewriters » enregistre les informations sur les écrivains de chaque film.

Description	Tuples
Attributs : ▪ movieid : int (4) ▪ writer : varchar(45) ▪ writerinfo : varchar(110) Clé primaire : movieid, writer	8766

19. La table «identity »

La table «identity» enregistre les informations sur les utilisateurs du système.

Description	Tuples
Attributs : ▪ uid : smallint (4) ▪ gender : varchar(1) ▪ age : tinyint(2) ▪ occupation : varchar(50) ▪ zipcode : varchar(5) ▪ town : varchar(100) ▪ region : varchar(200) ▪ dateofregistration : date ▪ persinfoaccess : tinyint(4) ▪ cntprofileaccess : tinyint(4) ▪ qltprofileaccess : tinyint(4) ▪ userid : varchar(4) ▪ username : varchar(30)	Nombre d'utilisateurs

▪ password : varchar(30) Clé primaire :uid	

20. La table «lastlogin »

La table «lastlogin» enregistre les dernières dates de connexion des utilisateurs.

Description	Tuples
Attributs : ▪ uid : smallint (4) ▪ date : datetime Clé primaire : uid	Nombre d'utilisateurs

21. la table « movie_genres»

La table « genre» enregistre les genres et le poids de chaque film. Ce poids est le même pour tous et est égal à 1.00000000.

Description	Tuples
Attributs : ▪ movieid : smallint (4) ▪ genre : varchar (40) ▪ weight : double (10,8) Clé primaire : movieid, genre	6407

22. la table «movies »

La table « movies » contient tous les films disponibles. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ movieid : smallint (4) ▪ titlemovielens: varchar (100) ▪ titleimdb : varchar (250) Clé primaire : movieid	3881

23. la table «newrecommendations »

Cette table stocke toutes les recommandations faites aux utilisateurs depuis leur dernière connexion. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ uid : smallint(4) ▪ movieid : smallint(4) ▪ critere : varchar(200) ▪ predictiveeval : double (11,8) ▪ date : datetime Clé primaire : uid, movieid, critere	Néant

24. la table «qlt_distance»

La table «qlt_distance » enregistre les distances entre les utilisateurs par rapprochement leurs évaluations. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ uid1 : smallint(4) ▪ uid2 : smallint(4) ▪ distance : double (11,8) ▪ userid1 : varchar(4) ▪ userid2 : varchar(4) Clé primaire : uid1, uid2	Néant

25. La table «qlt_partition »

La table « qlt_partition » enregistre la classification et les positions dans la carte 2D des utilisateurs par le critère évaluation. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ uid1 : smallint(4) ▪ x : smallint(4) ▪ y : smallint(4) ▪ cluser : smallint(4) ▪ userid1 : varchar(4) Clé primaire : uid1	Nombre d'utilisateurs

26. la table «rating »

La table « rating» enregistre les évaluations des utilisateurs. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ uid : smallint(4) ▪ movieid : smallint(4) ▪ ratingdate : datetime ▪ score : tinyint(2) ▪ userid : varchar(4) Clé primaire : uid,movieid	Néant

27. La table «refuser_partition »

La table «refuser_partition » enregistre les classifications des utilisateurs par les critères de recommandations. Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ period : varchar (6) ▪ uid : smallint(4) ▪ a1 : smallint(4) ▪ a2 : smallint(4) ▪ … ▪ a9 : smallint(4) Clé primaire : period, uid	Néant

28. la table «stats »

La table « stats » enregistre les statistiques sur les évaluations des utilisateurs. Cette table est remplie à partir de la table « rating ». Le tableau ci-dessous donne la description de la table.

Description	Tuples
Attributs : ▪ uid : smallint(4) ▪ genre : varchar(40) ▪ ratnb : smallint(4) ▪ ratsum : double (16,8) ▪ ratmean : double (16,8) ▪ ratvar double (16,8) Clé primaire : uid, genre	Néant

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Réalisation des tests système

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	20/08/2007
Version	1.0

Historique des versions

Version	Date	Modifications
1.0	20/08/2007	Début de rédaction

Sommaire

1. Introduction

Ce document a pour but de réaliser les scénarios définis dans le dossier plan tests système.
Ces scénarios permettront d'effectuer des tests systèmes et d'intégration des trois incréments.

2. Réalisation des tests – scénarios nominaux

Chaque scénario a pour but de tester une fonctionnalité bien précise du système. Ici, on détaille les actions à effectuer pour réaliser les scénarios, le résultat attendu après chaque action et le résultat obtenu après la réalisation du scénario.

Remarque : Se reporter au document de spécifications externes pour les différents espaces de travail.

2.1 Scénario : Rechercher un film

Acteur : Bob, utilisateur quelconque (membre ou internaute).
Contexte : il utilise un navigateur Web.

Action	Résultat attendu	Résultat obtenu
Bob entre le mot clé « white » dans la zone de recherche et clique sur le bouton « rechercher ».	Une liste de films dont les titres contiennent la chaîne « white » dans la zone de travail.	OK
Bob clique sur le film « black and white ».	La fiche du film s'affiche dans la zone de travail.	OK
Bob clique sur le bouton « retour » depuis la fiche du film « back and white ».	Les résultats de la recherche sont de nouveau affichés dans la zone de travail.	OK

2.2 Scénario : S'inscrire

Acteur : Bob, utilisateur internaute.
Contexte : il utilise un navigateur Web.

