Projet BAB3 : Modélisation des données

Groupe 5 : Finance

Petro Borys Martin Coghetto Alexandre Daoust Louis Garlement

2025

Plateforme de gestion de portfolios financiers

Table des matières

1. Cahier des charges

1.1 Contexte et définition du problème

Cette application s'inscrit dans le contexte de gestion des placements financiers. Aujourd'hui, les marchés financiers étant devenus pour la plupart numérisés, il devient très facile d'acheter et de vendre des actifs financiers grâce à diverses plateformes de brokers.

Cependant, il existe plusieurs types d'actifs financiers tels que des actions d'entreprises cotées en bourse, des fonds négociés en bourse (ETF), des obligations . . .Tous ces produits d'investissement sont gérés par divers gestionnaires d'actifs. Lors de la multiplication d'actifs composant un portefeuille, il devient plus difficile de faire un suivi global de son patrimoine lorsque les plateformes de gestion sont multiples.

De plus, certaines plateformes ne permettent pas un suivi des plus-values latentes, des évolutions et des perspectives d'investissement. Certains titres ne possèdent également pas de système de suivi.

1.2 Objectif du projet

Ce projet se focalisera principalement sur la conception d'une application web permettant le suivi des titres financiers gérés depuis plusieurs plateformes.

Notre application peut concerner autant des entreprises qui gèrent leurs finances que des particuliers, nous pourrions alors nous-mêmes être clients et le but visé est donc de permettre aux utilisateurs une gestion plus facile de leurs portefeuilles en ligne.

L'objectif est de proposer à un ou plusieurs utilisateurs un gestionnaire de portfolios financiers permettant à chacun d'encoder ses transactions et de pouvoir avoir accès à l'entièreté de leurs informations centralisées, accessibles de manière plus pratique et au même endroit.

Ainsi, le portfolio d'un utilisateur lui permet de visualiser ses actifs, mais également le cours des monnaies, actions. . . qu'il souhaite connaître, l'historique de ses achats et ventes (encodés par l'utilisateur lui-même) ainsi que des statistiques sur ses gains (plus-values latentes, pourcentages, etc.)

1.3 Périmètre du projet

L'application n'a pas pour but d'être une plateforme de trading, elle se focalise sur la gestion de portefeuille uniquement. On ne peut donc que visionner ses actifs (pas d'achat ni de revente).

Nous pourrions imaginer introduire dans l'application la gestion d'actifs autres que financiers, mais cela n'est pas l'objectif de l'application. Elle gérera uniquement des actifs financiers.

L'application sera capable de recevoir des données de la part de l'utilisateur. Celui-ci pourra, par exemple, encoder manuellement ses transactions. Nous pourrions envisager l'encodage automatique des transactions via l'utilisation des APIs des courtiers ou encore via la lecture de fichiers exportés. Cependant, la trop grande diversité de brokers et la myriade de format de fichiers différents risque de nous compliquer la tâche. Nous avons estimé que cela sortait du cadre du projet de base de données et ne serait pas implémenté pour l'instant.

L'application utilisera un système de gestion de base de données supportant le **SQL**.

L'application sera accessible depuis un **navigateur web**.

L'application sera **multi-utilisateurs**.

La langue de l'application sera le **français** et elle se basera sur un **fuseau horaire belge** (`UTC+1` ou `UTC+2` en fonction de l'heure d'hiver ou d'été).

1.4 Description fonctionnelle des besoins

L'application est multi-utilisateur, chaque utilisateur peut se connecter en utilisant son identifiant (adresse mail) et son mot de passe. Un nouvel utilisateur peut également créer un compte sur l'application, pour cela il doit indiquer son adresse mail et un mot de passe à utiliser. Il ne peut pas y avoir 2 utilisateurs ayant la même adresse mail. L'utilisateur doit pouvoir modifier son mot de passe et supprimer son compte. En cas de suppression d'un compte les données associées au compte de l'utilisateur doivent également être supprimées.

Une fois connecté à son compte, un utilisateur peut accéder à la liste de ses portfolios. Un portfolio possède un nom. Plusieurs portfolios peuvent posséder le même nom. Un utilisateur peut posséder aucun ou plusieurs portfolios. L'utilisateur a la possibilité de créer un portfolio en indiquant son nom. Il devient alors son propriétaire.

L'utilisateur peut ensuite accéder au portfolio. Un utilisateur étant le propriétaire d'un portfolio peut décider d'ajouter d'autres utilisateurs (identifiés par leurs adresses mail) à un portfolio, il peut choisir si les utilisateurs ont des droits d'accès en modification ou en lecture seule. Il peut également avoir accès à la liste des utilisateurs ayant accès au portfolio et changer leurs droits d'accès, leur révoquer l'accès ou encore transférer la propriété du portfolio vers un autre utilisateur. L'utilisateur ayant la propriété du portfolio peut changer son nom et décider de supprimer le portfolio.

Le portfolio possède une et une seule devise, ce dernier est composé de divers titres, un titre peut être de plusieurs sortes : actions d'entreprise, fonds indiciels (`ETF`), obligations ou encore devises.

Les actions sont identifiées par la bourse où elles sont échangées (`euronext`, `nasdaq`, ...), l'identifiant de l'entreprise (`NVDA`, `AAPL`, `AMZN`, ...) ainsi que la devise (`EUR`, `USD`) dans laquelle le titre est échangé. Les informations sur les bourses sont également renseignées (identifiant, nom complet, emplacement, heure d'ouverture, heure de fermeture dans le fuseau horaire `UTC`). L'entreprise est renseignée par des informations (code ISIN, identifiant, nom complet, secteur d'activité, emplacement).

Les spécificités des autres types de titres (fonds indiciels, obligations, devises) seront détaillées dans la phase de modélisation technique.

Le portfolio conservera l'historique complet des transactions, chaque entrée comprend le type de titre (`action`, `entreprise`, `ETF`, ...), la date et l'heure de l'achat (dans la devise du titre), le nombre acheté (qui n'est pas forcément entier, pour supporter les parts fractionnées), le prix d'achat (dans la devise du titre), l'éventuel prix et date et heure de vente si le titre ne fait plus partie du portfolio, les frais de courtage (lors de l'achat et de la vente si vendu).

L'utilisateur pourra consulter cet historique trié par date ou par type d'opérations.

Lorsqu'un titre est vendu, il n'apparaît plus dans le portfolio mais peut rester affiché dans l'historique des ventes du portfolio.

Les données du cours d'échange associées à chaque titre sont, dans la mesure du possible, enregistrées dans la base de données. Dans le cas d'une action si la bourse est supportée par l'application, les données du cours de l'action de l'entreprise seront synchronisées [1] dans la base de données périodiquement afin d'avoir des données historiques permettant de déterminer les valeurs de reventes théoriques des titres possédés, calculer les plus values latentes et donner la performance du portfolio. Les données des instruments financiers seront sauvegardées dans la base de données pour chaque jour. Ces données comprendront les valeurs d'ouverture, de clôture, ainsi que les valeurs les plus hautes et les plus basses. Pour les données plus granulaires, nous ferons la requête au service api lorsque l'utilisateur aura besoin de ces données.

Via l'application, un client connecté à son compte pourra encoder les nouveaux placements de trésorerie, consulter l'historique de ses actifs ainsi que les performances en temps réel de ceux-ci. Pour les entreprises, un portfolio peut être partagé entre plusieurs employés. Pour ajouter une action, un client doit préciser les paramètres éventuels (actif acheté/vendu, date d'achat/vente, prix d'achat/vente). Par le menu principal,

1. Via service `API` externe

un client peut aussi consulter l'historique des actions avec les statistiques financières comme ROI [2] et PnL [3] fournies.

Fonctionnalité	**Description**	**Acteurs concernés**
Gestion des utilisateurs	Création d'un compte à partir d'une adresse mail et d'un mot de passe. Connexion et déconnexion sécurisées.	Utilisateur
Création de portfolio	Un utilisateur peut créer un ou plusieurs portfolios dont il est propriétaire.	Utilisateur
Partage de portfolio	Le propriétaire peut donner des droits d'accès (lecture ou écriture) à d'autres utilisateurs par adresse mail.	Propriétaire, utilisateur invité
Gestion des titres	Ajout, suppression et modification de titres (actions, ETF, obligations, devises).	Utilisateur autorisé
Historique des transactions	Suivi des achats et ventes, montants, frais, plus-values latentes, etc.	Utilisateur autorisé
Consultation des marchés	Affichage des cours récents ou historiques provenant d'une API externe.	Tous les utilisateurs
Statistiques et visualisations	Présentation graphique des performances et répartitions du portefeuille.	Tous les utilisateurs

TABLE 1.1 – Synthèse des fonctionnalités principales de l'application

1.5 Partage des tâches entre les membres du groupe

Nous allons diviser les tâches entre les différents domaines de l'application.

Domaine	**Membre**
Gestions des utilisateurs	Louis Garlement
Gestions des transactions/titres	Martin Coghetto
Gestions des portfolios	Petro Borys
Gestions des données des cours de bourse	Alexandre Daoust

TABLE 1.2 – Répartition des tâches en fonction du domaine de l'application

Dans chaque domaine, le membre du groupe assigné réalisera la partie correspondante de modélisation des données (Modèles conceptuels, logiques et physiques) et sera chargé de l'implémentation des requêtes SQL associées, du backend et du frontend.

1.6 Enveloppe budgétaire

1.6.1 Coûts de développement

Nous allons estimer les coûts de développement en nous basant sur les crédits ECTS. Le cours de *Modélisation des données, Big Data et projet* est valorisé à 5 crédits ECTS. Un crédit ECTS est défini comme 30h de travail. 50% de la note est attribué à notre projet : cela correspond à 2.5 crédits ECTS soit 75h de travail. Nous allons considérer que 20% du projet est consacré a la préparation de l'épreuve orale, il ne seront donc pas considérés dans le temps de développement. Il reste donc 60h de travail par membre du projet.

Cela fait donc un temps total de développement pour 4 développeurs de **240h**. À 20€ de l'heure cela donne donc **4800€** de coûts de développement.

2. Retour sur investissement
3. Profit And Loss : pertes et profits sur une transaction financière

1.6.2 Coûts opérationnels

Base de données

En envisageant une réplication pour le système de gestion de base de données (`SGBD`) sur 3 serveurs et compte tenu de la taille grandissante de la base de données en raison des données temporelles des cours de bourse, le coût d'hébergement de celle-ci est non linéaire et dépendra du nombre d'utilisateurs et de la quantité de données accumulées au cours du temps.

Type de solution	**Petit volume (< 100 GB)**	**Moyen volume (100-500 GB)**	**Grand volume (> 500 GB)**
VPS (OVH)	13,74-21,42 €/mois	21,42-166,83 €/mois	166,83-229,35 €/mois
Base managée (OVH)	0,068 € HT/heure	0,1346 € HT/heure	0,5436 € HT/heure
Cloud public (AWS)	$71,13-1517,26/mois	$1517,26-1623,66/mois	$1623,66-1756,66/mois

Table 1.3 – Estimation des coûts d'hébergement de la base de données

Application

L'application aura besoin d'un serveur au minimum et éventuellement de plusieurs pour avoir une réplication du service (et ainsi éviter toute interruption liée à un problème sur le serveur).

Puisque l'application est indépendante de la base de données, nous pouvons considérer plusieurs architectures pour l'hébergement de l'application :

— Architecture verticale
— Architecture horizontale

Les besoins en serveurs seront différents selon l'architecture. Nous pouvons choisir entre les deux, ou bien implémenter la combinaison des deux. Voici un tableau reprenant chaque cas envisagé :

Architecture	**Verticale**	**Horizontale**	**Combinée**
VPS (OVH)	4,68-49,98 €/mois	14,28 €/mois/serveur	4,68-49,98 €/mois/serveur
Cloud (AWS)	$0.0047-5.517/h	$0.4224/h/instance	$0.0047-5.517/h/instance
Dédié (OVH)	65,99-1199,99 €/mois	65,99 €/mois/serveur	65,99-1199,99 €/mois/serveur

1.6.3 Api

Pour que l'application fonctionne correctement, nous aurons besoin d'une api fiable. En effet, nous ne disposons pas, tel quel, des données boursières des instruments financiers des clients. Cette api doit être capable de nous fournir toutes les données pertinentes, données que l'application utilisera pour fournir le service.

Voici les services que nous avons analysés :

Service	**Prix**	**Limites**	**Notes**
Alpha Vantage	gratuit	25 appels api/jour	
	$49,99/mois	75 appels/minute	délai de 15min sur le marché
Finhub	gratuit	60 appels api/minute	fonctionalités très limitées
	+ $199,99/mois	900 appels/minute	grand coût d'utilisation
MarketStack	gratuit	100 appels/mois	données hors-USA non disponibles
	$9,99/mois	10000 appels/mois	
Polygon.io	gratuit	5 appels/minute	données hors-USA non disponibles
	$29/mois	illimité	
Finage	$89/mois	100000 appels/mois	
TwelveData	gratuit	800 appels/jour	données hors-USA non disponibles
	$329/mois	2584 crédits/mois	
Yahoo Finance	gratuit	limite inconnue	

Nous concluons que, dans le cadre de ce projet, il nous sera préférable d'utiliser l'api de Yahoo Finance (à travers une librairie).

Ceci dit, pour un projet de plus grande ampleur, nous aurions envisagé un autre service tel que Finage, ou bien une combinaison de services pour combler les lacunes de chacun à moindre coût.

