

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the text [Date].

[Date]

Rapport Projet JEE

À l'attention de Besma Zeddini

Several thin, curved, light blue lines that sweep upwards from the bottom left towards the center of the page.

Florian Briand, Erwan Bouvart, Raphael Coutinho,
Emmanuel Cousin

CY TECH

Objectif

Attirés par le monde de la finance, nous nous sommes fixés comme objectif de réaliser un site de gestion d'argent et de compte en bourse.

Spécifications des besoins/Fonctionnalités

Le site doit permettre de d'ouvrir différents types de comptes comme des comptes courants, des enveloppes fiscales comme des assurances vies ou des PEAs.

Cette application doit permettre à l'utilisateur de gérer différentes « enveloppes » d'argent.

Chaque enveloppe d'argent possède différentes caractéristiques, certaines sont attribuées des comptes en bourse d'autre à des comptes courants.

L'accès aux comptes, ce fait se connectant via une page de login qui nous emmène sur une page Dashboard, où l'on peut gérer l'argent des différents enveloppes/comptes.

Le site permet à un utilisateur de s'inscrire. Il permet à chaque utilisateur de souscrire à des nouvelles enveloppes.

Le but principal du site est de pouvoir réaliser des opérations sur nos comptes, faire des virements à partir d'un compte courant et d'acheter des actions via des comptes en bourse comme un PEA ou une assurance vie.

Conception

Nous avons clarifié nos fonctionnalités durant la conception et définie ce qui nous semblait réalisable. Nous avons effectué nos recherches sur les technologies à utiliser et leur complexité tout en restant avec les Framework demandé pour la réalisation du projet. Nous nous sommes finalement dirigés vers le Framework Spring Boot, avec Maven comme gestionnaire de projet.

Pour le lien avec notre base de données SQL, nous avons utilisé l'interface JPA qui est une implémentation du Framework Hibernate.

Pour afficher, les données de la BDD à l'HTML, nous avons utilisé le moteur de templates : Thymeleaf.

Pour notre front, nous avons utilisé la librairie Bootstrap en version 5.1.3, ainsi que du CSS et Javascript.

Afin de récupérer toutes les données liées aux actions en temps réel, notamment le prix des actions, nous avons utilisé l'API : Financial Modeling Prep.

Nous avons utilisé IntelliJ et Visual Studio Code comme IDE, accompagné du gestionnaire de version Git et Workbench comme gestionnaire de BDD.

Pour ce projet, l'une des missions était de créer une sécurité pour le site. Nous avons donc travaillé avec Spring Security. La première chose à faire avec cette dépendance est de définir une méthode « configure » qui prends en paramètre un `HttpSecurity`. Cela va permettre de traiter toute les requêtes http.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .csrf().disable()
        .authorizeRequests()
        .antMatchers( ...antPatterns: "/css/**", "/js/**", "/images/**").permitAll()
        .anyRequest()
        .authenticated()
        .and()
        .formLogin()
        .loginPage("/login")
        .permitAll()
        .defaultSuccessUrl("/index")
        /* .failureUrl("/error") permet de redirigé vers une page erreur si un
        .and()
        .logout()
        .invalidateHttpSession(true)
        .permitAll();
}
```

Méthode configure avec paramètre `HttpSecurity`

Il faut savoir qu'avec la dépendance Spring Security l'application web ne sera pas accessible. En effet, en la démarrant on arrivera sur une page de connexion basique (on peut définir les logs et les mots de passe dans le `application.properties`). Avec la méthode `configure` on décide quelle page on veut afficher avant de pouvoir se connecter et on peut mettre le lien d'une page de login « faites à la main » plutôt que d'afficher celle par défaut (qui reste très basique).

Pour ce qui est de la connexion avec une base de donnée, on a créé une seconde méthode `GlobalConfig` qui permet de faire le lien avec la base de données en utilisant JDBC (java

database connector). Dans cette méthode on dit ce que l'on veut aller chercher dans la base de données pour se connecter, on indique aussi le rôle que possède la personne en allant le chercher dans une autre table de la base de données. Dans cette méthode, on entre en paramètre un authenticationmangerbuilder qui fait comprendre à la méthode que nous « parlons » des personnes qui peuvent se connecter.