Action	Résultat attendu	Résultat obtenu
Bob clique sur le lien « inscrivez vous » dans la zone d'inscription.	Le formulaire d'inscription s'affiche dans la zone de travail.	OK
Bob remplit tous les champs du formulaire puis clique sur le bouton « valider ».	Le message « votre inscription s'est bien passée » s'affiche dans la zone de travail.	OK

2.2 Scénario : Consulter ses recommandations

Acteur : Bob, utilisateur membre.
Contexte : il utilise un navigateur Web.

Action	Résultat attendu	Résultat obtenu
Bob clique sur le lien « Nouvelles recommandations» dans la zone menu.	La liste des nouvelles recommandations s'affiche dans la zone de travail.	OK
Bob clique sur le lien « Toutes recommandations» dans la zone menu.	La liste des anciennes recommandations s'affiche dans la zone de travail.	OK
Bob change la note d'un film dans la liste de ces anciennes recommandations en choisissant la valeur 3 dans le menu déroulant.	Le nombre d'étoiles pour le film passe de 0 à 3. Le nombre d'anciennes recommandations diminue de 1, le nombre d'évaluations augmente de 1. Au rechargement des anciennes recommandations, le film doit avoir disparu de la liste, lorsque Bob consulte ses évaluations il doit retrouver le film avec 3 étoiles et la valeur du menu déroulant à 3.	OK
Bob clique sur la chexbox d'un film dans la liste de ces anciennes recommandations.	Le nombre de favoris augmente de 1. Lorsque Bob consulte ses favoris, il doit retrouver le film.	OK

2.3 Scénario : Consulter ses évaluations

Acteur : Bob, utilisateur membre.
Contexte : il utilise un navigateur Web.

Action	Résultat attendu	Résultat obtenu
Bob clique sur le lien « Evaluations» dans la zone menu.	La liste des évaluations s'affiche dans la zone de travail.	OK
Bob supprime un film qu'il n'aime plus de ses évaluations en choisissant dans le menu déroulant la valeur « Pas vu ».	Le nombre d'évaluations diminue de 1, le nombre d'étoiles du film passe à 0. Au rechargement des évaluations, le film doit avoir disparu de la liste.	OK

2.4 Scénario : Visualiser les communautés

Acteur : Bob, utilisateur membre.
Contexte : il utilise un navigateur Web.

Action	Résultat attendu	Résultat obtenu
Bob clique sur le lien « visualiser selon le contenu» dans la zone menu.	La carte de communauté s'affiche dans la zone de travail.	OK
Bob clique sur un utilisateur.	Le profil de l'utilisateur s'affiche dans la zone de travail.	OK
Bob clique le bouton « retour ».	La carte de communauté s'affiche dans la zone de travail.	OK

2.5 Scénario : Gérer son profil

Acteur : Bob, utilisateur internaute.
Contexte : il utilise un navigateur Web.

Action	Résultat attendu	Résultat obtenu
Bob clique sur le bouton « Gérer son profil » dans la gestion de profil utilisateur.	Le menu de gestion de profil utilisateur s'affiche dans la zone de travail.	OK
Bob clique sur le lien « Modifier ses données personnelles ».	Un formulaire pré-remplit avec les données personnelles de Bob s'affiche dans la zone de travail.	OK
Bob modifie son adresse et clique sur le bouton « valider ».	Le message « Vos modifications ont bien été prises ! » s'affiche dans la zone de travail.	OK
Bob clique de nouveau sur le bouton « Gérer son profil » dans la gestion de profil utilisateur.	Le menu de gestion de profil utilisateur s'affiche dans la zone de travail.	OK
Bob clique sur le lien « Visualiser son vecteur de positionnement ».	Les 5 communautés de Bob s'affichent dans la zone de travail.	OK
Bob remarque que sa communauté selon le critère localisation a changé dû à la modification de son adresse.	Néant	OK
Bob clique sur le lien vers sa communauté selon le critère âge.	Les informations sur la communauté s'affichent dans la zone de travail.	OK
Bob clique sur le bouton « Retour ».	Les 5 communautés de Bob s'affichent dans la zone de travail.	OK

2.6 Scénario : Agir sur un film

Acteur : Bob, utilisateur internaute.
Contexte : il utilise un navigateur Web.

Action	Résultat attendu	Résultat obtenu
Bob entre son nom d'utilisateur dans le champ « Nom d'utilisateur », entre son mot de passe dans le champ « Mot de passe » puis clique sur le bouton « Se connecter ».	Un message de bienvenue, la date de la dernière connexion, le bouton « Gérer son profil » et le bouton « Se déconnecter » s'affiche dans la zone gestion du profil utilisateur. Les fonctionnalités accessibles au membre s'affichent dans la zone menu.	OK
Bob clique sur le lien « Favoris » dans la zone menu.	Les favoris s'affichent dans la zone de travail.	OK
Bob clique sur un film de la liste des favoris.	La fiche du film s'affiche dans la zone de travail.	OK
Bob clique sur le bouton « Recommander le film à un contact ».	La liste des contacts de Bob s'affiche dans un menu déroulant, un bouton ok ; le bouton « Recommander le film à un contact » disparaît	OK
Bob choisit un contact dans le menu déroulant puis clique sur le bouton « ok ».	Un message indiquant si le film a pu être au contact ou non apparaît.	OK
Bob clique sur le bouton « retour ».	Les favoris s'affichent dans la zone de travail.	OK
Bob clique sur l'image de la poubelle d'un film.	Une alerte demande à Bob de confirmer s'il souhaite bien supprimer le film.	OK
Bob confirme en cliquant sur le bouton « ok ».	Le film disparaît des favoris.	OK

2.6 Scénario : Gérer son carnet d'adresses

Acteur : Bob, utilisateur membre.
Contexte : Il utilise un navigateur web.

