Projet Expensive-Tracker

Antoine Drouhin
Aurélien Garret

Master 1 MIAGE en apprentissage, promotion 2016/2017
Université Paris 1 Panthéon-Sorbonne

Dossier d'analyse

Version 1.0 - Date : 15/02/2017

Sommaire

Introduction

Le projet consiste à développer une application de gestion de budget. Il apparaît que dans le passé il existait un logiciel Microsoft Money qui permettait la gestion d'un ou plusieurs budgets. Depuis l'arrêt de ce produit, aucun acteur majeur a réussi à se détacher sur ce marché.

D'autre part, le besoin auquel répondait Microsoft Money tend à se modifier avec l'usage et l'habitude des nouveaux consommateurs. La modernité de l'idée de ce projet consisterait à intégrer différentes fonctionnalités tel que la gestion de budget collaborative, une analyse des revenus/dépenses afin de mieux allouer son capital.

Cette application de gestion de Budget se positionne entre les applications dédiées des banques, des applications de saisie tierces et isolées des banques et des systèmes plus anciens comme Money.

Aucun échange avec les banques n'est prévue. Il s'agit d'une saisie anonyme d'une dépense ou d'un gain avec un typage associée et d'un outil de visualisation de l'historique. Une seconde partie du projet serait de rendre l'application collaborative en permettant un usage des fonctionnalités inter-utilisateur.

Qualités attendues

Implémentation

Identifiant	Qualité	Description
IMPL1	Gestion des comptes utilisateurs	Le produit doit pouvoir gérer la création de compte utilisateur, ainsi que la possibilité d'un utilisateur donné d'accéder à une interface connectée.
IMPL2	Gestion des feuilles de comptes	Le produit doit pouvoir proposer à l'utilisateur la création, la suppression et l'utilisation de feuille de compte pour qu'il parvienne à saisir ses opérations et à en visualiser l'historique. Une partie optionnelle consisterait aussi à pouvoir bénéficier de feuille de compte collaborative avec d'autres utilisateurs.
IMPL3	Gestion des portefeuilles	Le produit doit pouvoir agréger un ensemble de feuille de compte sous l'appellation de portefeuille. On y retrouvera le solde global qui résulte du solde de chaque feuille de compte.

IMPL4	Gestion multilangue de l'application	Le produit devra supporter à la fois l'anglais et le français.
IMPL5	Gestion de la sécurité	Les données utilisateurs doivent être sécurisée pour garantir leur confidence et leur consistance.
IMPL6	Gestion de la mobilité	Le produit doit supporter les terminaux mobiles comme les ordinateurs de bureau. L'interface doit s'adapter aux diverses résolutions et connexions.

Evolutivité

Identifiant	Qualité	Description
EVOL1	Découplage des éléments de l'application	Les éléments développés composant les fonctionnalités du logiciel devront répondre à un couplage faible notamment par l'utilisation de composants pour permettre l'évolutivité du logiciel.
EVOL2	Ajout de nouvelles langues et séparation des ressources de langues	Les traductions linguistiques seront séparés des éléments d'interface ou de logique.

Maintenance

Identifiant	Qualité	Description
MAINT1	Démarche composant des éléments d'interface et de logique	Faciliter l'isolation pour détecter plus facilement les anomalies et optimiser le cadre des corrections (tests unitaires notamment)
MAINT2	Isolation des ressources	Ressources séparées du code pour faciliter leur mise à jour ou leur remplacement sans perturber ou altérer toute ou partie du logiciel.

Interopérabilité

Identifiant	Qualité	Description
INTER1	API selon la philosophie REST	Uniformisation des points d'entrées selon REST pour que l'API puisse éventuellement s'ouvrir à d'autres collaborations ou systèmes.
INTER2	Technologie de développement permettant le basculement entre plateforme	Pouvoir porter le code vers une autre plateforme sans avoir à refondre le code métier ou d'interface.