1.7 Délais

1.7.1 Échéances

Date	Livrable
05/10/2025	Cahier des charges
26/10/2025	Cahier des charges + MCD + MLD
12/12/2025	Rapport final, Code source et présentation du projet

Table 1.4 – Échéances des livrables du projet

1.7.2 Planification temporelle

Semaine	Description
29/09/2025 - 05/10/2025	Réalisation cahier des charges
06/10/2025 - 12/10/2025	Réalisation MCD
13/10/2025 - 19/10/2025	Réalisation MCD
19/10/2025 - 26/10/2025	Réalisation MLD
27/10/2025 - 02/11/2025	Réalisation MPD
03/11/2025 - 09/11/2025	Implémentation
10/11/2025 - 16/11/2025	Implémentation
17/11/2025 - 23/11/2025	Implémentation
24/11/2025 - 30/11/2025	Implémentation
01/12/2025 - 07/12/2025	Implémentation
08/12/2025 - 12/12/2025	Implémentation & présentation orale

Table 1.5 – Planification temporelle des tâches du projet

2. Modèle Conceptuel de données

2.1 MCD

Voici le schéma correspondant au modèle conceptuel de notre base de données (figure 2.1). Ce dernier a été généré à l'aide de outil gratuit et open-source MoCoDo.

Le Portfolio est l'élement central de notre application. Le MCD présenté est normalisé en 3^{e} forme normale (3NF + 2NF + 1NF).

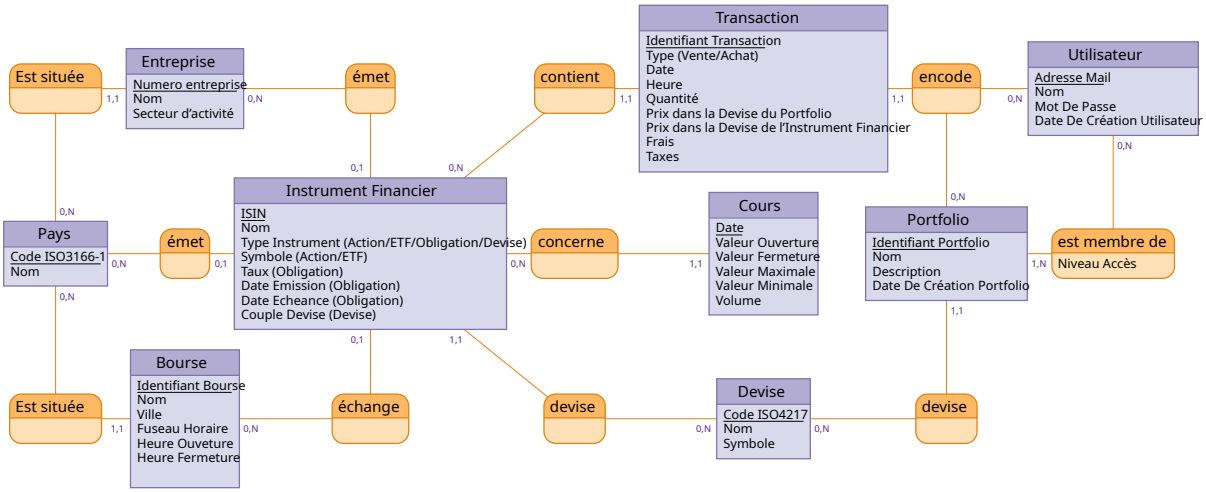


FIGURE 2.1 – Modèle conceptuel de données

2.2 Justifications

2.2.1 Relations

Un **utilisateur** est membre d'un ou de plusieurs **portfolios**. Il peut éventuellement n'être membre d'aucun portfolio s'il vient de créer son compte et n'a pas encore créé de portfolio. La relation d'un utilisateur à un portfolio comporte un attribut représentant son niveau d'accès (lecture seule, lecture/écriture ou propriétaire).

Un **portfolio** est associé à au moins un **utilisateur**.

Un **portfolio** possède une **devise** principale pour les actifs qu'il contient. Un portfolio doit nécessairement exister dans une et une seule devise. Dans le cas où un utilisateur souhaite gérer des actifs dans plusieurs devises, il doit créer un portfolio distinct pour chacune de ces devises.

Chaque **portfolio** contient un historique de **transactions**. L'historique de transactions d'un portfolio peut initialement être vide, c'est-à-dire qu'il ne comporte encore aucune transaction enregistrée.

Une **transaction** est nécessairement contenue dans un **portfolio**.

Une **transaction** est nécessairement encodée par un **utilisateur**.

Un **utilisateur** encode plusieurs **transactions**. Un utilisateur peut n'encoder aucune transaction.

Une **transaction** contient un seul **instrument financier**.

Un **instrument financier** peut être contenu dans plusieurs **transactions**.

Un **instrument financier** peut être échangé sur une **bourse**.

Un **instrument financier** peut être émis par un **pays**.

Un **instrument financier** peut être émis par une **entreprise**.

Un **instrument financier** est toujours exprimé dans une **devise** d'échange.

Un **cours** concerne nécessairement un **instrument financier**.

2.2.2 Séparation d'entreprise et de pays

Nous avons décidé de séparer les entités *entreprise* et *pays* au lieu de les regrouper dans une entité *Émetteur*. En effet, l'entité *pays* est reliée à une *bourse*. Il est plus cohérent qu'une **bourse** soit située dans un pays plutôt que dans un *Émetteur*.

2.2.3 Entité instrument financier

L'entité *instrument financier* est une entité particulière, car elle peut représenter plusieurs types d'entités, au nombre de quatre : *Action*, *ETF*, *Obligation*, *Devise*. Les champs et les relations de cette entité ne sont pas tous utilisés en fonction de la valeur du champ *type*.

Les instruments financiers ont pour caractéristique commune d'être tous identifiés de manière unique par un International Securities Identification Number (*ISIN*), qui est un standard (ISO 6166) utilisé dans le monde de la finance.

Tous les instruments financiers sont également échangés dans une seule devise.

Les *actions* et les *ETF* sont échangés sur une *bourse* et émis par des *entreprises*. Ils possèdent un symbole [1]

Les *obligations*, quant à elles, sont émises par des *pays* ou des *entreprises*, et possèdent un taux, une date d'émission et une date d'échéance.

1. Le symbole utilisé pour nommer cette action sur la bourse (ex. : AAPL, GOOGL, MSFT, AMZN, NVDA, ...).

3. Modèle Logique de données

3.1 MLD

Voici le modèle logique de notre base de données (figure 3.1).

```
Utilisateur(Email, Nom, Prenom, Mot_De_Passe, Date_Creation)

Membre_Portfolio(#Email_Utilisateur, #Identifiant_Portfolio, Niveau_Acces)

Instrument_Financier(ISIN, Nom, Type_Instrument, Symbole, Taux,
                     Date_Emission, Date_Echeance, Couple_Devise,
                     #Identifiant_Bourse_Echange, #Code_Pays_Emission,
                     #Numero_Entreprise_Emission, #Pays_Entreprise_Emission,
                     #Code_Devise)

Pays(Code_Pays, Nom)

Bourse(Identifiant_Bourse, Nom, Ville, Fuseau_Horaire, Heure_Ouverture,
       Heure_Fermeture, #Code_Pays)

Entreprise(Numero_Entreprise, #Code_Pays, Nom, Secteur_Activite)

Cours(#ISIN, Date, Valeur_Ouverture, Valeur_Fermeture,
      Valeur_Maximale, Valeur_Minimale, Volume)

Portfolio(Identifiant_Portfolio, Nom, Description, #Code_Devise,
          Date_Creation_Portfolio)

Transaction(Identifiant_Transaction, #Identifiant_Portfolio, #ISIN, #Email_Utilisateur, Type,
            Date, Heure, Quantite, Valeur_Devise_Portfolio,
            Valeur_Devise_Instrument, Frais, Taxes)

Devise(Code_Devise, Nom, Symbole)
```

FIGURE 3.1 – Modèle logique de données en 3e forme normale

3.2 Justifications

Dans notre MCD, les entités *Utilisateur* et *Portfolio* étaient liées par une relation plusieurs à plusieurs. De ceci est née la table Membre_Portfolio, afin de n'avoir que des relations un à plusieurs.

De plus, certains éléments ont été renommés (comme "Code ISO3166-1" => "Code_Pays") par soucis de lisibilité, ce dernier sera également simplifié lors de l'implémentation du MPD.

4. Modèle Physique de données

4.1 MPD

Voici un schéma physique de notre base de données (figure 4.1).

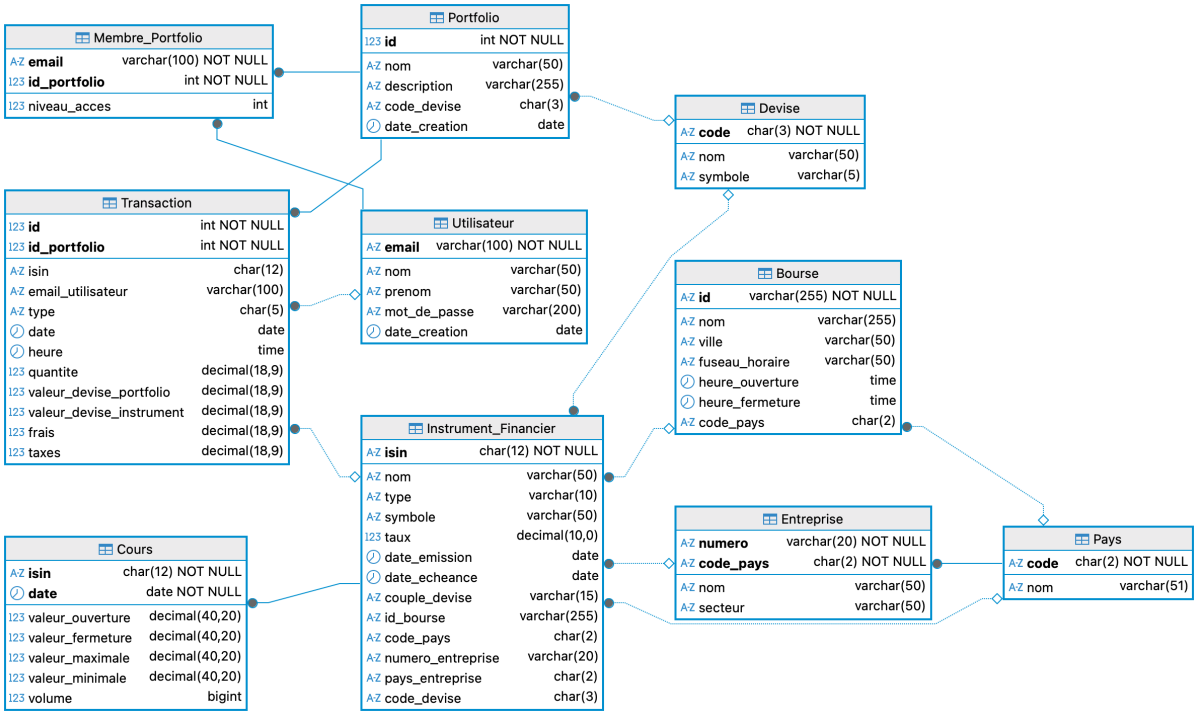


FIGURE 4.1 – Modèle logique de données

4.2 Justifications

Il est évidemment impératif que chaque identifiant ne soit pas nullable et que chaque attribut associé à l'argent soit un decimal [1]. Au delà de ceci, plusieurs choix de types et de taille de chaînes de caractères peuvent être précisés :

L'attribut *niveau_acces* de la table *Membre_Portfolio* caractérise la possibilité d'accéder à un portfolio en lecture seule, avec l'autorisation d'écrire, ou bien en admin. Ainsi, au lieu de traiter une chaîne de caractères, un entier représentera chaque niveau d'accès pour plus de simplicité.

L'attribut *type* de la table *Transaction* mesure 5 caractères : "Achat" ou "Vente".

L'identifiant de la table *Devise* est son code ISO4217, qui est toujours une suite de trois caractères.

1. Pour éviter les erreurs de calculs liées aux floats

Similairement, l'identifiant de la table *Instrument_ Financier* est son ISIN, qui est également toujours une chaîne de douze caractères.

Encore une fois, l'identifiant de la table *Pays* est son code ISO3166-1 alpha-2, qui représente donc deux caractères, et le nom peut mesurer jusqu'à 51 caractères, ce qui représente le nom complet du Royaume-Uni, pays avec le nom le plus long (cette petite précision peut cependant très probablement être considérée "overkill", et sa limite pourrait ainsi être revue à la baisse, comme pour certaines autres variables).

5. Choix technologiques et Implémentation

Liste des requêtes SQL

5.1 Choix technologiques

Le cahier des charges du projet requiert que l'application soit accessible depuis un navigateur. De plus, le système de gestion de bases de données doit supporter le SQL. Pour répondre à ces contraintes, nous avons réalisé les choix technologiques suivants :

- **GIT** : comme système de gestion de version.
- **MySQL** comme système de gestion de bases de données.
- **PHP** comme langage de programmation backend. Notre choix est motivé par la facilité de déploiement sur des serveurs Web, sa robustesse, ainsi que sa présence bien établie dans le monde du développement Web.
- **HTML/CSS/JavaScript** pour la réalisation du frontend, il s'agit du standard utilisé pour la réalisation de pages web.
- **NGINX** comme serveur pour l'hébergement de l'application, en raison de sa facilité de déploiement et de ses performances.

Cette architecture correspond à une architecture classique pour la création de sites web interactifs nécessitant la gestion d'une base de données.

Pour une partie des membres de l'équipe, l'environnement de développement n'utilise pas le serveur web NGINX mais Apache, pour des raisons de facilité d'installation de WAMP Server [1] sur Windows.

L'application est compatible avec NGINX et Apache. Elle demande une version de PHP supérieure à 8.1.0 et utilise les extensions php-json, php-mysql et php-pdo.

Nous avons également déployé sur un serveur VPS la base de données MySQL. Travailler sur un seul et même serveur de développement, permet à toute l'équipe de disposer des mêmes données en temps réel.

5.2 Interface

L'interface utilisateur est accessible via un navigateur web à l'adresse suivante : `https://finance.examboost.be`.

L'application, déployée sur un serveur VPS, est constituée de différentes pages web, générées à l'aide de PHP.

5.2.1 Pages

Nous avons identifié et développé les différentes pages nécessaires au fonctionnement de l'application. Le tableau 5.1 résume les fonctionnalités associées à chaque page.