Action	Résultat attendu	Résultat obtenu
Bob clique sur le lien « Toutes recommandations» dans la zone menu.	La liste des anciennes recommandations s'affiche dans la zone de travail.	OK
Bob parcourt la liste de ses anciennes recommandions et tombe sur un film recommandé par un contact, il n'a pas aimé ce film donc il décide de supprimer le contact. Il clique sur le lien « Consulter son carnet d'adresses »	Le carnet d'adresses de Bob s'affiche dans la zone de travail.	OK
Bob entre le nom du contact à supprimer dans le champ « Supprimer le contact ».	Une alerte demande à Bob de confirmer la suppression du contact.	OK
Bon confirme la suppression.	Le contact disparaît.	OK
Bob souhaite maintenant ajouter son ami dans son carnet d'adresse. Il entre le mot « Amis » dans le champ « Ajouter le groupe » puis clique sur le bouton « OK».	Le groupe « Amis » apparaît dans le carnet d'adresses.	OK
Bob entre le nom de son ami dans le champ « Ajouter le contact » puis clique sur le bouton « OK ».	Le contact apparaît dans le carnet d'adresses.	OK
Bob clique sur le menu déroulant et choisit le groupe « Amis ».	Le contact est déplacé vers le groupe « Amis ».	OK

3. Réalisation des tests – scénarios D'exception

Les scenarios ont pour but de tester de prouver que les erreurs sont bien gérées dans l'application COCoFil3. Ici, on détaille les actions à effectuer pour réaliser les scénarios, le résultat attendu après chaque action et le résultat obtenu après la réalisation du scénario.

Remarque : Se reporter au document de spécifications externes pour les différents espaces de travail.

3.1 Scénario : Formulaire mal rempli

Acteur : Bob, utilisateur internaute.
Contexte : Il utilise un navigateur web.

Action	Résultat attendu	Résultat obtenu
Bob clique sur le lien «Inscrivez vous» dans la zone d'inscription.	Le formulaire d'inscription s'affiche dans la zone de travail.	OK
Bob remplit le formulaire on omettant de fournir son code postal et clique sur le bouton « Valider ».	Un message d'erreur au dessus du formulaire indique à Bob qu'il doit fournir son code postal. Le formulaire est affiché dans la zone de travail avec les informations déjà fournies par Bob.	OK
Bob entre son code postal dans le champ « Code postal » et clique de nouveau sur le bouton « Valider».	Le message « votre inscription s'est bien passée » s'affiche dans la zone de travail.	OK

3.2 Scénario : Erreurs à l'identification

Acteur : Bob, utilisateur internaute.
Contexte : Il utilise un navigateur web.

Action	Résultat attendu	Résultat obtenu
Bob entre son nom d'utilisateur dans le champ « Nom d'utilisateur » et clique sur le bouton « Se connecter ».	Un message d'erreur indiquant à Bob que le champ « Mot de passe » doit être renseigné s'affiche dans la zone d'identification.	OK
Bob saisit un mot de passe erroné dans le champ « Mot de passe » et clique sur le bouton « Se connecter ».	Un message d'erreur indiquant à Bob que le mot de passe est erroné s'affiche dans la zone d'identification.	OK
Bob saisit son mot de passe dans le champ « Mot de passe » et clique sur le bouton « Se connecter ».	Un message de bienvenue, la date de la dernière connexion, le bouton « Gérer son profil » et le bouton « Se déconnecter » s'affiche dans la zone gestion du profil utilisateur. Les fonctionnalités accessibles au membre s'affichent dans la zone menu.	OK

3.3 Scénario : Recommandation d'un film non évalué

Acteur : Bob, utilisateur membre.
Contexte : Il utilise un navigateur web.

Action	Résultat attendu	Résultat obtenu
Bob est entrain de lire la fiche d'un film se trouvant par ses nouvelles recommandations. Il clique sur le bouton « Recommander ce film à un contact ».	Une alerte avertit Bob qu'il doit d'abord évaluer le film.	OK
Bob choisit la valeur 3 dans le menu déroulant.	Le nombre d'étoiles passe à trois, le nombre de nouvelles recommandations diminue de 1 le nombre d'évaluations augment de 1.	OK
Bob clique sur le bouton « Recommander ce film à un contact ».	Le bouton « Recommander ce film à un contact » disparaît, la liste des contacts de Bob apparaît dans un menu déroulant, un bouton « ok » apparaît.	OK
Bob choisit un contact dans le menu déroulant puis clique sur le bouton « ok ».	Un message indique à Bob que le contact a déjà évalué le film.	OK
Bob choisit un autre contact dans le menu déroulant puis clique sur le bouton « ok ».	Un message indique à Bob que le film a bien été recommandé au contact.	OK

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Réalisation des tests d'acceptation

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	20/08/2007
Version	1.0

Historique des versions

Version	Date	Modifications
1.0	20/08/2007	Début de rédaction

Sommaire

1. Introduction

Ce document a pour but de réaliser les scénarios d'acceptation définis dans le dossier plan tests d'acceptation afin de valider le logiciel COCoFil3.

