Moneythoring

Rapport

HEIG-VD

Projet de semestre

Table des matières

1 Descriptif 2

2 Fonctionnalités (A METTRE A JOUR A LA FIN DU RAPPORT) 3

2.1 Fonctionnalités de base 3

2.2 Fonctionnalités optionnelles 5

3 Architecture (A METTRE A JOUR) 7

4 Choix d’implémentation 8

4.1 Langage de programmation et GUI 8

4.2 Framework 8

5 Bases de données (DB) 9

6 Interface graphique (GUI) 10

7 Couche d’accès aux données(DAL) 11

8 Logique métier (BLL) 12

9 Conclusion 13

# Descriptif

Avec MoneyThoring, nous proposons à l’utilisateur une gestion de ses transactions, qu’elles soient de simples dépenses ou revenus, des virements entre comptes, des dettes ou encore des factures. Nous lui apportons aussi la possibilité de catégoriser ses transactions, de créer des objectifs de budget pour pouvoir suivre les dépenses dans des domaines précis et ainsi d’avoir une vue la plus informative possible sur l’évolution de son capital.

Ainsi, MoneyThoring est une application proposant à l’utilisateur de créer un compte (fonctionnalité en ligne) ou de travailler uniquement en local (hors ligne). Selon ce choix, certaines fonctionnalités sont accessibles ou pas. Bien sûr, même si l’utilisateur possède un compte, s’il vient à manquer d’une connexion internet, il peut sans autre continuer son travail hors ligne. Ce dernier étant synchronisé une fois la connexion avec la base de données en ligne rétablie.

Certaines fonctionnalités ne sont disponibles qu’avec la création d’un compte utilisateur et d’une connexion à internet. Ces fonctionnalités concernent effectivement plusieurs utilisateurs et ne peuvent donc pas être gérées localement. Exception faite de ces dernières, tout est réalisable en local, avec ou sans compte utilisateur.

MoneyThoring propose un design poussé pour que l’information soit la plus claire, précise et que l’utilisateur n’ait pas à se creuser la tête pour comprendre son utilisation.

# Fonctionnalités (A METTRE A JOUR A LA FIN DU RAPPORT)

## Fonctionnalités de base

### Création compte utilisateur

Si l’utilisateur ne possède pas de compte à l’ouverture de l’application, il peut choisir d’en créer un ou de continuer sans. En choisissant la première option, qui n’est accessible que si l’utilisateur à une connexion internet, la création du compte se fait à l’aide d’un formulaire qui lui demande son adresse email, son nom d’utilisateur et un mot de passe, qu’il est nécessaire de confirmer une deuxième fois. Une fois le formulaire envoyé, un email de validation contenant un code est envoyé. Une fois le code saisi dans l’application, le compte est activé.

En choisissant la deuxième option, l’utilisateur peut sans autre utiliser l’application mais n’aura pas accès aux fonctionnalités partagées entre utilisateurs.

### Connexion sécurisée

Pour les utilisateurs possédant un compte validé, une connexion est exigée au démarrage de l’application. Cette connexion évite qu’un tiers puisse modifier les informations, même en travaillant en hors ligne (les données étant synchronisées par la suite). Cette demande de connexion est un simple formulaire dans lequel l’utilisateur doit entrer son nom d’utilisateur et son mot de passe. L’application ne nécessite pas de double authentification.

Le mot de passe est haché, avec un sel ajouté, dans la base de données. Il n’est pas sauvé en clair.

### Compte bancaire

Un utilisateur peut ajouter un ou plusieurs comptes bancaires. Chaque compte possède un nom, un type, le nom de la banque qui le concerne (optionnel), le montant actuel, un pourcentage d’intérêt éventuel et s’il faut l’utiliser comme compte par défaut lors de transactions. Toutes ces informations, exceptée le solde du compte qui est modifié automatiquement par les transactions, peuvent être modifiées par la suite.

Les comptes bancaires peuvent également être supprimés. L’utilisateur peut alors choisir de virer le solde vers un autre compte ou de perdre ce dernier.

### Catégorie

Il est possible de créer d’autres catégories que celles proposées par défaut. Une catégorie est définie par un nom et une couleur. Toutes les catégories, même celles proposées par défaut, peuvent être modifiées et supprimées.

### Devise

Une conversion automatique est faite lorsque les transactions effectuées sont dans une autre devise que CHF, en fonction du taux du jour. Cette fonctionnalité nécessite une connexion internet, le logiciel devant aller chercher les taux sur internet.