Page	**Fonctionnalités**
Connexion	Authentification des utilisateurs via email et mot de passe.
Accueil	Tableau de bord principal. Affiche la liste des portfolios de l'utilisateur et permet la création de nouveaux portfolios.
Paramètres utilisateurs	Modification des paramètres utilisateur (nom, prénom, mot de passe).
Vue Portfolio	Affiche la synthèse des actifs, la valorisation totale et les graphiques de performance.
Transactions	Historique complet des transactions (achats/ventes)
Gestion Transaction	Formulaire permettant d'ajouter ou de modifier une transaction.
Instrument	Vue détaillée d'un actif détenu : performance individuelle, quantité possédée et données de cotation.
Entreprise	Fiche d'information sur une société émettrice
Bourse	Fiche d'information sur une bourse d'échange
Paramètres	Configuration du portfolio, Gestions des membres.

Table 5.1 – Liste des pages et fonctionnalités principales de l'application

5.3 Implémentation

Plutôt que d'opter pour un framework complet tel que Laravel, nous avons opté pour une approche "from scratch". Cette approche est motivée par le fait que nous souhaitions apprendre et réaliser nous-mêmes la gestion de base de données. De plus, notre approche permet de réellement comprendre et maîtriser en profondeur le fonctionnement de l'application. Nous n'avons pas non plus de couche ORM (Object Relationnal Mapping) et avons réalisé les requêtes SQL nous-mêmes.

5.3.1 Architecture Backend

Routeur

Au lieu de multiplier les points d'entrée (un fichier PHP par page), nous avons centralisé l'intégralité du trafic via un fichier unique `index.php`. Ce point d'entrée unique instancie notre routeur [2] et distribue les

1. WAMP : Windows, Apache, MySQL, PHP (`https://www.wampserver.com/`)
2. inspiré du site WebReference.com

requêtes vers les bons fichiers de vue ou d'action.

Celui-ci nous permet, entre autre, de :

- Transformer des URLs dynamiques telles que `/portfolio.php?id=<id>` en `/portfolio/<id>`
- Grouper des routes (ex : toutes les routes commençant par `/portfolio/`)
- Utiliser des **Middleware** pour sécuriser l'accès à certaines routes (par exemple : `AuthMiddleware` vérifie si l'utilisateur est connecté avant d'autoriser l'accès à une route, et `CheckPortfolioAccess` vérifie s'il a les droits sur un portfolio spécifique).

L'entièreté du code de l'application ne se trouve pas dans le dossier racine du serveur web. Seul le dossier `public/` est servi par le serveur web. Dans celui-ci, nous retrouvons le dossier `assets/` avec les scripts, images et feuilles de styles utilisées par l'application et le fichier `index.php`. Ce dernier contient les définitions des routes de notre application et importe d'autres fichiers se trouvant dans le dossier parent du dossier racine web. Nous avons configuré le serveur Web pour renvoyer toutes les requêtes ne se trouvant pas dans le dossier `assets/` vers `index.php`. Le script s'occupe ensuite d'initialiser et de charger la route correspondante à l'aide de *Regex* (expressions régulières).

Nous avons procédé de la sorte afin de réduire les vecteurs d'attaque de notre application. En effet, un éventuel attaquant ne pourrait pas exécuter un script PHP quelconque et exploiter ses failles. Seul le fichier `index.php` est éxecuté et lui seul décide de quels autres scripts charger.

Sessions utilisateurs

Nous avons utilisé les *sessions* PHP pour sauvegarder l'état de connexion d'un utilisateur.

Lorsque l'utilisateur se connecte (Section 5.3.2), nous sauvegardons l'identifiant de l'utilisateur (email) dans la variable spéciale `$_SESSION`. PHP nous permet de faire persister de manière sécurisée, au travers d'un cookie spécial (PHPSESSID), cette variable pendant toute la visite du site par l'utilisateur.

L'utilisation de l'application nécessite à un utilisateur d'être connecté à son compte via son adresse mail. Afin de forcer la connexion avant d'accéder au reste de la plateforme, nous avons implémenté la classe `AuthMiddleware` qui assure la redirection des utilisateurs non authentifiés vers les pages de `login` ou `register`.

Droits d'accès

L'utilisation de Middleware sur le routeur permet la vérification des droits d'accès directement par groupe d'URLs, nous regroupons les URLs (utilisateur non-connecté, /portfolio/{portfolio_id}).

La requête SQL 5.1 montre un exemple de requête de droit d'accès réalisée à chaque chargement du portfolio. Dans cette requête, nous réalisons une jointure sur les tables `Membre_Portfolio` et `Utilisateur` par les clés `id_portfolio` et `email`, celle-ci nous permet de récupérer le niveau d'accès (1, 2 ou 3) associé à l'utilisateur dans ce portfolio. Si l'utilisateur n'a pas pas accès au portfolio, aucune ligne n'est retournée.

Requête SQL 5.1 – Vérification des droits d'accès au portfolio

```sql
SELECT Membre_Portfolio.niveau_acces
FROM Portfolio
    JOIN Membre_Portfolio ON Portfolio.id = Membre_Portfolio.id_portfolio
    JOIN Utilisateur ON Utilisateur.email = Membre_Portfolio.email
WHERE Utilisateur.email = ? AND Portfolio.id = ?;
```

Afin d'éviter les injections SQL, nous utilisons des requêtes préparées. Ainsi, les paramètres `"?"` sont automatiquement remplacés par les valeurs passées en argument lors de l'exécution de la requête SQL.

Cette requête permet de s'assurer qu'aucun utilisateur n'ayant pas les droits d'accès requis ne peut accéder aux pages du portfolio en lecture (1) ou en écriture (2) en fonction de ses droits d'accès. De plus, seul le propriétaire du portfolio (3) peut accéder aux pages de paramètres du portfolio.

5.3.2 Gestions des utilisateurs

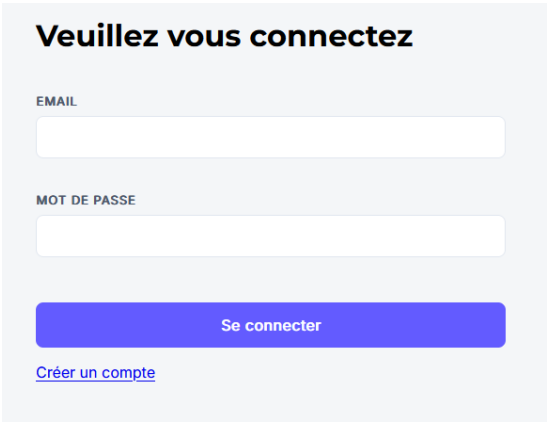
Nous avons implémenté trois pages dédiées à la manipulation des utilisateurs : `login` et `register`, qui gèrent la connexion et l'inscription des futurs utilisateurs ainsi que la page `update` qui permet à ces derniers de mettre à jour certaines de leurs informations personnelles.

Inscription et connexion

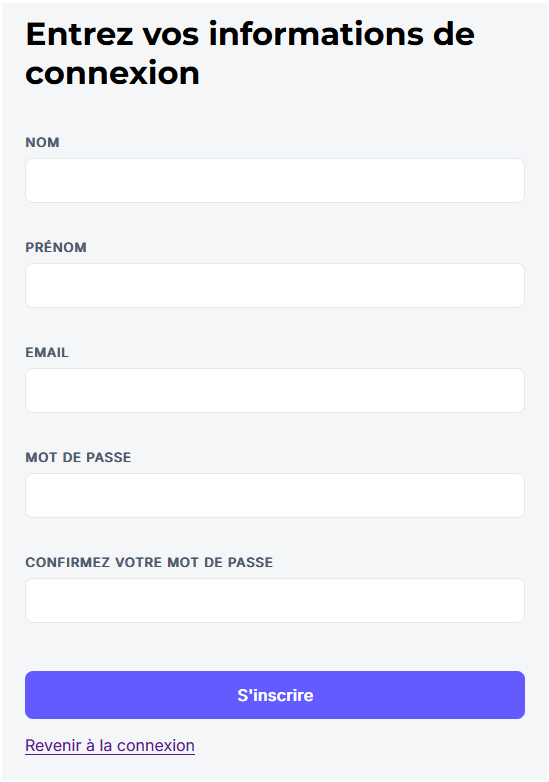
La figure 5.1a est l'interface de connexion sur laquelle sont redirigés automatiquement les utilisateurs non connectés. Sur celle-ci, il est possible de se connecter en inscrivant son adresse mail (identifiant unique) ainsi que son mot de passe.

Dans le cas d'une première utilisation de l'application, cette page est également liée à la page d'inscription visible sur la figure 5.1b. Là, un utilisateur référence pour la première fois ses informations personnelles.

Une vérification est effectuée sur chaque paramètre afin de forcer l'utilisateur à remplir tous les champs de manière valide (email validé par une `Regex`, mot de passe identique à la confirmation, aucun champ vide) et renvoie une erreur lorsque les contraintes ne sont pas respectées, sans supprimer les champs `Nom`, `Prénom` et `Email` pour plus d'ergonomie en cas d'erreur.



(a) Interface de connexion



(b) Menu de création de compte

FIGURE 5.1 – Gestion de connexion des utilisateurs

Une fois les vérifications de validité des informations effectuées, le mot de passe entré par l'utilisateur est haché grâce à la fonction `password_hash()` de `PHP`. Ensuite, l'utilisateur est inséré dans la base de données via la requête SQL 5.2. Ainsi, la base de données ne contient que le mot de passe haché, et aucun des membres ayant accès aux tables ne peut connaître le véritable mot de passe d'un utilisateur.

Requête SQL 5.2 – Insertion d'un nouvel utilisateur

```sql
INSERT INTO Utilisateur
(email, nom, prenom, mot_de_passe, date_creation)
VALUES (?, ?, ?, ?,  CURRENT_DATE())
```

Comme pour la première requête, et comme pour toutes les requêtes suivantes, l'utilisation des paramètres `"?"` permet d'éviter les injections `SQL`.

Modification d'un compte

Comme explicité plus tôt, une autre page permet aux utilisateurs de modifier leurs informations personnelles. Disponible depuis la page d'accueil (voir figure 5.2, coin supérieur droit), un bouton redirige l'utilisateur vers la page `update`.

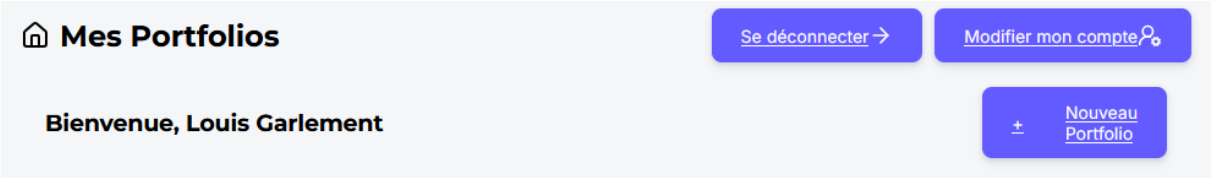
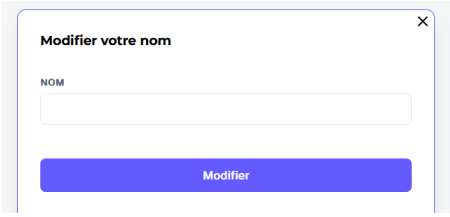


FIGURE 5.2 – Header de la page d'accueil, contenant un bouton d'accès à la page `update`

En arrivant sur cette page, trois options : la modification de son nom, prénom ou mot de passe. Cette dernière option requiert cependant d'encoder l'ancien mot de passe. Une vérification est alors à nouveau faite, et le nouveau mot de passe est également haché pour garantir la sécurité.

En cliquant sur l'un des trois boutons centraux visibles sur la figure 5.3a, une fenêtre pop-up s'ouvre (voir figure 5.3b) et permet d'entrer une nouvelle information (La pop-up pour modifier le mot de passe contient donc trois champs, un pour l'ancien, deux pour le nouveau).

(a) Interface de modification de compte



(b) Pop-up permettant la modification d'une information

FIGURE 5.3 – Menu de modification des informations personnelles d'un utilisateur

En appuyant sur `"Modifier"` dans la fenêtre pop-up, et après vérification de la conformité du champ entré par l'utilisateur, la requête SQL 5.3 est effectuée sur la base de données.

Requête SQL 5.3 – Modification d'un utilisateur

```sql
UPDATE Utilisateur
SET mot_de_passe = ?
WHERE email = ?
```

5.3.3 Gestion des portfolios

Sélection de portfolio

Une fois bien identifié l'utilisateur est transféré vers la page d'accueil. Sur celle ci apparaissent les portfolios auxquels cet utilisateur a accès, avec leurs noms, descriptions, valeurs totaux et les variations journalières de ceux-ci. Il y a aussi un bouton "Accéder au portfolio" pour aller dans le portfolio même. À noter que le fonctionnement de la page de portfolio choisi est expliqué dans le point 5.3.7

FIGURE 5.4 – Ensemble des portfolios

La recherche des portfolios de l'utilisateur particulier se fait par la requête SQL 5.6 :

Requête SQL 5.4 – Recherche des portfolios de l'utilisateur

```sql
SELECT
    p.id,
    p.nom,
    p.description,
    p.date_creation,
    d.symbole as devise
FROM Portfolio p
    JOIN Membre_Portfolio mp ON p.id = mp.id_portfolio
    JOIN Utilisateur u ON u.email = mp.email
    JOIN Devise d ON p.code_devise = d.code
WHERE u.email = ?
ORDER BY p.date_creation DESC
```

"?" est remplacé par le mail de l'utilisateur lors de l'exécution de la requête pour les raisons expliqués précédemment.