Terminologie

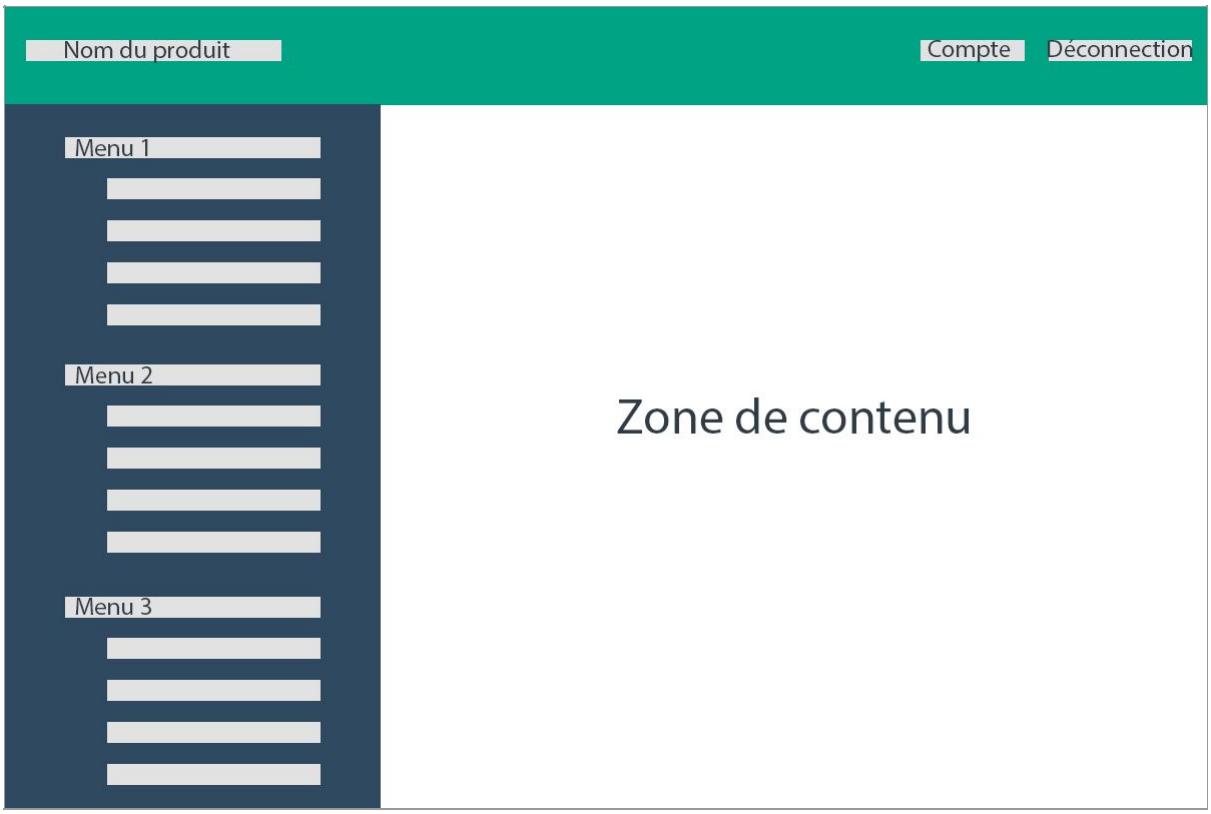
Terme	Définition
Portefeuille	Le portefeuille désigne un agrégat de feuilles de compte et représente un solde qui résulte de chacun des soldes des feuilles des comptes.
Utilisateur	Un utilisateur est la personne qui utilise notre application et ses fonctionnalités.
Feuille de compte	Une feuille de compte est un outil qui permet d'y ajouter ses opérations (entrées ou sorties) et d'en calculer le solde. C'est aussi l'historique des opérations saisies en son sein.

Maquette

Les maquettes proposées ici visent à éclaircir l'idée générale de l'interface qu'il serait souhaitable de mettre en place mais ne créditent en rien l'aspect final du produit.

Version ordinateur de bureau

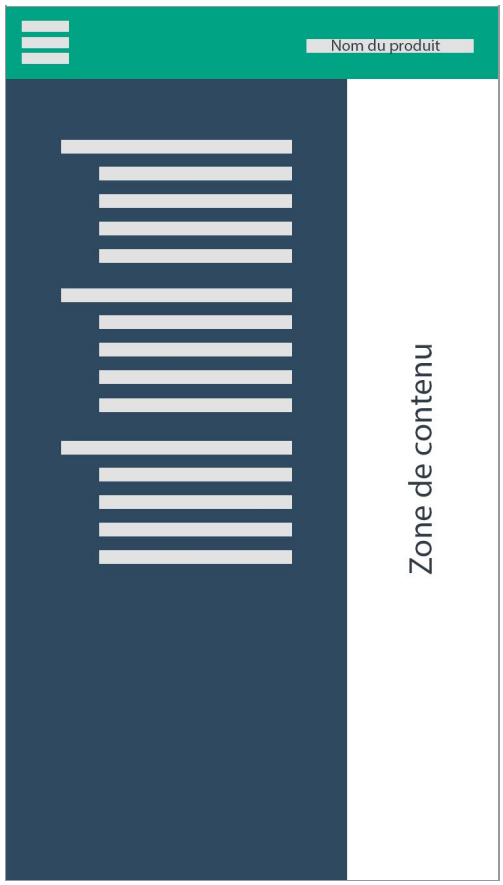
La version classique propose un menu latéral déplié en permanence puisque les écrans d'aujourd'hui ont tendance à être plutôt large que haut. Le contenu dépendant des fonctionnalités s'affiche à droite du menu.



Version mobile

Le mobile est en quelque sorte à l'antipode des ordinateurs de bureaux, parce qu'ici c'est plutôt la hauteur qui grandit et non pas la largeur. L'interface graphique propose donc un menu latéral masquable/démasquable sur un clic utilisateur pour gagner en zone de travail.