2. Réalisation des tests

Les scénarios ci-dessous sont réalisés par le maître d'ouvrage, DENOS Nathalie.
La réalisation de ces tests nécessite un ordinateur équipé d'un navigateur web.

2.1 Scénario « Découverte

Fonctionnalité	Résultat des tests
S'identifier	
Consulter les recommandations	
Evaluer un film	
Ajouter le film à sa liste des favoris	
Consulter sa liste des favoris	
Supprimer le film de sa liste des favoris	

2.2 Scénario « Découverte »

Fonctionnalité	Résultats des tests
Rechercher un film	
Fournir ses données personnelles	
Définir les paramètres par défaut	

Consulter ces recommandations	

2.3 Scénario « communautés »

Fonctionnalité	Résultats des tests
S'identifier	
Visualiser la carte selon le critère « contenu »	
Visualiser la carte selon le critère « qualité »	
Visualiser le profil d'un utilisateur	
Visualiser son vecteur de positionnement	
Se déconnecter	

2.4 Scénario « Bouche à oreille »

Fonctionnalité	Résultats des tests
S'inscrire	
S'identifier	
Ajouter un contact	
Recommander un film à un contact	

3. Bilan

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Bilan qualité

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	23/08/2007
Version	1.0

Historique des versions

Version	Date	Modifications
1.0	23/08/2007	Début de la rédaction

Sommaire

1. Introduction

But et portée du document

Ce document a pour but d'établir un compte rendu des mesures des facteurs et critères de qualité et d'analyser les résultats.

Le présent document fait référence aux clauses qualités définies dans le Plan d'Assurance Qualité Logicielle.

1.2 A propos du projet

Nous rappelons que le projet COCoFil3 est une application Web pour un système de recommandation de films grâce au filtrage collaboratif basé sur les communautés d'utilisateurs.

2. Mesure de l'utilisabilité

La mesure de l'utilisabilité se fait à l'aide d'un questionnaire soumis à des utilisateurs. Dans ce questionnaire, il est demandé à l'utilisateur d'exécuter 14 scénarios couvrants la totalité des fonctionnalités du système. A chaque scénario, l'utilisateur doit répondre au questionnaire suivant :

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Les réponses permettent ainsi de vérifier que l'utilisateur a bien accompli le scénario, de savoir si il l'a réalisé en un nombre de clics minimum, si l'action est facilement réalisable et enfin les difficultés rencontrées et les remarques (notamment sur le temps d'exécution).
On attribue à chaque réponse à un scénario une note sur 5 suivant le barème ci-dessous :

- 1 point si le scénario a été accompli, 0 sinon
- 1 point si il a été accompli en un nombre de clics minimums à 1 près, 0 sinon
- 0 point si très dur, 1 si dur, 2 si moyen et 3 points si facile

La moyenne des notes sur 5 des 14 scénarios et ensuite ramener sur 10 pour donner une note à la globalité du questionnaire. Ces notes nous permettrons d'évaluer le critère d'utilisabilité et de communicativité. Les observations et les remarques des utilisateurs nous servirons à estimer si le critère de communicativité est bien respecté.
On a en tout questionné quatre utilisateurs, ce qui nous a parue suffisant étant donné que les résultats et les remarques étaient relativement similaires d'un utilisateur à un autre.

Questionnaire du premier utilisateur :

Scénario n°	Action accomplie	Nombre de clics requis	Nombre de clics de l'utilisateur	Niveau de difficulté	Difficultés rencontrées, remarques	Note sur 5
1	oui	2	2	simple		5
2	oui	2	3	simple		5
3	oui	2	2	moyen	Indiquer que le nom d'utilisateur correspond à l'email. Temps de connexion long	4
4	oui	2	2	simple		5
5	oui	1	1	simple		5
6	oui	2	3	très dur	Manque d'explications	2
7	oui	1	1	moyen	Manque d'explications	4
8	oui	2	2	simple		5
9	oui	2	2	simple		5
10	oui	3	3	simple		5
11	oui	5	6	moyen	Bloquer des contacts pas très clair	4
12	oui	3	3	simple		5
13	non	2				0
14	oui	4		simple		5

Questionnaire du deuxième utilisateur :

Scénario n°	Action accomplie	Nombre de clics requis	Nombre de clics de l'utilisateur	Niveau de difficulté	Difficultés rencontrées, remarques	Note sur 5
1	oui	2	2	simple		5
2	oui	2	2	simple		5
3	oui	2	2	moyen	Pseudo=email Long	4
4	oui	2	2	simple		5
5	oui	1	1	simple	Pas compris à quoi correspondent les critères de recommandations	5
6	oui	2	2	moyen		4

7	oui	1	1	simple		5
8	oui	2	2	moyen	Lent	4
9	oui	2	2	simple		5
10	oui	3	3	simple		5
11	oui	5	5	simple		5
12	oui	3	3	simple		5
13	oui	2	3	moyen	Pas facile à trouver	4
14	oui	4	5	simple		5

Questionnaire du troisième utilisateur :