Création de portfolio

Il est évidemment aussi possible de créer un nouveau portfolio. Comme montré sur la figure 4 sur la page d'accueil il y a un bouton dédié. En appuyant déçu vous êtes invités à encoder les informations nécessaires pour la création d'un nouveau portfolio comme son nom, description et la devise principale. À noter qu'après la création le portfolio ne possède qu'un seul membre, vous. Pour en ajouter d'autres veuillez passer par la procédure expliqué au point *Modification du portfolio*

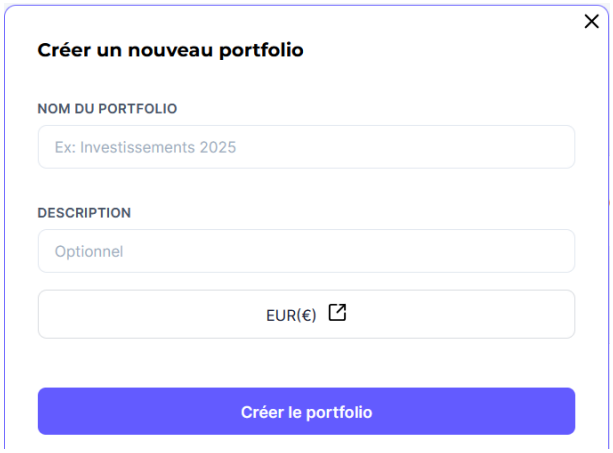


FIGURE 5.5 – Création de portfolio

La création de nouveau portfolio se fait par l'intermédiaire d'une requête SQL suivante. Les " ?" sont remplacés par nom, description et devise choisis respectivement.

Requête SQL 5.5 – Création de portfolio

```sql
INSERT INTO Portfolio (
    nom,
    description,
    code_devise,
    date_creation)
VALUES (?, ?, ?, NOW())

```

Modification du portfolio

Une fois sur la page de portfolio sélectionné son propriétaire peut procéder à la modification de ces paramètres en cliquant sur le bouton "Paramètres". À noter que ce bouton est visible seulement pour le propriétaire, mais bien sur néanmoins il y une vérification d'accès qui se fait lors de la tentative d'y accéder. La sécurité principale est évidemment présente du côté back-end et non front-end.

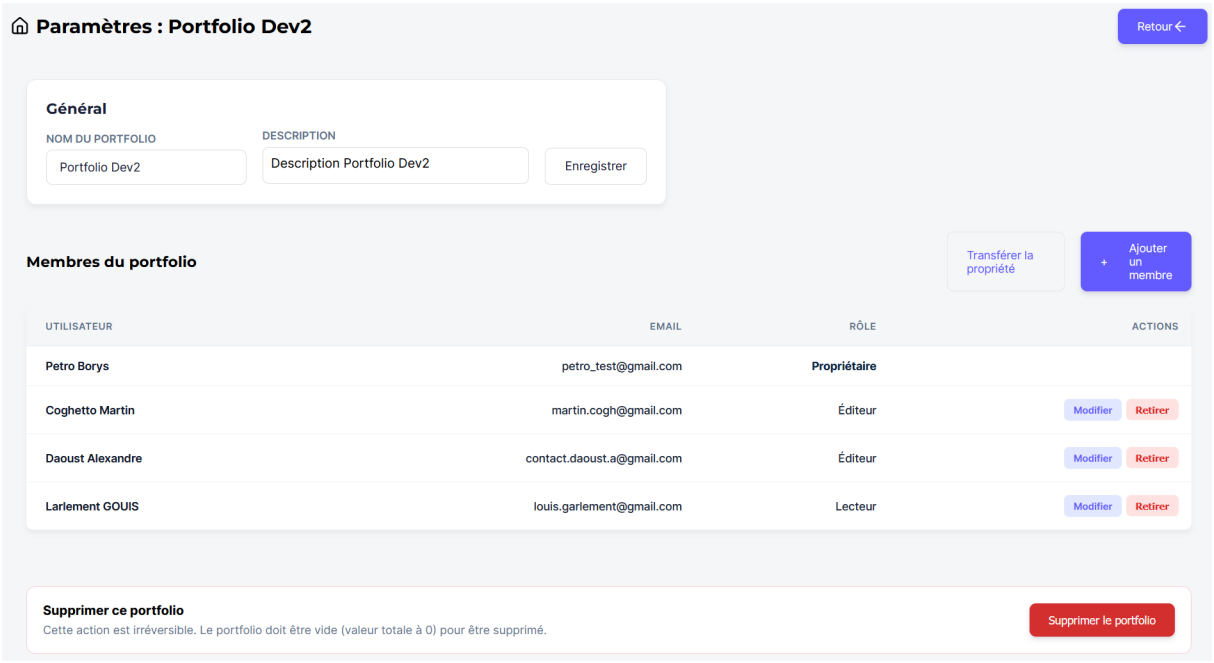


FIGURE 5.6 – Page des paramètres de portfolio

Une fois sur la page des paramètres le propriétaire peut modifier le nom du portfolio et son description, ainsi que gérer les membres de ce portfolio. Pour cela en bas de page est présente une liste des membres actuels avec la possibilité de les retirer ou modifier leur niveau d'accès. Il y a également une possibilité d'ajouter un nouveau membre par le bouton prévu au cet effet. Pour ajouter un nouveau membre il suffit de spécifier son email et son niveau d'accès initial. On prévoit aussi le moyen de transfert de la propriété à un membre de portfolio.Également on prévoit une possibilité de supprimer un portfolio à condition que sa valeur totale est nulle.

Mise à jour des droits d'accès se fait par une requête SQL basique où les trois " ?" sont remplacés par le nouveau niveau d'accès, le mail de l'utilisateur et id de portfolio respectivement.

Requête SQL 5.6 – Mise à jour des droits d'accès

```sql
UPDATE Membre_Portfolio
    SET niveau_acces = ?
WHERE
    email = ? AND id_portfolio = ?
```

5.3.4 Système CRUD

Nous avons regroupé la logique de gestion de chaque table de la base de données. Nous avons développé un système composé de fenêtres "pop-up" permettant de créer et d'ajouter de nouveaux enregistrements.

Du coté serveur, nous utilisons la classe abstraite `AffichageTable` définissant le squelette des opérations de lecture, création et modification.

Chaque entité (Transaction, Instrument, Bourse, etc.) possède son propre contrôleur qui implémente cette classe abstraite. Cette implémentation définit la logique spécifique à chaque modèle :

- L'affichage du formulaire de création/modification (`form($data)`)
- La logique de validation des données (`parse($input)`).
- Les requêtes SQL de recherche paginées (`search($params)`, `count($(count)`)
- Les requêtes SQL d'insertion et de modification (`insert($data)`, `update($id, $data)`))

Cette approche orientée objet nous permet d'uniformiser les fonctionnalités d'insertion/suppression et de recherche des entités.

Requête SQL 5.7 – Exemple de requête SQL d'insertion

```sql
INSERT INTO `Transaction`
(id_portfolio, isin, email_utilisateur,`type`, `date`, heure, quantite,
    valeur_devise_portfolio, taxes, frais)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

Requête SQL 5.8 – Exemple de requête de recherche d'entité

```sql
SELECT Transaction.*
FROM Transaction
    JOIN Instrument_Financier ON Instrument_Financier.isin = Transaction.isin
WHERE LOWER(Instrument_Financier.nom) LIKE CONCAT('%', :recherche, '%')
LIMIT :limit OFFSET :offset
```

La Requête SQL 5.8 montre un exemple de recherche de transaction par nom d'instrument financier dans la base de données. Nous utilisons la fonction de recherche `LIKE` combinée à une entrée '%<recherche>%'. Les symboles '%' permettent de rechercher n'importe quelle sous chaîne de caractères comportant notre chaîne <recherche>. Les paramètres `:recherche`, `:limit` et `:offset` sont automatiquement remplacés par leurs valeurs lors de l'éxécution de la requête PHP.

Interactivité et AJAX

Nous utilisons JavaScript pour intercepter la soumission des formulaires et communiquer avec le serveur via des requêtes **AJAX** (Asynchronous JavaScript and XML). Ceci permet d'éviter le raffraichissement des pages et rajoute une couche d'intéractivité à l'application.

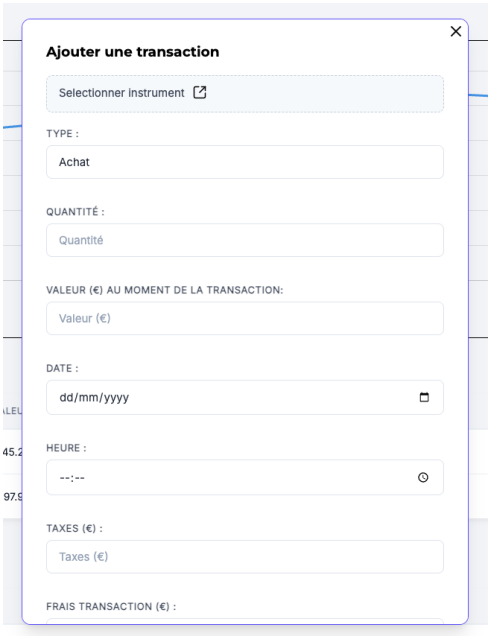
Le flux de données est le suivant :

1. L'utilisateur remplit le formulaire généré sur le serveur.
2. JavaScript capture l'événement `submit`, bloque le rechargement de la page et envoie les données en arrière plan au contrôleur PHP correspondant.
3. Le serveur valide les données via la méthode `parse`.
4. En cas d'erreur de validation, le formulaire est régénéré de manière dynamique avec la valeur des champs actuels et l'affichage des erreurs, et Javascript remplace le contenu HTML du formulaire par le nouveau.
5. En cas de réussite, JavaScript exécute le callback correspondant : fermer le "pop-up", rafraîchir la page, ...

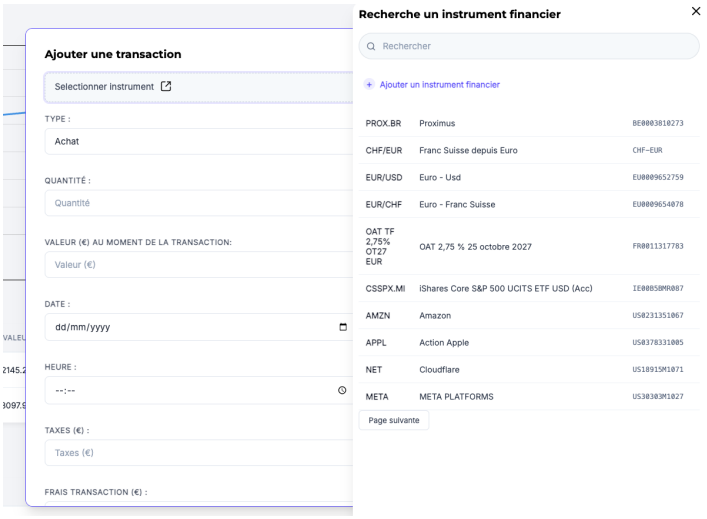
Gestion des Relations (Sélecteur Interactif)

Nous avons également imaginé un sélecteur interactifs pour la gestion des entités étrangères (clés étrangères). Au lieu de proposer un simple menu déroulant classique (select) à l'utilisateur, nous avons voulu proposer une solution plus adaptée à des listes potentiellement longues.

Ce composant permet, via une fenêtre "pop-up", de rechercher une entité liée dans la base de données ou d'en créer une nouvelle, sans quitter le contexte de création initial.



(a) Interface d'ajout de transaction



(b) Sélecteur interactif permettant la recherche et la liaison dynamique d'entités

FIGURE 5.7 – Gestions des entités

5.3.5 Données des cours de bourses

Récupération des données

Pour que l'application puisse répondre à nos besoins, il lui faut un moyen d'obtenir les données relatives aux instruments des utilisateurs. Pour ce faire, nous utilisons l'`API` de `YahooFinance` par le biais de la librairie `scheb/yahoo-finance-api`.

Nous avons décidé de récupérer automatiquement les données de tous les instruments présents dans la base de données avec un script `PHP` qui est appelé à un intervalle de 2 heures [3] par l'utilitaire `crontab` tel que suit :

```
0 */2 * * * curl "https://finance.examboost.be/donnees"
```

L'`API` que nous utilisons ne présente aucune garantie ni de limite de fréquence sur les appels que nous effectuons. De plus, nous ne pouvons être certains de l'exactitude des données rentrées par l'utilisateur. C'est pourquoi nous avons décidé d'ajouter une table supplémentaire dans l'application (ne faisant pas partie du mcd) permettant de mettre en cache les symboles des instruments (ce qui nous permet d'être plus sélectif sur les requêtes).

Fonctionnellement cette récupération de donnée est composée de plusieurs étapes.

Vérification du cache La première étape est de vérifier si les instruments possèdent déjà une entrée dans la table de cache. Ce cache, si égal à "MISS" signifie que cet instrument n'as pas su être récupéré durant une exécution précédente du script. Cet instrument sera alors ignoré lors des appels `API`. Si le cache est nulle, l'instrument n'as jamais encore été traité par le script. Autrement, le symbole présent dans le cache sera utilisé pour les requêtes.

Vérification des historiques Ensuite, nous récupérons la dernière date présente dans l'historique des instruments. Celle-ci peut être nulle, indiquant alors un historique vide (qui sera à récupérer).

Ces deux première étapes préliminaires ont été combinées en une seule requête `SQL` :

Requête SQL 5.9 – Requête SQL préliminaire

```sql
SELECT DISTINCT Instrument_Financier.isin, symbole, nom, MAX(Cours.date) as date,
    YahooFinanceCache.symbol as yahoo_symbol
FROM Instrument_Financier
LEFT JOIN YahooFinanceCache ON YahooFinanceCache.isin = Instrument_Financier.isin
```

3. Nous appelons le script toutes les 2 heures sur notre serveur.