Menu masqué

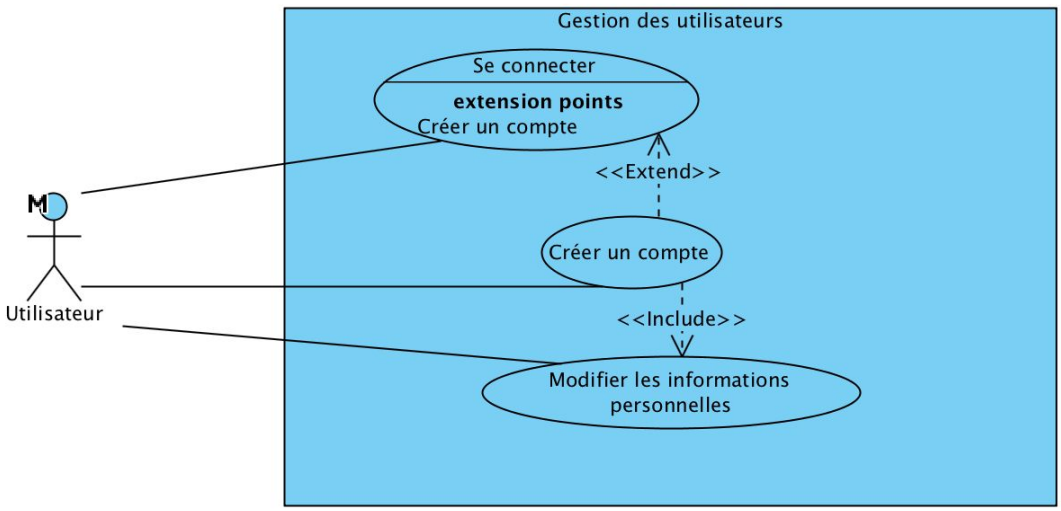


Menu démasqué

Système

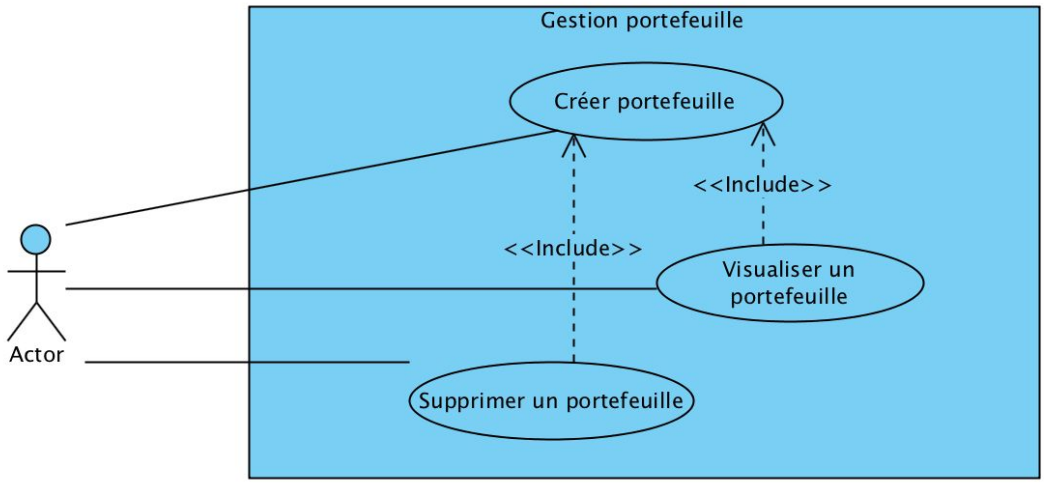
Vue fonctionnelle

Gestion des utilisateurs



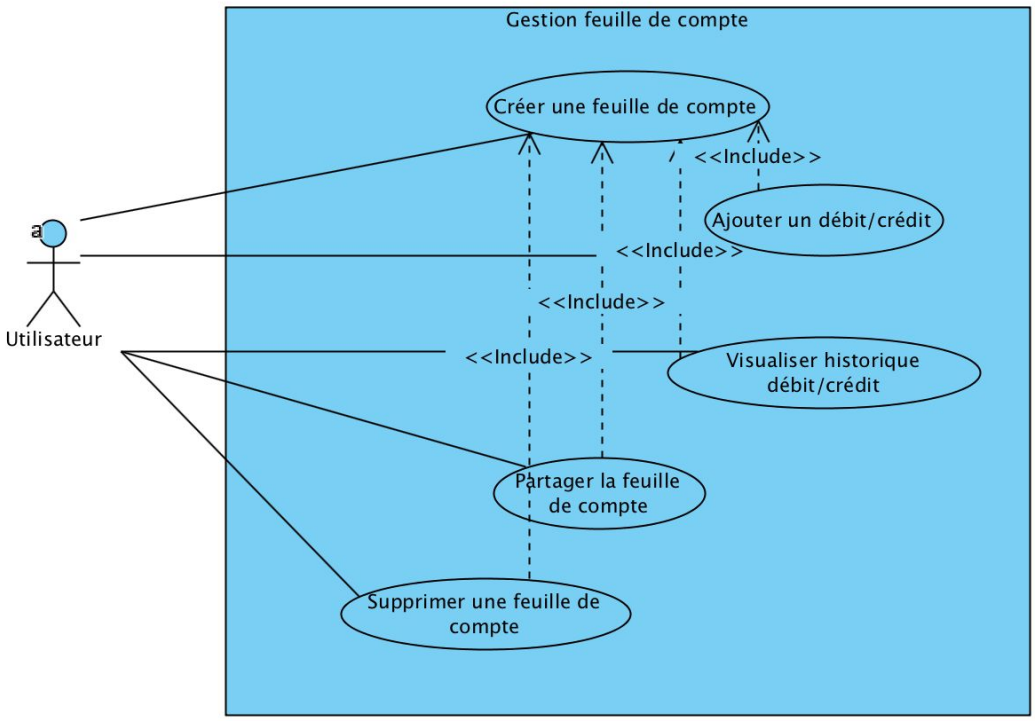
Identifiant	Label	Description
UCUSER1	Se connecter	Permet à l'utilisateur de se connecter au produit. Nécessite de posséder un compte précédemment enregistré.
UCUSER2	Créer un compte	L'utilisateur doit pouvoir créer un compte via une interface pour accéder au produit.
UCUSER3	Modifier les informations personnelles	Une fois un compte enregistré, l'utilisateur peut à tout moment en modifier les informations par une interface.