Scénario n°	Action accomplie	Nombre de clics requis	Nombre de clics de l'utilisateur	Niveau de difficulté	Difficultés rencontrées, remarques	Note sur 5
1	oui	2	2	simple		5
2	oui	2	2	simple		5
3	oui	2	2	simple	Pseudo=email Un peu long	5
4	oui	2	3	simple		5
5	oui	1	2	simple	Pas compris à quoi correspondent les critères de recommandations	5
6	oui	2	1	dur		3
7	oui	1	2	simple		5
8	oui	2	2	moyen	Légender la carte pour faciliter la compréhension	4
9	oui	2	2	simple		5
10	oui	3	3	simple		5
11	oui	5	6	simple		5
12	oui	3	3	simple		5
13	oui	2	3	simple		5
14	oui	4	5	simple		5

Questionnaire du quatrième utilisateur :

Scénario n°	Action accomplie	Nombre de clics requis	Nombre de clics de l'utilisateur	Niveau de difficulté	Difficultés rencontrées, remarques	Note sur 5
1	oui	2	2	simple		5
2	oui	2	2	simple		5
3	oui	2	2	moyen	Pseudo=email Long	4
4	oui	2	2	simple		5

5	oui	1	1	simple		5
6	oui	2	2	moyen		4
7	oui	1	1	simple		5
8	oui	2	2	moyen	Lent	4
9	oui	2	2	simple		5
10	oui	3	3	simple		5
11	oui	5	5	simple		5
12	oui	3	3	simple		5
13	oui	2	3	moyen	Pas facile à trouver	4
14	oui	4	5	moyen		4

On a donc les notes suivantes :

- Utilisateur 1 : 4,2 sur 5
- Utilisateur 2 : 4,7 sur 5
- Utilisateur 3 : 4,8 sur 5
- Utilisateur 4 : 4,6 sur 5

2.1 Facilité d'apprentissage et d'utilisation

En ramenant sur 10 les notes obtenues aux questionnaires soumis aux utilisateurs et en faisant la moyenne de ces notes on obtient (8.4 + 9,4 + 9,6 + 9,2) / 4 = 9,15. Cette note sert également à évaluer la communicativité.

2.2 Bonne compréhension des entrées/sorties

La compréhension des critères de recommandations reste floue pour l'utilisateur. De ce fait, les critères des communautés ne sont pas compris immédiatement par l'utilisateur, il a donc des difficultés à percevoir l'intérêt des cartes. La notion de critères sur lesquels on forme les communautés doit mieux être expliquée à travers le système afin que l'utilisateur comprenne mieux la notion de communauté.

2.3 Performance (temps de réponse)

Les utilisateurs ne sont pas gênés par les temps de réponse du système à l'exception de la connexion et de l'affichage du profil des communautés. Pour l'affichage du profil des communautés, la lenteur est due au nombre important de données et d'utilisateurs mis en jeu. Des mesures seront prises pour améliorer le temps de réponse en optimisant la requête réalisant le profil d'une communauté ou en changeant les informations choisit pour caractériser une communauté. Concernant la connexion, le temps trop important est due au fait que le système calcule les nouvelles recommandations pour l'utilisateur suivant le critère âge, profession, localisation et genre. Pour remédier à ce problème, le calcul devra se faire en temps masqué, la création d'un thread pour chaque calcul de recommandations permettra donc à l'utilisateur d'accéder directement à sa session. Les recommandations qui lui seront faites arriveront au fur et à mesure de sa session.
Pour ce critère, on procède à une estimation. La note attribuée par le responsable qualité en fonction des temps constatés et des remarques des utilisateurs est de 8.

2.4 Bilan pour l'utilisabilité

On a obtenu une note de 9.15 pour l'utilisabilité et la communicativité et de 8 pour la performance. Ce qui nous fait une moyenne de 8.8

En appliquant la formule suivante :

$$CoefficientSatisfaisabilité = 100 * \frac{MesureObtenue}{NoteImportance}$$

On obtient un coefficient de satisfaction de 100 * 8.8 / 9.5 = 92.6% ce qui est supérieur à 90%. L'objectif est donc atteint pour le facteur utilisabilité. Néanmoins des modifications restent à réaliser pour améliorer la compréhension de certains aspects du système liés aux recommandations et aux communautés. De plus certains temps de réponse sont trop importants pour que le confort d'utilisation soit optimum.

3. Mesure de la maintenabilité

Les mesures pour le facteur de maintenabilité se font à l'aide de l'outil Metrics. Metrics est un plug-in Eclipse qui produit de nombreuses métriques sur un projet (nombres de classes, de méthodes, de lignes de code, …). Les mesures ne prennent en compte que les sources Java du projet, elles ne considèrent pas les pages JSF ou les fichiers de configurations dans les calculs.

Mesure de la modularité

Un couplage supérieur à 5 est considérée comme trop élevée. La figure 5 montre les valeurs de couplage de notre projet.