```
LEFT JOIN Cours ON Cours.isin=Instrument_Financier.isin
GROUP BY Instrument_Financier.isin
```

Pour chaque tuple renvoyée par cette requête, nous effectuons un appel à l'`API` pour effectuer une recherche en utilisant de manière itérative l'isin, le symbole [4] et le nom si le symbole provenant du cache est nulle (autrement nous savons que nous pourrons effectuer une récupération d'historique sur l'instrument en utilisant le symbole du cache, nul besoin de le rechercher).

Le premier résultat de recherche est le symbole qui est alors enregistré dans le cache pour les futurs utilisations. Si aucun résultat n'est trouvé, il est impossible de garantir que l'`API` sera en mesure de retourner un historique pour l'instrument spécifié, nous enregistrons alors "MISS" comme symbole dans le cache.

Là est donc l'intérêt du cache, permettre de réduire le nombre d'appels de recherche fait à l'`API`. Dans le pire des cas c'est 3 requêtes qui sont effectuées par instrument !

Une fois cette étape préliminaire effectuée nous pouvons passer à la récolte des données à proprement parler.

Récupérations des données La récupération des données, vu la préparation des instruments, est un appel `API` direct. Pour chaque instrument ayant une clé non nulle et différente à "MISS", nous effectuons cet appel `API` en spécifiant le symbole du cache :

```php
// $client est le client api permettant d'effectuer les requêtes
$history = $client->getHistoricalQuoteData($instrument['yahoo_symbol'],
    ApiClient::INTERVAL_1_DAY,
    new DateTime($date. " +1 day 23:59", new DateTimeZone("UTC")),
    new DateTime('-1 day 23:59', new DateTimeZone("UTC"))
);
```

Pour le cadre du projet, nous ne récupérons que les données journalières étant dans l'intervalle de temps [$date, hier] avec $date étant soit le jour même l'année passée soit le lendemain de la dernière date de l'historique dans la base de données.

Autrement dit, si aucune donnée n'as encore été enregistrée dans la base de données, nous récupérons un an d'historique pour cet instrument.

Affichage des données

Un des besoins de notre application est de pourvoir visualiser les données les cours de ses instruments (et de ses portfolios).

Pour ce faire, nous avons utilisé `Chartjs` une librairie `Javascript` permettant de créer des graphiques sur une page web.

Tous les graphiques sont gérés et construit par un même script `Javascript` [5]. Le script permet de créer un graphique avec des attributs prédéfinis dans le code `html` de la page actuelle. C'est notamment le script qui va effectuer la requête vers le serveur pour récupérer les données d'un cours (ou d'un portfolio).

Le seul rôle du serveur est de remplir les champs des attributs `html` lors de l'affichage de la page et la récupération des données, ainsi que leur formatage pour pouvoir être correctement affichées.

Le serveur présente alors une url spécifique à la récupération des données formatées (/donnees/{type}/{isin}) où type est le type de donnée à récupérer (cours ou portfolio) et isin l'isin de l'instrument ou l'identifiant du portfolio (donc en fonction du type).

Cette url possède la propriété de restreindre l'accès aux utilisateurs connectés lorsqu'il s'agit de données relatives à un instrument et restreins l'accès aux utilisateurs ayant accès au portfolio lorsqu'il s'agit de données relatives à un portfolio donné.

Voici ce qu'il se passe techniquement lorsque l'utilisateur souhaite visionner le cours d'un de ses instruments :

4. Correspondant au symbole encodé par l'utilisateur
5. Ce script s'appelle `graph.js` dans le code

Chargement de la page de l'instrument L'utilisateur peut accéder à cette page via la page principale du portfolio ou l'instrument a été créer. Cette page va s'occuper de charger des données qui seront affichées dès l'envoi de la page et va alors préremplir les attributs `html` qui seront utilisés dans le script `Javascript` pour récupérer les données du cours et les afficher. L'élément dans la page qui va être utilisé pour effectuer le rendu du graphique est le suivant :

```
// $ins est l'instrument concerné
<canvas id="graph"
    data="<?= $ins["isin"] ?>"
    data-type="cours"
    currency="<?= $ins["devise"] ?>"
    label="<?= $ins["nom"]?>"
></canvas>
```

Récupération des données Une fois la page chargée par l'utilisateur, le script va s'exécuter. Il va utiliser les attributs susmentionnés pour récupérer les données.

```
// response est un dictionaire contenant:
// ['type': le type de graphique,
//  'data': les données formatées pour ce type de graphique]

// 'this' est une classe créée pour gérer les graphiques
let response = await fetch(`/donnees/${type}/${source}?range=${this.timeRange}`)
    .then(r => r.json());
```

Comme affiché dans la figure 5.8, l'utilisateur peut changer la durée des données affichées dans le graphique. L'utilisateur a le choix entre une semaine, un mois et un an.

FIGURE 5.8 – Graphique de l'action Amazon

Affichage des données agrégées pour un portfolio

Lorsque l'utilisateur arrive sur la page principale d'un de ses portfolios, il sera accueilli avec un graphique représentant la valeur totale de ses instruments au fil du temps. Celui-ci est construit de manière similaire aux graphiques relatifs au instruments eux-mêmes sauf qu'ici il est question d'agréger les données (faire la somme de la valeur des instruments).

L'agrégation de données se fait de telle sorte :

Récupération de la devise du portfolio Nous récupérons en premier lieu la devise du portfolio, cela nous permettra de comparer les devises des instruments avec celle-ci pour effectuer les conversions nécessaires.

Requête SQL 5.10 – Récupération de la devise du portfolio

```sql
SELECT p.code_devise as devise FROM Portfolio p WHERE p.id = ?
```

Récupération de devises de tous les instruments Nous récupérons ensuite toutes les devises distinctes des instruments pour pouvoir récupérer les cours des instruments et les sommer selon une certaine devise et appliquer la bonne conversion au résultat.

Requête SQL 5.11 – Récupération des devises constituantes du portfolio

```sql
SELECT DISTINCT(i.code_devise) as devise FROM Transaction t
    LEFT JOIN Instrument_Financier i ON i.isin = t.isin
    WHERE t.id_portfolio = ?
```

Agrégations de cours Comme cité ci-dessus, nous effectuons l'agrégation des données pour chaque devise différente. Ensuite, nous appliquons la conversion de divise [6] pour finalement agréger le tout pour pouvoir l'afficher.

Requête SQL 5.12 – Agrégation des données

```sql
SELECT c.date,
SUM(
    c.valeur_fermeture
    *
    CASE
        WHEN t.type = 'achat' THEN t.quantite
        WHEN t.type = 'vente' THEN -t.quantite
        ELSE 0
    END
) as valeur_fermeture
FROM (
    SELECT DISTINCT(i.isin) as isin
    FROM Portfolio p
    LEFT JOIN `Transaction` t on t.id_portfolio = p.id
    LEFT JOIN Instrument_Financier i on i.isin = t.isin AND i.code_devise = ?
    WHERE p.id = ?
    ) as f
LEFT JOIN Cours c on c.isin = f.isin
LEFT JOIN `Transaction` t on t.isin = f.isin AND t.date <= c.date AND t.id_portfolio = ?
WHERE c.date >= ?
GROUP BY c.date
```

La requête imbriquée de la requête 5.12 permet de récupérer la liste des `isin` des instruments ayant au minimum une transaction dans le portfolio. Cette liste est ensuite utilisée pour faire une jointure avec les cours de ces instruments et les transactions correspondantes dans le portfolio.

Les transactions sont filtrées par date. Pour chaque date dans l'intervalle sélectionné par l'utilisateur, nous récupérons toutes les transactions qui ont été faites à cette date ou avant pour pouvoir faire la somme sur la quantité d'un certain instrument présent dans le portfolio à cette date.

Ensuite, comme expliqué précédemment, nous effectuons la conversion monétaire si les devises sont différentes. Finalement, nous agrégeons les données (nous les sommons) pour les renvoyer à l'utilisateur pour les afficher.

La figure 5.9 montre un exemple de résultat final.

6. Dans notre implémentation finale, cette étape n'as pas été réalisée

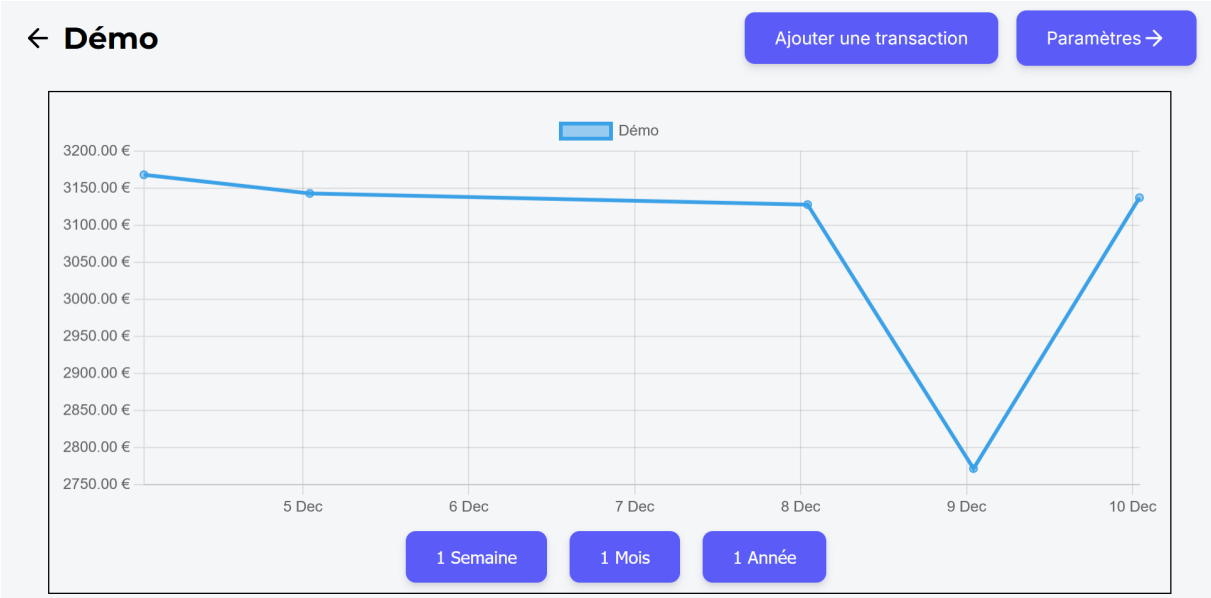


FIGURE 5.9 – Graphique d'un portfolio

5.3.6 Transactions

Les 3 dernières transactions du portfolio sont affichées sur la page principale de celui-ci (fig 5.10). Il est possible d'accéder à tout l'historique de transactions du portfolio en cliquant sur le bouton "Transactions", celui-ci redirige vers la page dédiés à la liste des transactions. L'affichage réutilise exactement la même logique que l'affichage l'historique des transactions (voir ci-dessous).

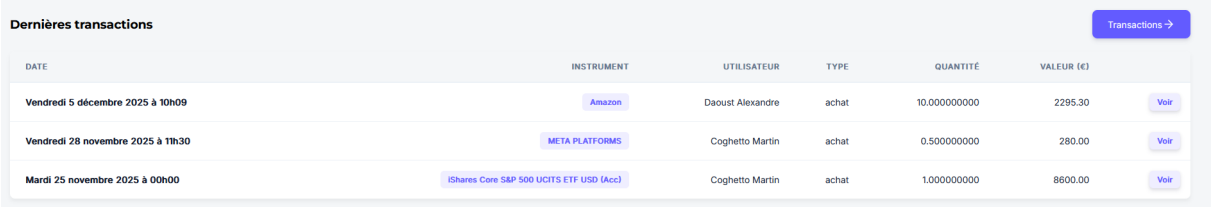


FIGURE 5.10 – Dernières transactions encodées dans le portfolio

Historique des transactions

La page d'historique des transactions permet de lister les transactions du portfolio. Celle-ci affiche une table avec les colonnes Date, Instrument (lien cliquable vers la page dédiée de l'instrument), l'utilisateur ayant encodé la transaction (nom et prénom), le type (vente ou achat), la quantité et la valeur (montant) de la transaction (figure 5.11).

Les résultats sont, par défaut, triés selon la date de transaction la plus récente. Il est possible de changer la colonne et l'ordre du tri (croissant ou décroissant) en cliquant sur la colonne correspondante.

Il est également possible de rechercher des transactions particulières en utilisant le champ de recherche. Celui-ci permet de filtrer par le nom de l'instrument financier.

Le tableau possède une pagination ; lorsqu'il y a plus de 20 transactions à afficher, celui-ci affiche automatiquement un bouton "page suivante" et "page précédente".

Toutes les interactions avec le tableau (tri, recherche, pagination) sont gérées par JavaScript en arrière-plan et ne requièrent pas de rafraîchissement de la page. Un TableHelper PHP permet de définir la forme du tableau (colonnes à afficher) et gère la validation du tri et de la pagination.

Requête SQL 5.13 – Requête d'historique de transactions