Gestion des portefeuilles



Identifiant	Label	Description
UCPORT1	Créer un portefeuille	L'utilisateur a la possibilité de créer un portefeuille qui permet l'agrégation de plusieurs feuille de compte (voir gestion des feuilles de compte)
UCPORT2	Visualiser un portefeuille	L'utilisateur peut visualiser les informations d'un portefeuille à tout moment. (Solde global, feuilles de compte associées)
UCPORT3	Supprimer un portefeuille	L'utilisateur peut supprimer l'un de ses portefeuilles quand il le souhaite.

Gestion des feuilles de comptes



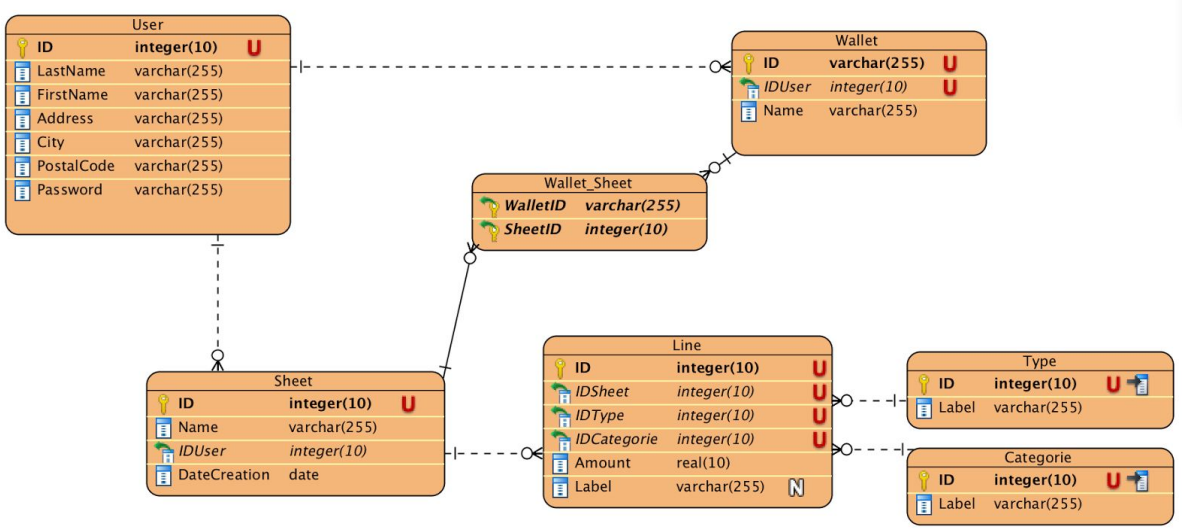
Identifiant	Label	Description
UCSHEET1	Créer une feuille de compte	L'utilisateur peut créer une feuille de compte. Cette feuille de compte permet le suivi d'opération.
UCSHEET2	Ajouter un débit/crédit	L'utilisateur peut saisir des entrées ou des sorties financières en spécifiant le type et la catégorie de gain ou de dépense.
UCSHEET3	Visualiser l'historique débit/crédit	L'utilisateur peut visualiser l'ensemble des opérations saisies dans sa feuille de compte avec le solde courant de celle-ci.
UCSHEET4 (Optionnel, faire à la fin de la réalisation du Backlog si du temps est	Partager la feuille de compte	Le propriétaire d'une feuille de compte peut partager la feuille de compte avec

encore disponible)		d'autre utilisateur pour une saisie collaborative.
UCSHEET5	Supprimer feuille de comptes	L'utilisateur doit pouvoir supprimer de son environnement une feuille de compte.