Metric	Total	Mean	Std. Dev.	Maximum
⊟ Afferent Coupling (avg/max per packageFragment)		4,556	7,395	25
COMMUNITY	25			
GRE	5			
GIU	4			
CVC	3			
GF	1			
GRE.RECOMMENDATIONS	1			
GRE.RECOMMENDATIONS_EVALUATION	1			
GRE.RECOMMENDATIONS_GENRE	1			

Figure 1 : Couplage du projet

On a donc le prédicat suivant :

$$PrédicatCouplage = Couplage < 5$$

On calcul ensuite la note de modularité :

$$NoteModularité = 10 * \frac{NombreDePackagesVérifiantPrédicatCouplage}{NombreDePackagesTotal}$$

On a donc un seul package sur les 8 qui a un couplage supérieur à 5.
La modularité est donc de 10 * 7 / 8, ce qui nous fait une note de 8.75. Pour valider ce critère, la note obtenue devait être supérieur à 8.5. La qualité de ce critère est donc respectée.

3.2 Mesure de la simplicité

Afin d'obtenir un code simplement compréhensible, on fixe la taille maximum d'une classe à 500 lignes et la taille maximum d'une méthode à 150 lignes. Pour mesurer ce facteur, on calcul donc deux notes : une pour les classes et une pour les méthodes.

3.2.1 Note pour les classes

On a défini le prédicat suivant :

$$PrédicatClasse = TailleClasse < 500$$

et la formule :

$$NoteClasses = 10 * \frac{NombreDeClassesVérifiantPrédicatClasse}{NombreDeClassesTotal}$$

Comme le montre la figure 2, il y a 73 classes au total dans le projet. La figure 3 nous montre qu'il y a 6 classes qui font plus de 500 lignes (les classes marquées en rouge), donc on a 67 classes qui respectent le prédicat de classe.

La note de classe est donc de 10 * 67 / 73, ce qui nous fait une note de 9.2.

⊟ Number of Classes (avg/max per packageFragment)	73
⊟ src	72
⊞ CVC	21
⊞ COMMUNITY	16
⊞ GIU	9
⊞ Tests	7
⊞ GRE	5
⊞ GRE.RECOMMENDATIONS	5
⊞ GF	4
⊞ GRE.RECOMMENDATIONS_GENRE	3
⊞ GRE.RECOMMENDATIONS_EVALUATION	2

Figure 2 : Nombre de classes

Total Lines of Code	16218
src	16165
COMMUNITY	5124
Filtering.java	1199
RoughSets.java	1037
DB_Manager.java	640
QualityCF.java	487
ContentCF.java	479
RoughSets_Expr.java	447
Filtering_Expr.java	332
Sets.java	135
Filtering_Parm.java	120
Lib.java	104
RoughSets_Parm.java	42
Item.java	31
RoughSets_Test.java	23
Filtering_Test.java	23
Agent.java	14
Cell.java	11
CVC	4051
AntBasedClustering.java	778
SVG2.java	703
SVG.java	481
HierarchicalClustering.java	299
DB_Manager.java	296
ContentDistance.java	274
kMeans.java	271
QualityDistance.java	191
UserProfile.java	174
Sets.java	134
CarteContenu.java	110
CarteQualite.java	101
CommunityProfile.java	82
Cartes.java	70
Lib.java	42
DemonContent.java	21
Agent.java	13
Item.java	11

GIU	1812
UserBean.java	470
AddressBook.java	455
ProfilBean.java	272
InfoCommunaute.java	235
FavorisBean.java	207
Cmnt.java	87
Profil.java	46
Favori.java	40
GRE.RECOMMENDATIONS	1482
getRecommendations_a1.java	314
getRecommendations_a2.java	307
getRecommendations_a1Optimise.java	293
getRecommendations_a2Optimise.java	292
getRecommendations_a3.java	276
Tests	1098
test_genre.java	244
test_evaluation.java	233
test_localisation.java	195
test_age.java	195
test_profession.java	195
test_carte_contenu.java	18
test_carte_qualité.java	18
GRE	831
GREBean.java	550
FABean.java	154
Recommendation.java	72
Rating.java	39
affichage.java	16
GRE.RECOMMENDATIONS_GENRE	733
getRecommendations_a5.java	398
dataPreparation.java	290
Test.java	45
GF	691
Movie.java	356
Search.java	246
SortFilterModel.java	89
GRE.RECOMMENDATIONS_EVALUATION	343
getRecommendations_a6.java	329
Test.java	14

Figure 3 : Nombre de lignes de code

3.2.2 Note pour les méthodes

On définit une méthode comme valide si elle respecte la formule suivante :

$$\text{PrédicatMéthode} = \begin{array}{c}\text{NormesDeProgrammationRespectées} \\ \& \\ \text{TailleMéthode} < 150\end{array}$$

On calcule alors la note pour les méthodes par la formule suivante :

$$\text{NoteMéthodes} = 10 * \frac{\text{NombreDeMéthodesVérifiantPrédicatMéthode}}{\text{NombreDeMéthodesTotal}}$$

La figure 4 montre qu'il y a 679 méthodes.

Metric	Total	Mean	Std. Dev.	Maximum
Number of Methods (avg/max per type)	679	9,301	9,117	40
src	677	9,403	9,139	40
GRE	88	17,6	12,225	40
GIU	161	17,889	11,12	38
GF	67	16,75	11,344	33
COMMUNITY	116	7,25	8,54	27
CVC	125	5,952	5,214	18
Tests	71	10,143	3,907	13
GRE.RECOMMENDATIONS	30	6	0,632	7
GRE.RECOMMENDATIONS_GENRE	13	4,333	3,091	7
GRE.RECOMMENDATIONS_EVALUATION	6	3	3	6

Figure 4 : Nombre de méthodes

Les figures 5.1 et 5.2 montre qu'il y a 8 méthodes qui mesurent plus de 150 lignes.
On a donc une note de 10 * 671 / 679 = 9.9.