```sql
SELECT
    t.id,
    t.date,
    t.heure,
```

```
    CONCAT(u.nom, ' ', u.prenom) AS nom_utilisateur,
    t.type,
    t.quantite,
    i.nom AS nom_instrument,
    i.isin AS isin,
    ROUND(t.valeur_devise_portfolio * CASE WHEN t.type = 'achat' THEN 1 ELSE -1 END, 2)
    as valeur
FROM Transaction t
    JOIN Utilisateur u ON u.email = t.email_utilisateur
    JOIN Instrument_Financier i ON i.isin = t.isin
WHERE t.id_portfolio = ? AND i.nom LIKE CONCAT('%', ?, '%')
ORDER BY $orderBy $orderByType
LIMIT ? OFFSET ?
```

La requête SQL 5.13 permet de récupérer l'historique des transactions. Celle-ci réalise une sélection depuis la table Transaction, jointe à la table Utilisateur (par l'email utilisateur) et à la table Instrument_Financier (par isin, identifiant unique d'un instrument financier). Une sélection est réalisée pour récupérer uniquement les transactions liées au portfolio (via id_portfolio) ainsi que celles validant le nom de l'instrument financier entré par l'utilisateur. Un tri est réalisé sur la colonne demandée (`$orderBy`) ainsi qu'avec l'ordre spécifié (`$orderByType` : `asc` ou `desc`) [7]. Nous projetons ensuite les attributs suivants : l'id de la transaction, la date et l'heure de la transaction, le nom et prénom de l'utilisateur ayant encodé la transaction, le type de la transaction, la quantité concernée, le nom de l'instrument financier, l'isin de l'instrument financier ainsi que la valeur de la transaction (positive pour les achats et négative pour les ventes) arrondie à 2 unités après la virgule.

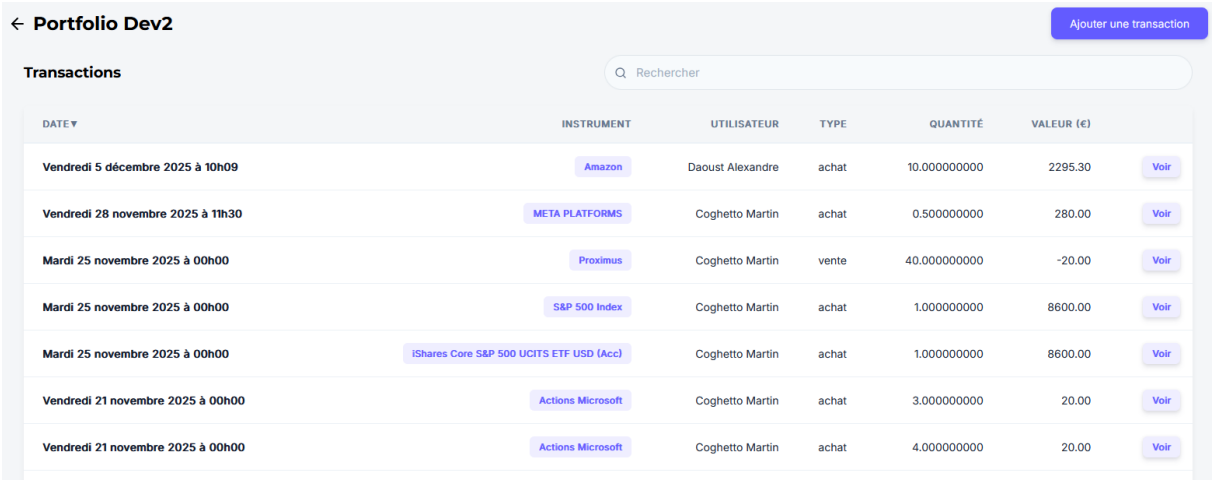


FIGURE 5.11 – Historique de transactions

Détails d'une transaction

Lorsqu'un utilisateur clique sur le bouton "Voir" d'une transaction, celui-ci est redirigé vers la page dédiée à cette transaction (figure 5.12).

Cette page affiche à l'utilisateur des informations supplémentaire sur la transaction tels que les frais et taxes encodés. Elle possède un lien renvoyant vers la page dédiée de l'instrument financier concerné par cette transaction.

7. Ces variables sont directement injectées dans la requête SQL, ce qui peut s'apparenter, en premier lieu, à une faille de sécurité (injections SQL). Cependant, le code PHP vérifie de manière très stricte la valeur des variables, et celles-ci ne peuvent prendre qu'un nombre fini de valeurs possibles. Il n'est donc pas possible de réaliser une injection SQL.

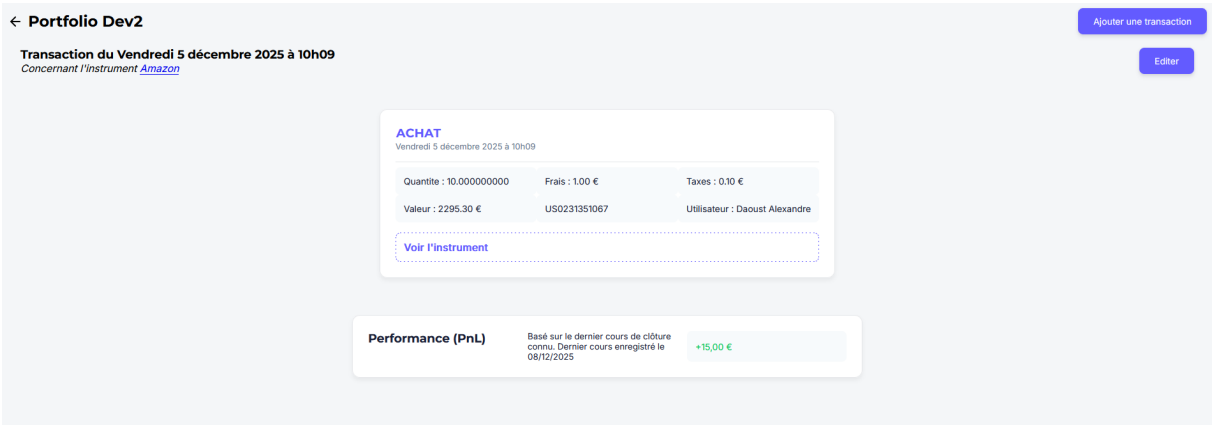


FIGURE 5.12 – Page transaction

Un bouton "Editer" permet de modifier la transaction dans le cas où des erreurs d'encodage seraient survenues. Celui-ci ouvre un popup (figure 5.13) permettant de modifier les informations de la transaction (Instrument Concerné, Type, Quantité, Valeur, ...). La logique utilisée est celle présentée à la section 5.3.4.

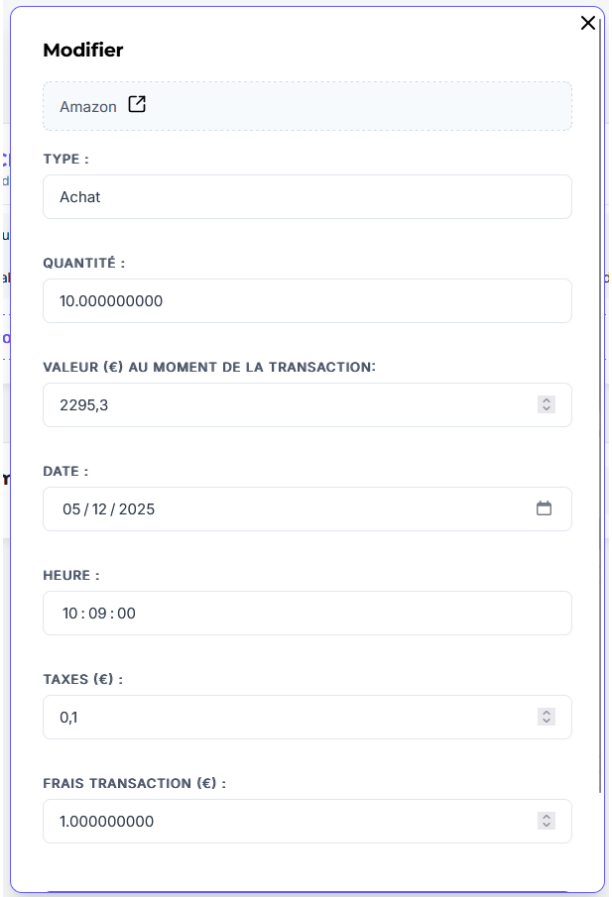


FIGURE 5.13 – Modifier une transaction

Dans le cas d'un achat, nous affichons également le PnL (Profit And Loses) de la transaction, c'est à dire la plus value latente de la transaction. Si la transaction a été achetée à un prix X, quel profit peut on obtenir en la vendant sur le marché financier à un prix Y aujourd'hui.

Nous utilisons la requête SQL 5.14 pour déterminer la dernière valeur à laquelle s'est vendue une transaction (Y) afin d'afficher son PnL.

Requête SQL 5.14 – Récupération du dernier cours de bourse pour un instrument financier donné

```sql
SELECT date, valeur_fermeture
```

```
FROM Cours Journalier
WHERE isin = ?
ORDER BY date DESC
LIMIT 1;
```

5.3.7 Instruments Financiers

La page d'accueil du portfolio affiche également les 3 instruments financiers détenus dans le portfolio ayant eu le plus de variations sur une journée (figure 5.14).

Nous considérons qu'un instrument financier est détenu dans le portfolio si la quantité totale de celui-ci est supérieur à 0. C'est à dire que toutes les transactions achats moins toutes les transactions ventes donnent une quantité positive. Un instrument financier peut donc disparaître du portfolio si il est acheté et vendu par la suite.

FIGURE 5.14 – Instruments Financier ayant le plus de variations

Le bouton "Contenu du portfolio" redirige vers la page dédiée au contenu du portfolio.

Contenu du portfolio

La page "contenu du portfolio" (figure 5.15) contient tous les instruments financier détenus dans le portfolio.

La table est interactive et peut être triée en cliquant sur les colonnes correspondantes. Par défaut, le tri s'effectue par ordre décroissant de la valeur totale possédée dans le portfolio.

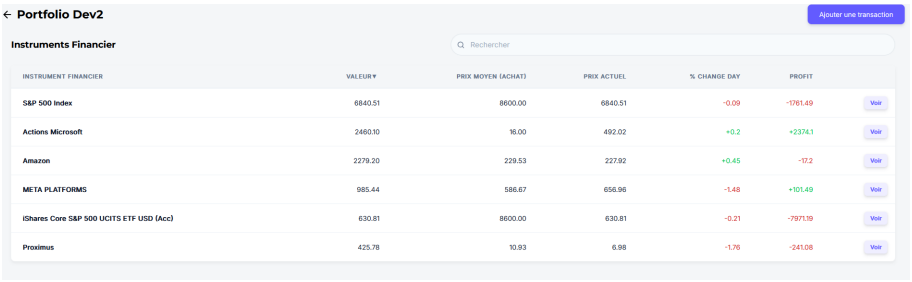


FIGURE 5.15 – Contenu du portfolio

La requête SQL 5.15 récupère le contenu du portfolio, c'est-à-dire la liste des actifs détenus ainsi que leurs performances financières. En raison de la complexité des calculs, nous avons structuré la requête à l'aide de deux *Common Table Expressions* (CTEs).

La première CTE, `Actifs`, isole la liste des instruments financiers présents dans l'historique du portfolio afin d'optimiser les jointures suivantes. La seconde, `ClassementCours`, utilise la fonction MySQL `ROW_NUMBER` pour attribuer un rang aux cotations boursières des 15 derniers jours par ordre décroissant de date. Cela nous permet d'identifier le cours le plus récent (rang 1) et le cours précédent (rang 2), même en cas de jours fériés ou de week-ends où la bourse était fermée [8].

Le cœur de la requête réalise ensuite une jointure entre les instruments financiers, l'historique complet des transactions (filtré par le portfolio cible) et nos CTEs de cours. Une agrégation est effectuée pour calculer la quantité totale détenue (`stock_qte`) et le montant total investi (`stock_investi`) en additionnant les

8. Cette partie de la requête SQL naïve équivalente était très lente. En effet, la complexité algorithmique pour récupérer le dernier et l'avant dernier jour de cotation disponible utilisant `select MAX()` et `select < MAX()` était de $O(n^3)$

achats et en soustrayant les ventes. Seuls les actifs dont la quantité détenue est strictement positive sont conservés via la clause `HAVING`.

Enfin, la projection finale calcule les indicateurs de performance clés :

- La **valeur actuelle** de la position.
- Le **prix moyen d'achat**, calculé en divisant le stock investi par la quantité.
- La **variation journalière** (`p_change`), représentant l'évolution en pourcentage entre le cours le plus récent et le précédent.
- Le **profit net** (ou plus-value latente), qui soustrait le montant investi, les frais et les taxes à la valeur actuelle.

Requête SQL 5.15 – Récupération du contenu du portfolio