Vue statique

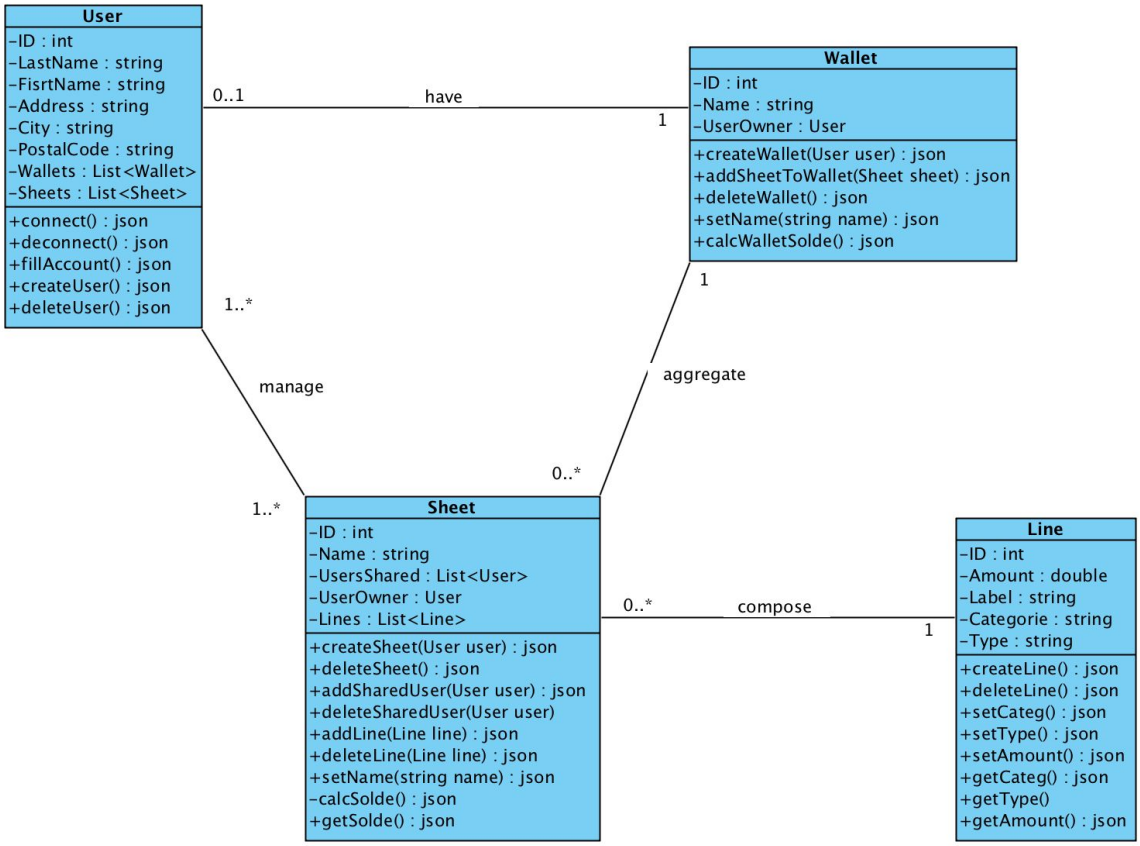
Représentation des données

Ce schéma désigne les données principales nécessaires au fonctionnement de l'application.



Représentation générale de l'architecture métier

Ce diagramme de classe permet de symboliser les "grandes" fonctionnalités de manipulation de la couche métier en les décomposant en méthodes.