Metric	Total	Mean	Std. Dev.	Maximum
⊟ Method Lines of Code (avg/max per method)	13619	19,681	35,348	433
⊟ src	13579	19,68	35,384	433
⊟ COMMUNITY	4504	38,169	56,516	433
⊟ RoughSets_Expr.java	433	216,5	216,5	433
⊟ RoughSets_Expr	433	216,5	216,5	433
rstExperimentation	433			
RoughSets_Expr	0			
⊟ Filtering_Expr.java	321	160,5	160,5	321
⊟ Filtering_Expr	321	160,5	160,5	321
cfExperimentation	321			
Filtering_Expr	0			
⊟ Filtering.java	1119	74,6	45,528	163
⊟ Filtering	1119	74,6	45,528	163
genreCF	163			
attributeCF	155			
pearsonCF	127			
qualityCF	98			
hybridFiltering	90			
communityEvaluationProfile	82			
communityEvaluationHistogram	70			
recommendationCorrelation	69			
recommendationMAE	68			
recommendationSimilarity	66			
topRecommendations	42			
recommendationEvaluation	39			
extractPreferredGenreRatings	36			
getCommunityWidth	14			
Filtering	0			
⊞ ContentCF.java	440	55	39,978	117
⊞ RoughSets.java	921	34,111	30,433	110
⊞ Filtering_Parm.java	76	76	0	76
⊞ QualityCF.java	444	49,333	23,547	75
⊞ DB_Manager.java	542	20,846	11,051	47
⊞ Sets.java	89	6,846	7,979	32
⊞ RoughSets_Parm.java	21	21	0	21
⊞ Lib.java	74	10,571	6,091	21
⊞ RoughSets_Test.java	12	6	5	11
⊞ Filtering_Test.java	12	6	5	11
⊞ Agent.java	0	0	0	0
⊞ Item.java	0	0	0	0
⊞ Cell.java	0	0	0	0

Figure 5.1 : Nombre de lignes de code par méthode

Metric	Total	Mean	Std. Dev.	Maximum
CVC	3496	26,09	40,481	253
SVG2.java	667	111,167	80,352	253
SVG2	667	111,167	80,352	253
svgFileHeader	253			
svgFileCreate	160			
couleur	121			
userGen	82			
htmlFileCreate	43			
SVG2	8			
HierarchicalClustering.java	270	45	53,107	161
HierarchicalClustering	270	45	53,107	161
haClustering	161			
nearestClsts	37			
completeLink	29			
singleLink	29			
HierarchicalClustering	8			
intraInner2D	6			
AntBasedClustering.java	705	44,062	44,635	140
ContentDistance.java	260	86,667	31,668	131
SVG.java	432	54	43,852	121
CarteContenu.java	87	21,75	28,446	70
kMeans.java	245	35	18,346	62
CarteQualite.java	78	19,5	24,047	60
QualityDistance.java	173	43,25	17,81	59
DB_Manager.java	249	22,636	13,117	47
UserProfile.java	110	4,4	7,673	33
Sets.java	89	6,846	7,979	32
CommunityProfile.java	54	6	9,104	24
Lib.java	31	10,333	8,576	21
DemonContent.java	15	15	0	15
Cartes.java	31	2,583	3,593	12
Item.java	0	0	0	0
Agent.java	0	0	0	0
GRE.RECOMMENDATIONS_GENRE	646	46,143	44,998	165
getRecommendations_a5.java	370	52,857	52,824	165
getRecommendations_a5	370	52,857	52,824	165
genreCF	165			
periodGenreCommunity	92			
getRecommendationsForId	40			
ToNewRecommendations	28			
getGenre	24			
getCommunityForId	21			
getRecommendations_a5	0			
dataPreparation.java	267	44,5	34,408	98
Test.java	9	9	0	9
GRE.RECOMMENDATIONS_EVALUATION	304	43,429	38,851	107
GRE.RECOMMENDATIONS	1349	44,967	26,831	100
GIU	1308	8,124	16,001	86
GF	476	7,104	13,634	75
GRE	583	6,625	14,956	67
Tests	913	12,859	7,851	21

Figure 5.2 : Nombre de lignes de code par méthode

3.2.2 Bilan pour la simplicité

On a une note de 9.2 pour les classes et 9.9 pour les méthodes. La note de simplicité est donc de 9.55. Pour valider ce critère, la note obtenue devait être supérieur à 8.5. La qualité de ce critère est donc respectée.

3.3 Bilan pour la maintenabilité

On a obtenu une note de 8.75 pour la modularité et une note de 9.55 pour la simplicité.
La moyenne des notes des critères de maintenabilité est donc de (8.75 + 9.55) / 2 = 9.15.

En appliquant la formule suivante :

$$\text{CoefficientSatisfaisabilité} = 100 * \frac{\text{MesureObtenue}}{\text{NoteImportance}}$$

On obtient un coefficient de satisfaction de 100 * 9.15 / 9 = 101% ce qui est supérieur à 90%. L'objectif est donc atteint pour le facteur maintenabilité.