### Transactions

Les transactions regroupent toutes les entrées et sorties d’argent.

L’utilisateur peut ajouter des revenus ou des dépenses. Chaque transaction est définie par un montant, une catégorie, un compte affecté (par défaut le compte sélectionné par l’utilisateur lors de sa création), une devise utilisée, le taux de change de la journée et son type (revenu ou dépense).

Les transactions peuvent également avoir une notion de récurrence (salaire et factures). Celle-ci peut être annuelle, mensuelle ou plus spécifique (2 semaines, 2 mois, etc.) et s’exécuter à une date précise, par exemple tous les 25 du mois.

Les transactions peuvent être supprimées ou modifiées dans le cas d’erreurs.

### Virement compte à compte

Il est possible d’enregistrer des virements entre les différents comptes d’un utilisateur. Un virement est pris en compte comme une transaction. Il y a donc une dépense (transaction sortante) pour le compte à débiter et un revenu (transaction entrante) pour le compte à créditer.

### Dettes

L’utilisateur a la possibilité d’enregistrer ses dettes, qu’il en soit le débiteur ou le créancier. Il y a deux types de dettes, les dettes simples, qui ne sont qu’une information pour l’utilisateur, et les dettes synchronisées, qui lient deux utilisateurs de l’application.

Chaque dette possède un montant, un intérêt, une date limite et une description. Pour les dettes synchronisées, il est possible de spécifier un nom d’utilisateur.

Une fois une dette acquittée, l’utilisateur peut la valider et la transaction qui en découle est automatiquement ajoutée. Dans le cas de dettes synchronisées, il est nécessaire que les deux partis confirment la dette, à la réception et à l’acquittement.

### Vue globale

L’utilisateur peut en tout temps suivre l’évolution de son budget, grâce à une vue globale (depuis le début du budget jusqu’à maintenant), annuelle ou mensuelle. Cette vue permet de comparer de manière graphique les différentes dépenses et rentrées d’argent. En plus d’être filtrée chronologiquement, elle peut également l’être selon le mode d’affichage ; graphique en courbe, diagramme circulaire, diagramme en barres.

#### Graphique en courbe

Le graphique en courbe permettra de voir rapidement et simplement l’évolution des comptes.



#### Diagramme circulaire

Deux diagrammes circulaires affichent les dépenses et les entrées par catégorie. Les pourcentages exacts sont ajoutés à côté du graphique.

### Budget

Un budget se caractérise en deux types : les budgets récurrents (en fonction des catégories) et les budgets ponctuels (en fonction de deux dates données début/fin). Ces budgets permettent de voir l’évolution des dépenses dans un domaine précis, tout en permettant à l’utilisateur d’être notifié quand il s’approche de la limite fixée. Les budgets sont représentés de manière graphique sur une ligne, avec le budget qu’il lui reste (vert par exemple) et les dépenses déjà effectuées (rouge). L’utilisateur a également la possibilité de créer des budgets partagés, où les dépenses de chaque utilisateur sont communes et le budget évolue en fonction des participants (par exemple un budget de voyage ou de commissions pour la famille).

Un budget possède un type, un nom, un montant et des catégories (aucune, une ou plusieurs). Lors de l’ajout du montant, l’utilisateur peut choisir une devise différente que celle par défaut. Si le budget est ponctuel, une durée est demandée (date de début et de fin).

Il est possible de supprimer n’importe quel budget. L’utilisateur peut modifier le montant, la devise, la durée et les catégories d’un budget.

### Budgets partagés

Pour les budgets partagés, l’utilisateur peut inviter d’autres utilisateurs (via leur nom d’utilisateur) à rejoindre le budget. Les personnes concernées reçoivent une notification pour accepter l’invitation.

N’importe quel utilisateur d’un budget partagé peut décider d’en sortir. Dans ce cas-là les dépenses effectuées par l’utilisateur restent enregistrées dans le budget. Mais seul le créateur peut le supprimer totalement.

## Fonctionnalités optionnelles

### Vue globale



#### Diagramme en barres

Le diagramme en barres permet la comparaison entre les dépenses et les revenus d’argent par mois et par année sous forme de deux barres de couleurs différentes.

### Liste de souhaits

L’utilisateur peut créer une liste de souhaits, composée de différents articles qu’il aimerait prochainement acheter. Ces articles sont identifiés par un nom, éventuellement un lien internet et un prix.