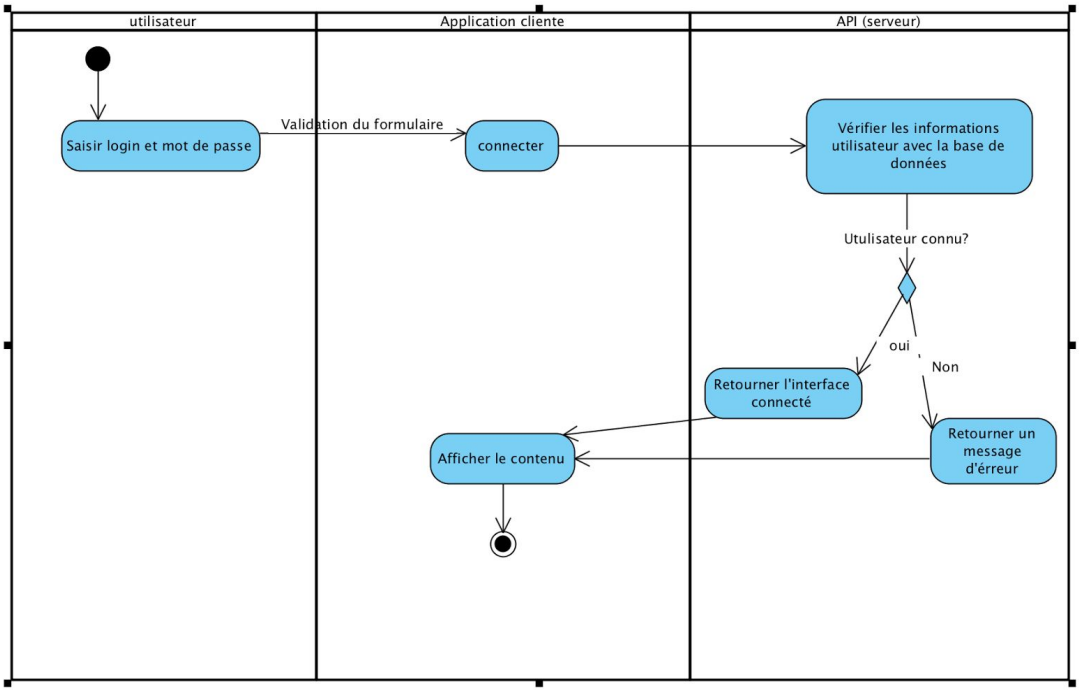


Vue dynamique

Gestion des utilisateurs

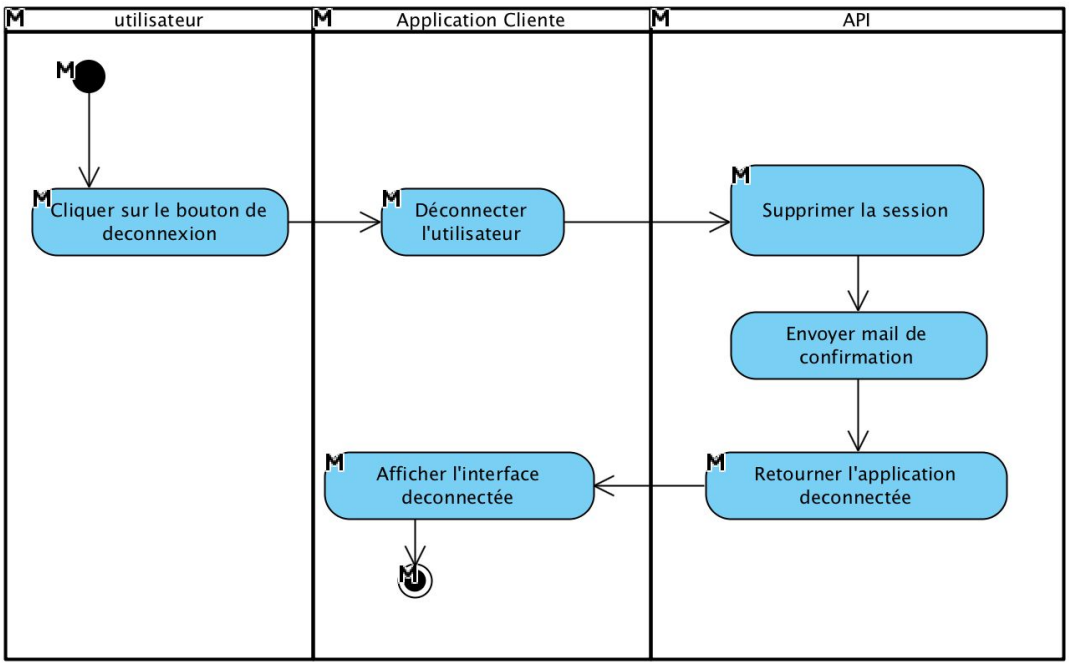
Connexion d’un utilisateur

Un utilisateur déjà enregistré dans le système doit pouvoir se connecter de la manière suivante :



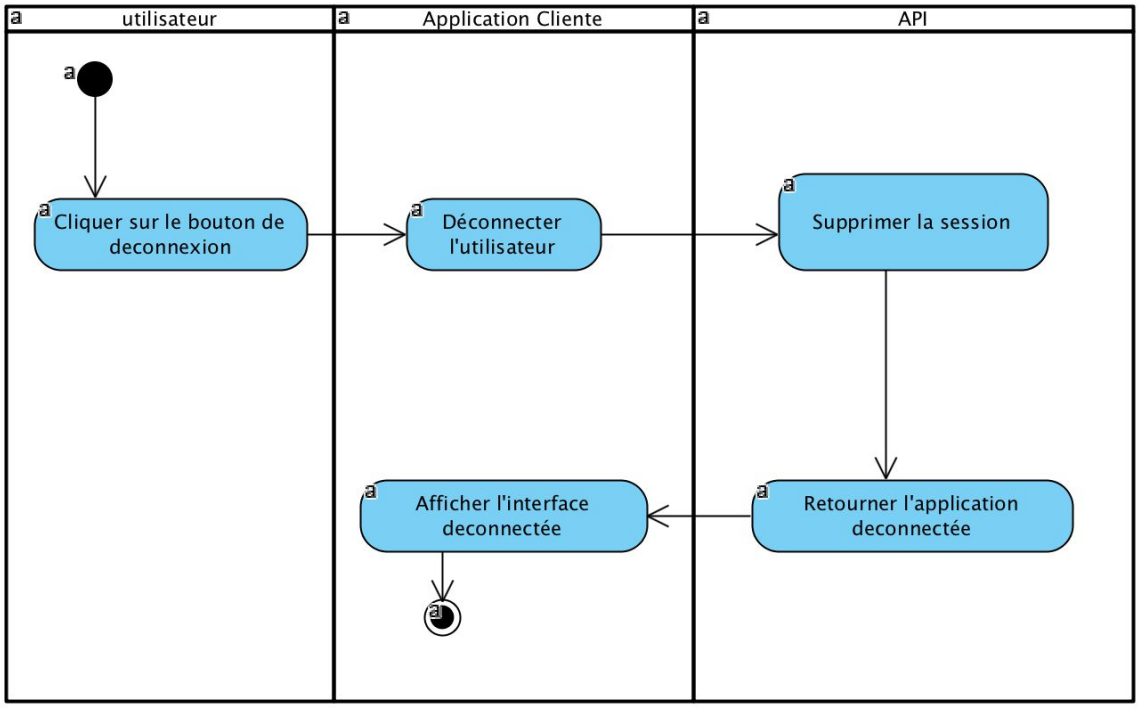
Déconnexion d’un utilisateur

Un utilisateur connecté doit pouvoir se déconnecter de la manière suivante :