```sql
SELECT
    isin,
    nom,
    ROUND(stock_qte * prix_ajd, 2) AS valeur,
    ROUND(prix_ajd, 2) AS prix_actuel,
    ROUND(stock_investi / stock_qte, 2) AS prix_moyen_achat,
    ROUND(((prix_ajd - prix_hier) / NULLIF(prix_hier, 0)) * 100, 2) AS p_change,
    ROUND((stock_qte * prix_ajd) - stock_investi - total_frais - total_taxes, 2) AS
    profit
FROM (
    WITH Actifs AS (
        SELECT DISTINCT isin
        FROM Transaction
        WHERE id_portfolio = ?
    ),
    ClassementCours AS (
        SELECT
            c.isin,
            c.valeur_fermeture,
            c.date,
            ROW_NUMBER() OVER(PARTITION BY c.isin ORDER BY c.date DESC) as rang
        FROM Cours c
        INNER JOIN Actifs a ON c.isin = a.isin
        WHERE c.date >= DATE_SUB(CURDATE(), INTERVAL 15 DAY)
    )
    SELECT
        t.isin,
        ins.nom,
        ajd.valeur_fermeture AS prix_ajd,
        hier.valeur_fermeture AS prix_hier,
        SUM(CASE
            WHEN t.type = 'achat' THEN t.quantite
            WHEN t.type = 'vente' THEN -t.quantite
            ELSE 0 END
        ) AS stock_qte,
        SUM(CASE
            WHEN t.type ='achat' THEN t.valeur_devise_portfolio
            WHEN t.type = 'vente' THEN -t.valeur_devise_portfolio
            ELSE 0 END
        ) AS stock_investi,
        SUM(COALESCE(t.frais, 0)) AS total_frais,
        SUM(COALESCE(t.taxes, 0)) AS total_taxes
    FROM Instrument_Financier ins
        LEFT JOIN Transaction t ON ins.isin = t.isin AND t.id_portfolio = ?
        LEFT JOIN ClassementCours ajd ON t.isin = ajd.isin AND ajd.rang = 1
        LEFT JOIN ClassementCours hier ON t.isin = hier.isin AND hier.rang = 2
    WHERE ins.nom LIKE CONCAT('%', ?, '%')
    GROUP BY t.isin, ins.nom, ajd.valeur_fermeture, hier.valeur_fermeture, ajd.date
    HAVING stock_qte > 0 ) s
ORDER BY $orderBy $orderByType
LIMIT ? OFFSET ?
```

Page détaillée d'un instrument financier

Chaque instrument financier possède sa propre page dédiée au sein d'un portfolio.

Cette page regroupe des informations liées à l'instrument financier (Symbole, nom, ISIN). Dans le cas d'une action et ETFs la bourse où elle est cotée et l'entreprise qu'elle concerne ; dans le cas d'une obligation, le pays qu'elle concerne ; et dans le cas des échanges de devises, les devises concernées.

Le graphique "candlestick" (graphique en chandelier) du cours de l'instrument financier est affiché sur une période définie par l'utilisateur (1 semaine, 1 mois, 1 an).

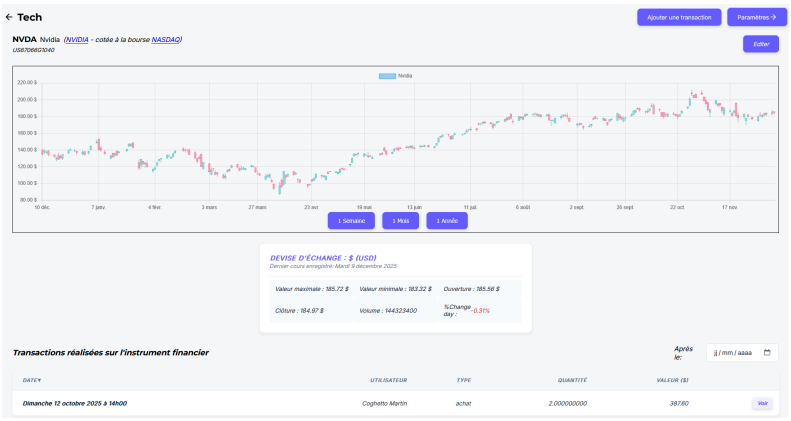
Des informations sur la dernière cotation enregistrée sont affichées : valeur maximale, valeur minimale, valeur à l'ouverture, valeur à la clôture, volume et %change day (variation en 1 jour).

La requête SQL 5.16 récupère les informations sur un instrument financier donné (isin). Celle-ci réalise une sélection depuis la table Instrument_Financier jointe à la table Devise (de l'instrument financier, via le code_devise), Portfolio (en ne sélectionnant que le portfolio concerné) ainsi que la Devise du portfolio. Une jointure à gauche, c'est à dire optionnelle, est réalisée pour récupérer l'Entreprise (via numero_entreprise et code_pays), la Bourse (via id_bourse) et le Pays (via code_pays). Nous utilisons une jointure à gauche car en fonction du type de l'instrument financier, ces relations peuvent ne pas être définie. Nous sélectionnons uniquement l'instrument financier concerné (via isin) et projetons les attributs suivants : isin, nom, symbole et type de l'instrument ; code et symbole de la devise de l'instrument ; le nom du portfolio ; le symbole de la devise du portfolio. Dans le cas des attributs nullables, nous récupérons : taux, date_emission, date_echeance pour les obligations ; le nom, le numéro, le code et le nom du pays de l'entreprise ainsi que l'id et le nom de la bourse dans le cas des actions et des ETFs ;

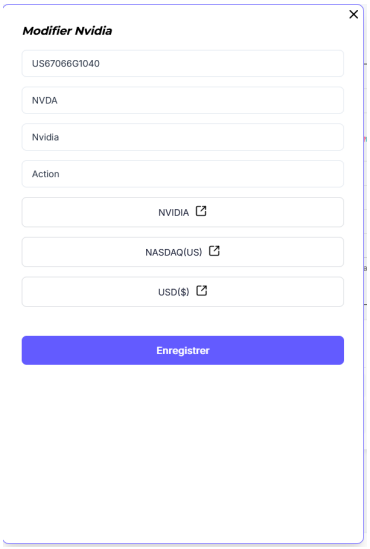
Requête SQL 5.16 – Information sur un instrument financier