Cette liste est mise à jour en fonction du budget actuel. Si l’épargne de l’utilisateur est suffisante, alors le produit est catégorisé comme étant achetable. Si au contraire l’utilisateur n’a pas suffisamment de fonds disponibles, une estimation de temps d’attente avant achat possible est calculée sur la base de l’évolution de l’épargne.

### Prévisions d’achat

Liste de souhaits “prioritaire”, qui correspond à “il faudrait (absolument) que j’achète l’article X avant la date Y”, X étant un objet (p.ex un cadeau, un outil, une fourniture, …).

La prévision d’achat est comme la liste de souhaits au niveau des informations, excepté qu’il faut une date limite pour l’achat. L’utilisateur a également la possibilité de mettre une alarme un certain temps avant la date limite, ainsi l’application pourra envoyer un rappel. La dépense sera effective une fois que l’utilisateur a confirmé l’achat.

### Budgets partagés

Depuis un budget partagé, il est possible de créer les dettes sous-jacentes concernant les utilisateurs du budget. Par exemple, si dans un budget partagé une personne paie l’entièreté d’un produit, il peut créer des dettes synchronisées avec chacun des autres utilisateurs, soit automatiquement (en divisant le montant de manière équitable entre les personnes) soit en précisant le montant pour chaque utilisateur.

### Exportation en PDF

Chaque vue est exportable au format PDF par l’utilisateur.

### Dettes

Possibilité de scanner les documents en lien avec la dette et les garder dans l’application avec la dette concernée. Fichier à titre informatif, l’application ne l’interprète pas.

### Simulation

Une simulation est possible en se basant sur les revenus et les dépenses récurrentes (salaire, abonnements, factures, loyer, capitalisation du compte, ...) ainsi que sur les données utilisateur déjà dans l’application, en prenant par exemple des moyennes de dépenses.

Un utilisateur peut modifier ses dépenses et revenus en mode simulation, s’il souhaite voir comment évolue son budget sans pour autant sauvegarder les modifications réalisées. L’activation de ce mode est repérée par des indications visuelles, permettant à l’utilisateur de ne pas se tromper de mode.

# Architecture (A METTRE A JOUR)



L’architecture logicielle est conçue pour fonctionner sur le système d’exploitation Windows 10. Le logiciel est découpé en trois couches, chaque couche a une responsabilité précise :

* La couche présentation en bleu permet de présenter les informations à l’utilisateur dans des fenêtres graphiques.
* La couche logique métier en magenta gère toute la logique métier de l’application.
* La couche d’accès aux données en rouge va permettre de gérer la persistance et l’accès aux données.

# Choix d’implémentation

## Langage de programmation et GUI

L’application a été implémentée en Java. En ce qui concerne les aspects graphiques, nous avons utilisé JavaFX, pour pouvoir séparer de manière claire la couche purement graphique de celle qui implémente les fonctionnalités graphiques, comme les boutons.

## Framework

Nous avons également utilisé Hibernate. Il s’agit d’un ORM (Object Relational Mapping) qui permet de développer des applications, qui peuvent aisément gérer et accéder à des bases de données, récupérer, modifier et supprimer des données. Hibernate a été utilisé pour réaliser la couche d’accès aux données internes (Derby) et la couche d’accès aux données externes (PostgreSQL). Ces couches d’accès aux données sont responsables de la persistance dans un système de gestion de base de données relationnel.

L’utilisation d’Hibernate nous a permis d’écrire du code plus facilement maintenable et compréhensible, de gérer les mises à jour et les changements sur plusieurs relations. Avec Hibernate, concevoir une couche capable de gérer la persistance des données est moins problématique et fastidieux que de le faire entièrement à la main.

# Bases de données (DB)



La base de données de MoneyThoring, comme nous l’avons implémentée, possède deux acteurs majeurs : « client » et « budget ». A eux deux, ils représentent la majorité des interactions qu’il y a au sein de la base de données. Pour mieux comprendre notre projet, voici une description des diverses entités et interactions.

Un client possède un nom d’utilisateur (qui permettra aux autres utilisateurs de facilement le retrouver pour l’ajouter à un budget partagé), une adresse email, un mot de passe et un sel pour l’authentification, et une clé d’activation pour la création de son compte. Tant que le client n’a pas fait vérifier son adresse email lors de la création de son compte, il continue d’exister en base de données mais en tant qu’utilisateur « non activé ». Une fois la vérification faite, il peut sans autre se connecter et être reconnu en tant qu’utilisateur actif.

# Interface graphique (GUI)

# Couche d’accès aux données(DAL)

# Logique métier (BLL)

# Conclusion