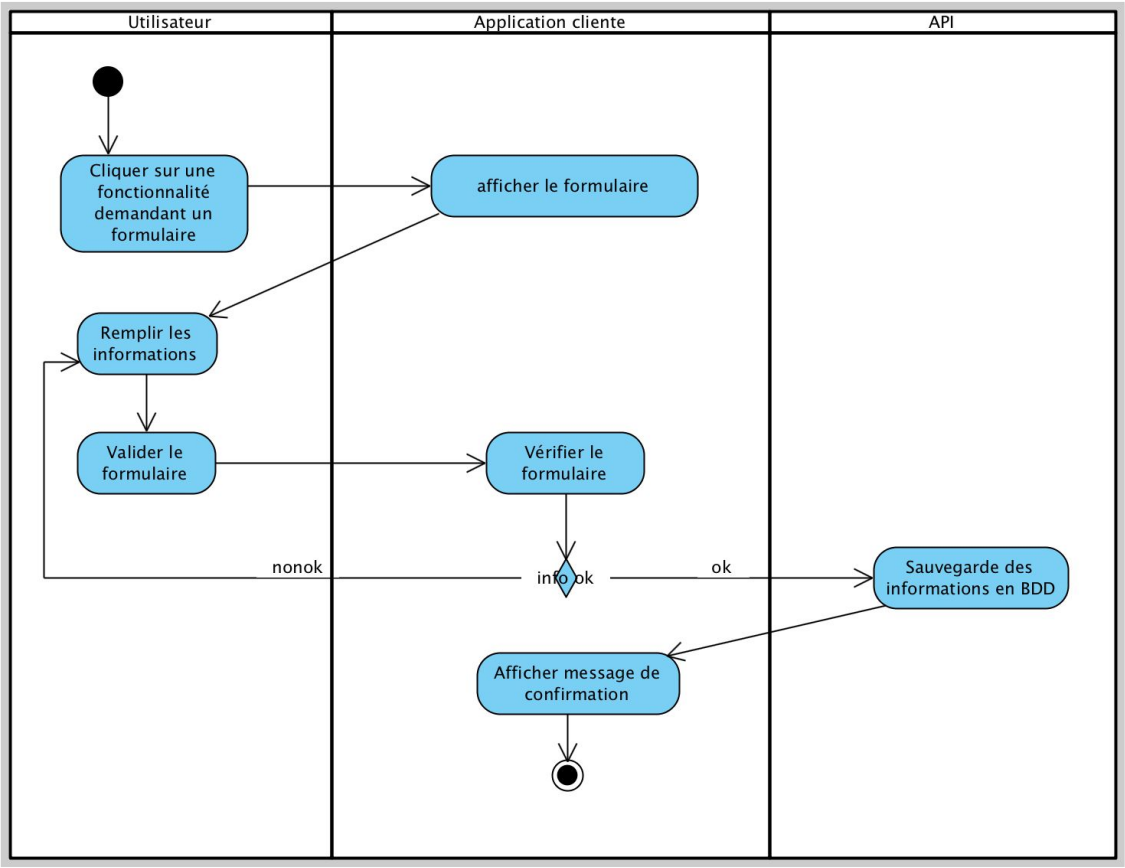
Création d’un utilisateur

Un utilisateur qui n’a pas de compte doit pouvoir en créer un pour accéder aux fonctionnalités du produit.



Gestion générique des autres fonctionnalités

Les autres fonctionnalités répondent aux même critères d'implantation et sont représentées de manière générique. (Utilisé notamment dans la création et la manipulation des objets Portefeuille et Feuille de compte).



Choix d'implémentation

Pour l'architecture, nous avons fait le choix d'utiliser un web service de type REST et une architecture distribuant une grande partie de la logique au coté client.

Cela nous permettra éventuellement d'ajouter une application mobile et un client desktop au projet tout en conservant une même application côté serveur.

Côté client : Reactjs et Redux

Pour l'application web nous avons fait le choix de Reactjs qui est la librairie javascript graphique développée par Facebook en open source. Ce choix a été motivé par la popularité croissante de cette librairie, notre connaissance du javascript et bien sûr notre désir de se former sur cette technologie.

Le fonctionnement de réact repose sur un système de composants graphiques imbriquées les uns aux autres, permettant de réutiliser facilement ces éléments. Contrairement à d'autres technologies populaires comme AngularJs (google), il n'existe pas de système de data binding permettant d'associer des données directement avec un affichage. Ici React maintient un "DOM virtuel" qu'il synchronise avec le DOM du navigateur grâce à un