```sql
SELECT
    ins.isin,
    ins.nom,
    ins.symbole,
    ins.taux,
    ins.date_emission,
    ins.date_echeance,
    di.code as code_devise,
    di.symbole as devise,
    p.nom as nom_portfolio,
    dp.symbole as devise_portfolio,
    ins.type,
    e.nom as nom_entreprise,
    b.id AS id_bourse,
    b.nom AS nom_bourse,
    CONCAT(e.code_pays, e.numero) as id_entreprise,
    pays.nom as pays,
    pays.code as code_pays
FROM Instrument_Financier ins
    JOIN Devise di ON di.code = ins.code_devise
    JOIN Portfolio p ON p.id = ?
    JOIN Devise dp ON dp.code = p.code_devise
    LEFT JOIN Entreprise e ON e.numero = ins.numero_entreprise AND e.code_pays = ins.
    pays_entreprise
    LEFT JOIN Bourse b ON b.id = ins.id_bourse
    LEFT JOIN Pays pays ON pays.code = ins.code_pays
WHERE isin = ?;
```

Un bouton éditer permet de modifier les informations de l'instrument financier concerné (figure 5.16b). Celui-ci utilise le système CRUD défini à la section 5.3.4.



(a) Page détaillée d'un instrument financier



(b) Modification d'un instrument financier

FIGURE 5.16 – Instrument financier

5.3.8 Entreprise

Une page permet d'afficher les détails d'une entreprise (figure 5.17a). Cette page affiche les instruments financiers contenus dans le portfolio liés à l'entreprise. Une entreprise peut en effet être liée à plusieurs instruments financiers : le cas où une entreprise émettrait plusieurs types d'actions ou encore la gestion de plusieurs ETFs différents.

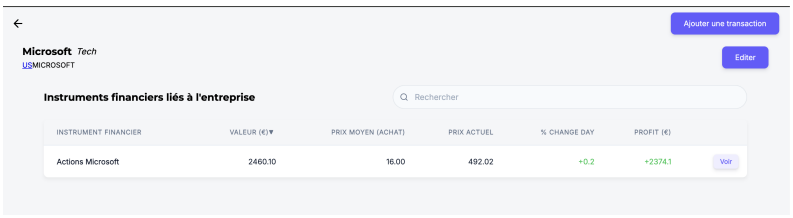
Comme pour les autres entités, un bouton "éditer" permet d'éditer l'entreprise (figure 5.17b).

Requête SQL 5.17 – Information sur une entreprise

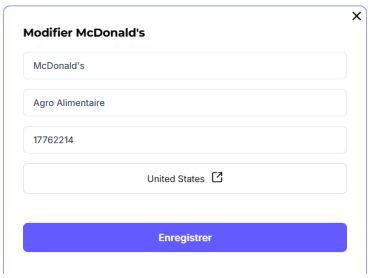
```sql
SELECT
    p.nom as nom_portfolio,
    e.*,
    ep.nom AS nom_pays
FROM Entreprise e
    JOIN Portfolio p ON p.id = ?
    JOIN Pays ep ON e.code_pays = ep.code
WHERE
    e.code_pays = ?
    AND e.numero = ?
```

La requête 5.17 permet de récupérer les informations d'une entreprise. Celle-ci sélectionne une entreprise donnée (code_pays, numero) et réalise une jointure sur Pays (via code_pays) et sur Portfolio (via l'id actuel du portfolio). Elle récupère toutes les informations de l'entreprise, le nom du pays ainsi que le nom du portfolio pour un affiche ultérieur.

L'affichage de la table des instruments financiers liés à l'entreprise est réalisé à l'aide d'une version légèrement modifiée (sélection de la requête en filtrant sur id_entreprise) de la requête présentée à la section 5.3.7.



(a) Page d'une entreprise



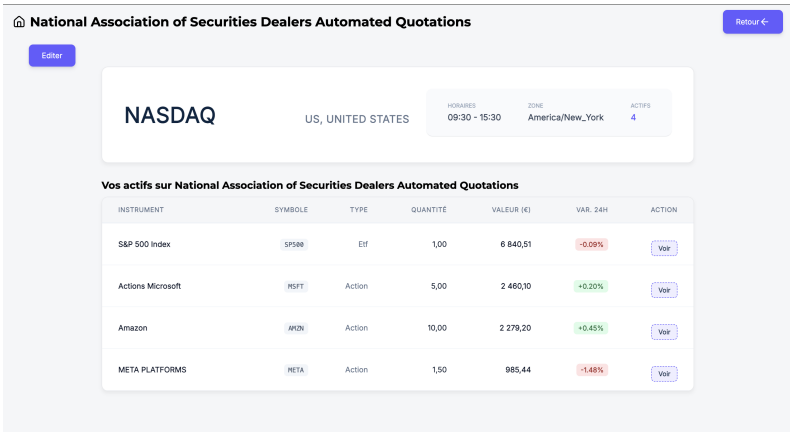
(b) Modifier une entreprise

FIGURE 5.17 – Entreprise

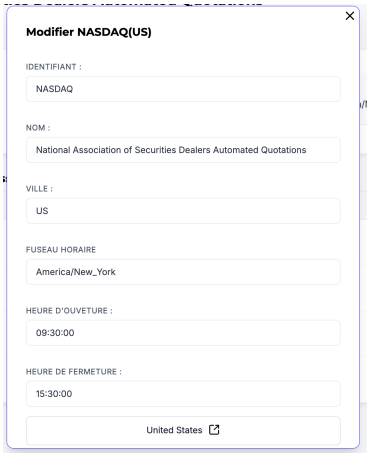
5.3.9 Bourses

Une page permet d'afficher les détails d'une bourse donnée (figure 5.18a) : nom, pays, horaire d'ouverture, fuseau horaire ainsi que les actifs financiers contenus dans le portfolio échangés sur cette bourse.

Celle-ci possède également un bouton éditer qui ouvre le "pop-up" d'édition (figure 5.18a).



(a) Page d'une bourse



(b) Modifier une bourse

FIGURE 5.18 – Bourse

6. Pistes d'amélioration

Bien que notre application soit fonctionnelle en l'état, il est évident qu'une multitude de changements pourraient améliorer la qualité et l'ergonomie de la plateforme, et ce dans chaque partie de l'application, ci-dessous se trouve ainsi une liste non exhaustive des potentielles améliorations à apporter à notre site.

6.1 Améliorations de l'interface / visuel

Ces propositions ont pour intérêt premier d'améliorer l'expérience utilisateur et l'utilisabilité de l'application.

Une première piste d'amélioration se situerait dans le design du site. En effet, l'esthétique n'étant pas le but de ce projet, nous avons opté pour un design très basique. Bien que suffisant, ce dernier pourrait toutefois être amélioré. Nous n'avons pas non plus considérés une utilisation sur téléphone portable du site, celui-ci ne possède pas de design "responsive" et son utilisation sur petit écran n'est pas optimale.

Nous pourrions ajouter une confirmation visuelle lors du bon fonctionnement d'une action (petit pop-up de checkmark vert) telle que la modification de son compte, la création d'un portfolio, l'encodage, etc.

Lorsque l'on se trouve sur la page d'accueil, les boutons de déconnexion et de modification peuvent sembler être de trop. Une solution serait de créer un menu "paramètres" plus discret et d'y déplacer ces boutons, permettant de centraliser les options liées au compte dans un même menu et d'alléger l'interface.

Il serait d'abord possible de laisser l'utilisateur modifier des paramètres personnels comme la possibilité d'utiliser l'application en thème sombre, et d'ajouter ces paramètres au menu évoqué plus tôt pour la déconnexion/modification.

6.2 Améliorations du backend / fonctionnalités

Ces suggestions sont cependant des fonctionnalités manquantes, ces dernières n'étaient pas spécialement nécessaires au développement et au lancement de la plateforme, mais peuvent être sollicitées par des utilisateurs fréquents.

Tout d'abord, la gestion des conversions de devises pourrait être améliorée. Dans certains cas, les conversions de devises sont mal gérées en raison des différences de fuseaux horaires dont nous ne tenons compte que partiellement. Dans le cadre de ce projet de base de données, nous considérons qu'il appartient à l'utilisateur de vérifier la cohérence de ses données (pas de portfolio multi-devises). Mais cela pourrait être correctement implémenté à l'avenir.

Ensuite, il pourrait être pratique pour une personne/entreprise gérant une grande quantité de portfolios de pouvoir épingler un ou plusieurs d'entre eux afin de ne jamais avoir à chercher ces derniers.

Il n'existe actuellement aucune manière de réinitialiser son mot de passe en cas d'oubli ou de perte. Cette fonctionnalité est assez cruciale, puisque créer un nouveau compte pour y encoder à nouveau chaque transaction devient très vite irréalisable. Notre infrastructure actuelle ne nous permet pas d'envoyer des mails (rendant la récupération de la sorte impossible) et le développement n'a pas connu de problème de la sorte, mais l'ajout de cette fonctionnalité pourrait prévenir d'une potentielle perte de temps pour des utilisateurs malchanceux.

7. Conclusion

Ce projet a été une opportunité de mettre en œuvre les concepts théoriques appris au cours de modélisation des données. Nous avons pu appliquer, dans un cas concret de gestion de portfolios financiers, les différents points de modélisation des données abordés au cours : cahier des charges, modèle conceptuel de données, modèle physique de données, modèle logique de données et l'interrogation de bases de données à l'aide du langage SQL.

En plus des concepts théoriques, nous avons également réalisé des recherches sur la partie métier de l'application : la gestion de portefeuilles en finance. Ces concepts étaient méconnus d'une partie des membres de l'équipe qui ont dû se former et découvrir une partie du monde de la finance. Pour les membres ayant déjà des connaissances dans ce domaine, ce projet fut l'occasion d'approfondir leurs connaissances sur le sujet. En effet, la modélisation d'un domaine particulier requiert une connaissance très pointue dudit domaine à modéliser.

Étant nos propres clients, nous avons pu réaliser un cahier des charges qui répondait réellement à nos besoins. Nous avons identifié les fonctionnalités manquantes dans les solutions existantes : centralisation, gestion multi-utilisateurs, historique des achats et des ventes, statistiques en vue de réaliser une analyse fonctionnelle des besoins. Nous avons défini les limites de l'application : ce que nous ne souhaitons pas implémenter dans l'application. Nous avons également réalisé une évaluation des coûts opérationnels et de développement tout en réalisant un partage des tâches au sein du groupe axés sur les différents domaines métiers de l'application. Nous avons défini les échéances ainsi qu'une planification temporelle des différentes étapes du projet.

Une fois notre MCD normalisé, le passage au modèle logique et physique des données s'est réalisé sans encombre : nous avons suivi les règles logiques de transformation. Nous avons réalisé notre MPD directement en SQL, ce qui nous a permis de définir les types de données ainsi que le schéma de notre base de données en une seule fois.

La phase la plus longue du projet fut l'implémentation. Nous avons utilisé PHP/MySQL. Cette partie nous a permis d'apprendre le langage PHP ainsi que le fonctionnement plus avancé du développement web : HTML, CSS, JavaScript. Nous avons également géré une base de données MySQL et déployé notre application directement sur un serveur (accessible à l'adresse : `https://finance.examboost.be`).

En conclusion, ce projet nous a permis d'appliquer les concepts théoriques vus au cours afin de réaliser un projet de gestion de base de données complet.

8. Manuel d'utilisation

Afin d'utiliser la plateforme, rendez-vous à l'adresse suivante : `https://finance.examboost.be/login`.

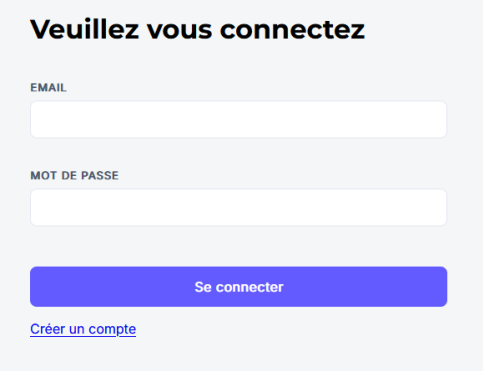


FIGURE 8.1 – Page "`login`"

Lors de votre première connexion, cliquez d'abord sur le lien hypertexte "`Créer un compte`", vous serez redirigé vers la page d'inscription décrite plus tôt, entrez vos informations personnelles et cliquez sur "`S'inscrire`".

Une fois inscrit, vous serez amené sur la page principale de l'application sur laquelle figureront vos portfolios. Bien sûr, aucun portfolio ne sera présent lors de la création de votre compte. Pour en ajouter un, cliquez sur le bouton "`Nouveau Portfolio`" visible dans le coin supérieur droit de la figure 8.2.

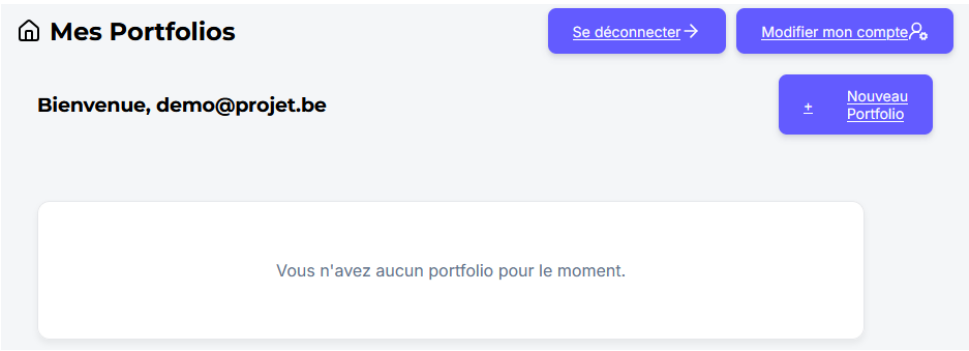


FIGURE 8.2 – Page "`home`"

Cliquer sur ce bouton ouvre une fenêtre pop-up au centre de votre écran (figure 8.3a). Vous pouvez alors nommer et donner une description à votre portfolio, ainsi qu'en choisir la devise via un menu s'ouvrant à droite (figure 8.3b). Si la devise que vous souhaitez utiliser pour le portfolio n'apparait pas déjà dans la liste des devises présentes, vous pouvez l'ajouter en cliquant le lien hypertexte "`Ajouter une devise`". Ceci ouvrira un autre menu (figure 8.3c) dans lequel vous devrez encoder le code (`USD, CHF, EUR, ...`), le nom (`Dollar américain, Franc Suisse, ...`) ainsi que le symbole (`$, €`) de la devise.

(a) Pop-up d'ajout de portfolio

(b) Sélecteur de devise

(c) Menu d'ajout de devise

FIGURE 8.3 – Création d'un nouveau portfolio

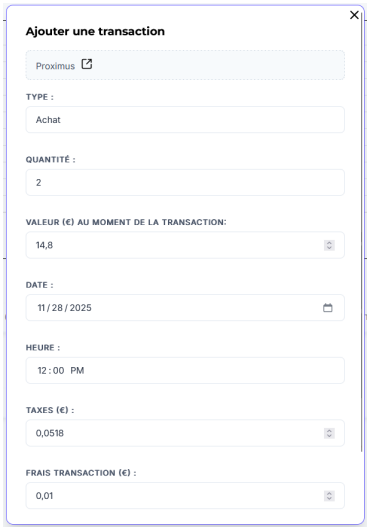
En revenant sur la page d'accueil (via la flèche de retour située dans le coin supérieur gauche de l'écran), votre nouveau portfolio, ainsi que sa description, et sa valeur actuelle exprimée dans la devise choisie. Vous pouvez ainsi accéder au portfolio avec le bouton dédié sur l'interface.

FIGURE 8.4 – Affichage du portfolio créé depuis la page `"home"`

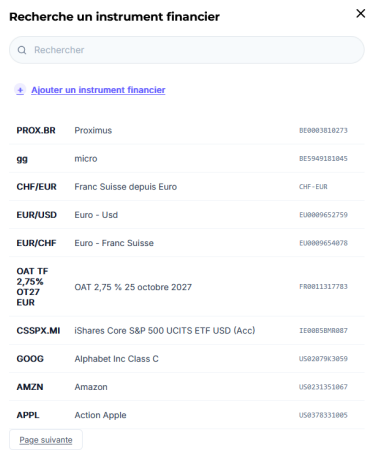
Une fois entré dans votre portfolio, vous pouvez alors consulter vos actifs et statistiques, soit actuellement aucune donnée. Pour y remédier, cliquez sur le bouton `"Ajouter une transaction"`. Ceci ouvrira une nouvelle fenêtre pop-up (figure 8.5a) sur laquelle vous pouvez encoder entièrement une transaction en précisant s'il s'agit d'un achat ou d'une vente (type), la quantité représentée dans la transaction ainsi que la valeur correspondant à la quantité (dans la bonne devise) au moment de l'achat ou de la vente. Ce moment est renseigné par la date et l'heure. Enfin, encodez le montant des taxes et frais de transactions avant de cliquer sur `"Enregistrer"`.

A nouveau, si l'instrument financier que vous souhaitiez encoder n'était pas déjà présents dans la liste des instruments disponibles, vous pouvez l'ajouter vous-mêmes via un menu similaire à l'ajout de devises (figure 8.5c). Dans ce dernier figurent alors de nouveaux sélecteurs d'entreprise, bourse et devise d'échange, que vous pouvez également ajouter si vous ne trouvez pas votre bonheur dans les quelques éléments présents.

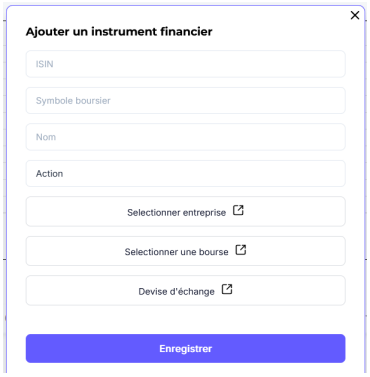
L'exemple de la figure 8.5a utilise les valeurs correspondants réellement à l'index de l'instrument `Proximus`, afin d'ensuite visualiser le comportement de cet investissement au fil du temps.



(a) Pop-up d'ajout de transaction



(b) Sélecteur d'instrument financier



(c) Menu d'ajout d'un instrument financier

FIGURE 8.5 – Ajout d'une nouvelle transaction

Une fois votre première transaction encodée, votre portfolio sera bien plus détaillé. Comme visible sur la figure 8.6, un graphique représente vos actifs sur un délai modifiable d'une semaine, un mois ou un an. En addition à cela, deux menus vous renseignent sur la variation de vos instruments financiers et sur vos dernières transactions en indiquant le taux de change, le profit, le prix à l'achat et actuel, etc. Et les boutons relatifs à ces catégories permettent d'afficher le contenu total du portfolio ou l'historique de toutes les transactions.

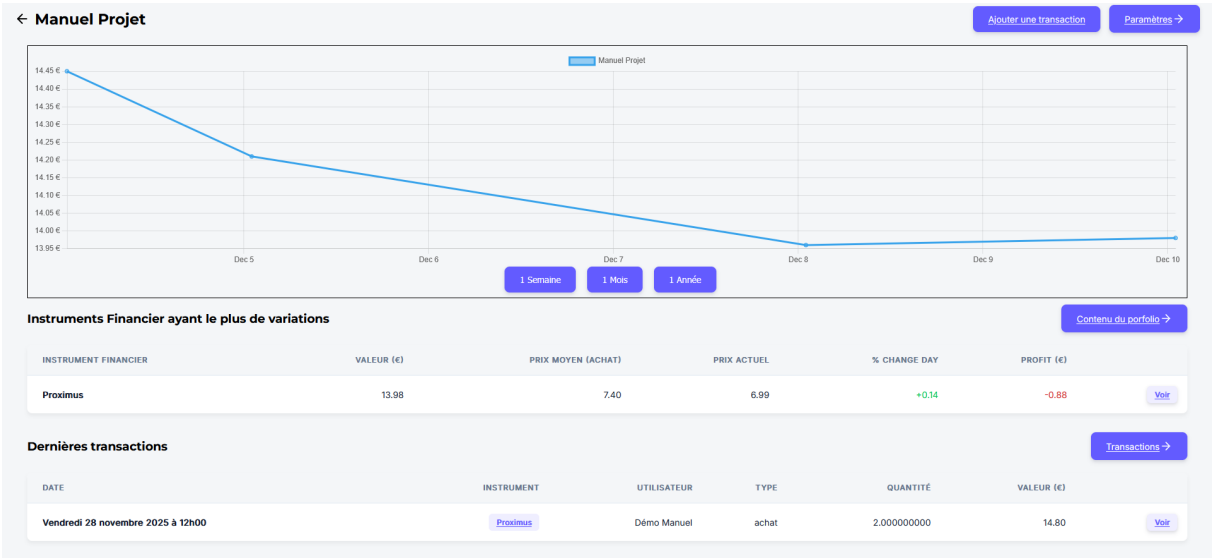
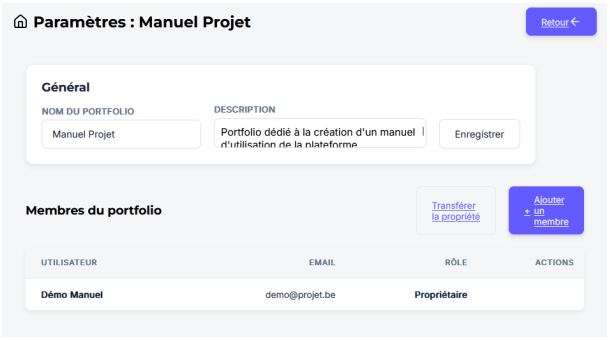
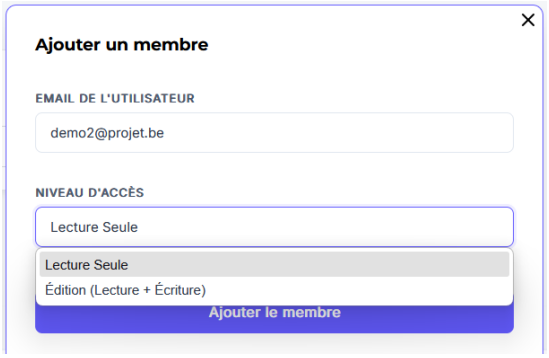


FIGURE 8.6 – Votre portfolio après l'ajout d'une première transaction

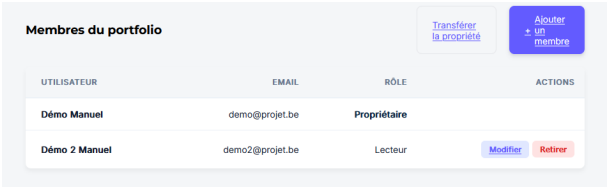
Chaque portfolio comporte également son propre menu de paramètres (figure 8.7a), afin de pouvoir en changer le nom et la description, de pouvoir y ajouter un membre (figures 8.7b et 8.7c), ou de transférer le droit d'administrateur à un autre membre (figure 8.7d).



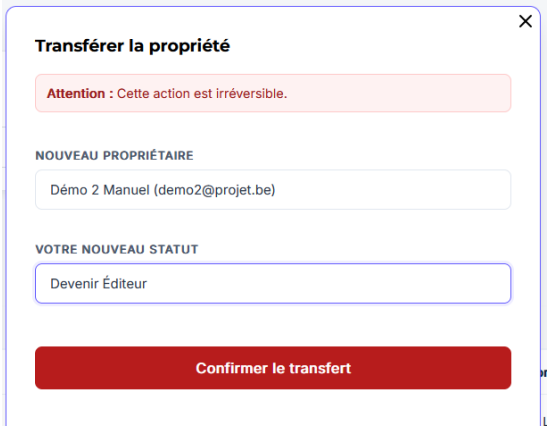
(a) Menu des paramètres de votre portfolio



(b) Ajout d'un second utilisateur en mode lecture seule



(c) Affichage de tous les membres du portfolio



(d) Transfert de propriété au second membre

FIGURE 8.7 – Paramètres du portfolio

Enfin, si vous souhaitez modifier les informations personnelles relatives à votre compte, un bouton présent dans le coin supérieur droit de la page d'accueil vous redirige vers la page dédiée à la mise à jour d'un utilisateur (figure 8.8). Vous pouvez y modifier votre nom, prénom et mot de passe, ce dernier n'étant possible qu'en connaissant votre ancien mot de passe.

FIGURE 8.8 – Page "`update`" permettant la modification de votre compte

Ce manuel d'utilisation touche à sa fin, il est alors temps de cliquer sur le bouton que ce tutoriel n'a pas encore couvert, vous ramenant sur la page de login : "`Se déconnecter`".