4. Conclusion

Les deux facteurs de qualité, l'utilisabilité et la maintenabilité, sont respectés avec des notes de satisfactions supérieures à 90%. Néanmoins, les résultats obtenus lors de la soumission de scénarios et de questionnaires aux utilisateurs a nécessité des actions correctives afin de pallier aux problèmes rencontrés. La compréhension des aspects liés au filtrage basé sur les communautés reste floue pour l'utilisateur, principalement les critères de formation des communautés ainsi que la difficulté à interpréter les cartes représentant les communautés.

Laboratoire LIG
Equipe MRIM
BP 53, 38041 Cedex 9
Grenoble

COCoFil3

Community-Oriented Collaborative Filtering

Dossier d'évaluation

Auteurs	ARGOUD Guillaume CAMARA Fatoumata Goundo
Responsables	BERRUT Catherine DENOS Nathalie
Date	21/05/2007
Version	1.0

Historique des versions

Version	Date	Modifications
1.0	21/05/2007	Début de la rédaction

Sommaire

1. Introduction

But et portée du document

Ce document a pour but d'analyser les critères de facilité d'apprentissage, d'utilisation et de communicativité. En effet, ce document est destiné « aux testeurs » du système COCoFil3. Il a pour but de parcourir les principales fonctionnalités du système COCoFil3 pour s'assurer que l'application est facile d'apprentissage et d'utilisation et que les données d'entrées sont faciles à produire et les données de sorties sont facilement interprétables.

Le présent document fait référence au Dossier de Spécifications Externes du projet et est rédigé en fonction des clauses qualités définies dans le Plan d'Assurance Qualité Logicielle.

1.2 A propos du projet

Nous rappelons que le projet COCoFil3 est une application Web pour un système de recommandation de films grâce au filtrage collaboratif basé sur les communautés d'utilisateurs.

2. Jeux de tests à effectuer

Il est important de savoir que le test que vous devez effectuer est d'une grande importance : toutes vos remarques seront prises en compte avec soin. C'est pourquoi nous vous demandons de noter ce qui vous paraît curieux et non intuitif (par exemple la formulation d'un bouton).

Recherche et détails d'un film

Entrer l'adresse suivante dans la barre d'adresses : http://localhost:8080/COCoFil3
Vous accédez à l'application Web COCoFil3. Tant que vous n'êtes pas inscrit et identifié, vous pouvez uniquement rechercher un film et consulter les détails d'un film (ainsi que vous inscrire et vous identifier).

Commençons par consulter la fiche du film « Back to the Future » :

- Entrer le mot « back » dans la zone de recherche et validé.
- Sélectionner le film « Back to the Future » dans la liste affichée.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Inscription

Nous allons à présent créer un compte afin d'accéder à toutes les fonctionnalités du système. Pour se faire, aller au bon endroit et donner les informations personnelles nécessaires à la création du compte.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Identification

Commencer par rentrer votre login accompagné d'un mot de passe erroné.
Rentrer ensuite votre login et le mot de passe correcte.
Observer les nouvelles fonctionnalités disponibles.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Les tests suivants sont à effectuer en restant identifié. Si pour une raison ou pour une autre vous venez à être déconnecté, vous devrez vous identifier de nouveau avant de poursuivre.

Evaluation

Chercher un film que vous avez déjà vu.
Attribuer une note au film.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Recommandations

Consulter, si il en existe, les recommandations qui vous ont été faites.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Visualisation des communautés

Consulter la carte des communautés suivant le critère "contenu".
Localiser votre position parmi tous les utilisateurs.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Visualisation du profil d'un utilisateur

Consulter le profil d'un utilisateur à partir de la carte.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Visualisation du profil d'une communauté

Toujours à partir de la carte, visualiser le profil d'une communauté.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Ajouter un film aux favoris

Chercher un film que vous aimez ou que vous aimeriez voir.
Ajouter ce film aux favoris.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Consulter les favoris

Consulter la liste de vos favoris.
Supprimer de la liste de vos favoris le film précédemment ajouté.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Carnet d'adresses

Ajouter le contact "toto@truc.fr" et le contact "tata@truc.fr" à votre carnet d'adresse.
Créer le groupe "Collègues" et y déplacer le contact "tata@truc.fr".
Bloquer le contact "toto@truc.fr".

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Données personnelles

Vous devez modifier vos données personnelles afin de changer votre mot de passe.

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Profil utilisateur

Consultez votre vecteur de positionnement dans les communautés suivant le critère "Genre".

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

Recommandation active

Vous devez recommander votre film préféré à "toto@truc.fr".

Action accomplie	Nombre de clics requis	Niveau de difficulté	Difficultés rencontrées, remarques, …
☐ Oui ☐ Non		☐ Très dur ☐ Dur ☐ Moyen ☐ Simple	

3. Vue d'ensemble

Toutes les remarques sur la globalité du système sont les bienvenues.
Les remarques suivantes visent à établir les éventuels problèmes généraux rencontrer par les utilisateurs.

Que pensez-vous du système ?

La charge d'information à l'écran est-elle correcte
Si non, insuffisante / trop importante ?

La compréhension des fonctionnalités (recherche, évaluer un film, visualiser les cartes, etc.) est-elle immédiate ?

Les cartes sont-elles facilement interprétables ?

Discerne ton facilement

- les différentes communautés

- la signification de l'utilisateur représentatif

- la localisation de sa position