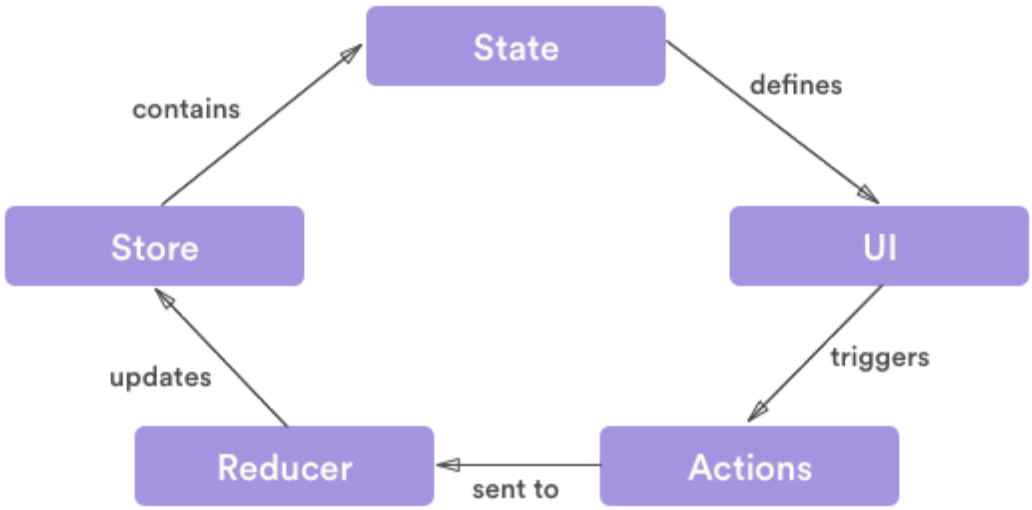
algorithme extrêmement performant. C'est rapide et permet d'éviter certain problèmes liés au data binding.

Cependant comme précisé, ReactJS n'est pas un framework mais une librairie graphique. Pour permettre une gestion des données uniforme, cohérente et stable avec l'accroissement de la complexité de notre produit, nous avons choisis d'utiliser **Redux.**

Redux est un conteneur d'etat prédictible inspiré de l'architecture Flux. Il permet de séparer les données, la logique, et le moteur graphique de notre application. Les données sont "rangées" à un endroit unique, le store.

Redux contient:

- Des **actions** définissent simplement l'ensemble des changements opérables sur le State sont définies. Les actions sont en quelque sorte un message demandant une transformation.
- Des **reducers** sont des fonctions qui ont la charge de réaliser les transformation du *State* en fonction d'actions qui leurs sont communiquées. Chaque reducer à la responsabilité d'une partie réduite de l'ensemble de l'état de l'application.



Nodejs

Nodejs est une plateforme permettant de développer des applications javascript côté serveur. Nous avons fait ce choix car nous connaissons déjà cette technologie et que cela nous permet d'utiliser un unique langage pour le client et le serveur. De plus nodejs est très rapide et conviendrait si l'application venait à être utilisée par un grand nombre d'utilisateurs simultanéments.

mongoDB

Pour permettre la persistance des données nous avons choisi d'utiliser une base de donnée noSQL orienté document. Ce choix a été guidé par la popularité de cette plateforme, la simplicité de mise en oeuvre et la modularité permise sur le schéma de base. Cette modularité est requise car, dans une logique agile, nous savons que le besoin est susceptible d'évoluer rapidement. Son utilisation avec javascript et nodeJS est particulièrement simple puisque la base permet de persister directement les objets JSON utilisés dans l'application.

Conduite de projet

L'agilité est la méthode de projet que nous avons choisi pour réaliser le projet. En effet, la structure organique de cette méthode nous permettra d'affiner notre produit tout au long de son cycle de développement.

Grandes phases du projet

Cependant, voici dans les grandes lignes les plages de jours des grandes phases que nous assignons aux projets. (Prendre en compte que les durées sont très étalés du fait de nos obligations salariales et universitaires). Les plages de dates correspondent à celles communiquées dans le calendrier des rendus fourni par l'université.

Phase	Description	Dates	Durée en jours
Analyse	Présentation du projet Définition des fonctionnels Précision sur les technologies adoptées	Du 01/01/17 Au 15/02/17	33 jours
Architecture	Définition de l'architecture Définition des composants du projet Montrer les utilisations des composants hérités de framework	Du 16/02/17 Au 31/02/17	32 jours

Developpement & tests	Développement des attendus fonctionnels, de référentiels de données et d'interface Exécution de la batteries de tests et correction des anomalies	Du 3/03/17 Au 2/06/17	45 jours
Livraison du projet à l'université	Mise en production Fourniture des rendus Bouclage du projet	Du 5/06/17 Au 16/06/17	15jrs

Agilité

L'utilisation de la méthode Agile version Scrum va permettre de réaliser 3 critères essentiels au cycle de développement du projet :

- La transparence : L'équipe doit se comprendre en utilisant le même langage
- L'inspection : Chaque anomalie du produit doit être détecté au plus tôt.
- L'adaptation : Chaque élément qui ne correspondrait plus aux attentes doit être revu.

Le backlog est la feuille de route concernant la liste des fonctionnalités à développer.

Le développement sera découpé en plusieurs phases que l'on nomme des sprints tel que le préconise la méthode originale.

Chacun de nos sprints va durer 5 jours environ et consistera en la réalisation d'une liste de fonctionnalités extraite du backlog.

Le backlog pourra être alimenté par de nouvelles fonctionnalités ou par la modification/correction de certaines d'entre elles pour conserver une dynamique de projet agile et correspondre au mieux aux besoins énoncés initialement.

Bibliographie

Méthode agile Scrum
https://fr.wikipedia.org/wiki/Scrum_(Boite_%C3%A0_outils)

Téchnologies Web :
https://nodejs.org/en/
https://facebook.github.io/react/
http://redux.js.org/
https://www.mongodb.com/fr