```
@Autowired
public void configure( AuthenticationManagerBuilder auth, DataSource dataSource) throws Exception{
    auth.jdbcAuthentication()
        .dataSource(dataSource)
        .usersByUsernameQuery("select prenom as principal, mdp as credentials, true from utilisateur where prenom= ?")
        .authoritiesByUsernameQuery("select utilisateur_idutilisateur as principal, roles_role as role from role_utilisateur where utilisateur_idutilisateur= ?")
        .rolePrefix("ROLE_");
}
```

Méthode Globalconfigure avec paramètre authenticationmangerbuilder

Cela a été l'une principale activité durant le projet, hélas, plusieurs problèmes de type SpringsecurityFilter sont apparus et ont rendu l'utilisation de Spring Security impossible. Pour réaliser la connexion au site, nous avons donc fait appel à un Controller, beaucoup moins sécurisé.

```
@PostMapping(path="/connexion")
public String connexion(Utilisateur utilisateur){
    System.out.println(utilisateur);

    boolean test = utilisateurRepository.existsUtilisateurByMdpEqualsAndAndNomAndAndPrenom(utilisateur.getMdp(), utilisateur.getNom(), utilisateur.getPrenom());
    System.out.println(test);
    if (test) {
        return "index";
    }
    else {
        return "login";
    }
}
```

Controller de connexion

Architecture

Accès à la bdd dans src/main/ressources/application properties :

spring.datasource.username=\$

spring.datasource.password=\$

Le script bdd est bddprojet.sql

Pour l'architecture du site web, nous avons utilisé le modèle-vue-contrôleur. On a le modèle qui contient les données à afficher. La vue contient la présentation de l'interface graphique et le contrôleur qui contient la logique concernant les actions effectuées par l'utilisateur.

Tout d'abord, nous avons défini un diagramme de classe très complet. A partir duquel nous avons à développer toutes les fonctionnalités de notre site, afin de la facilité notre Programmation Orienté Object.

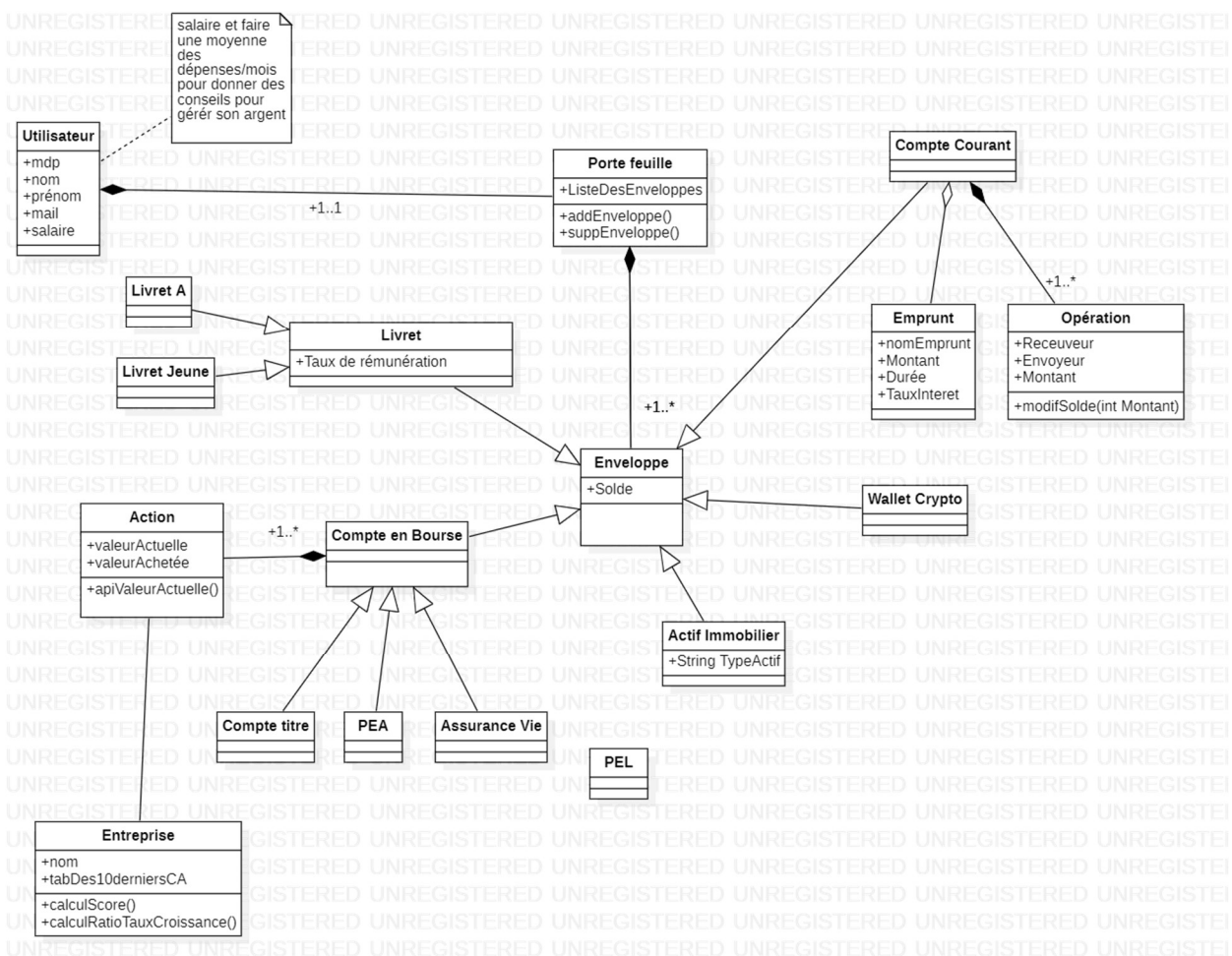
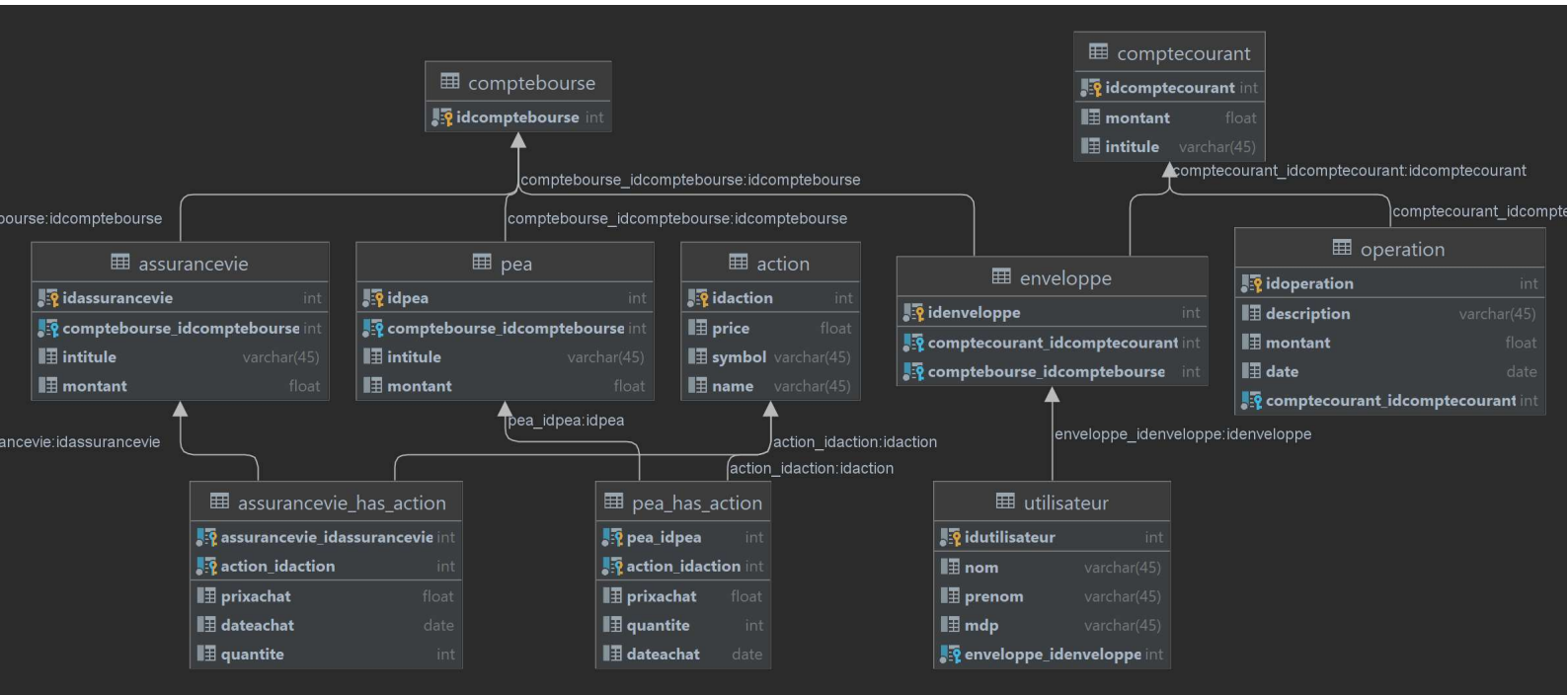


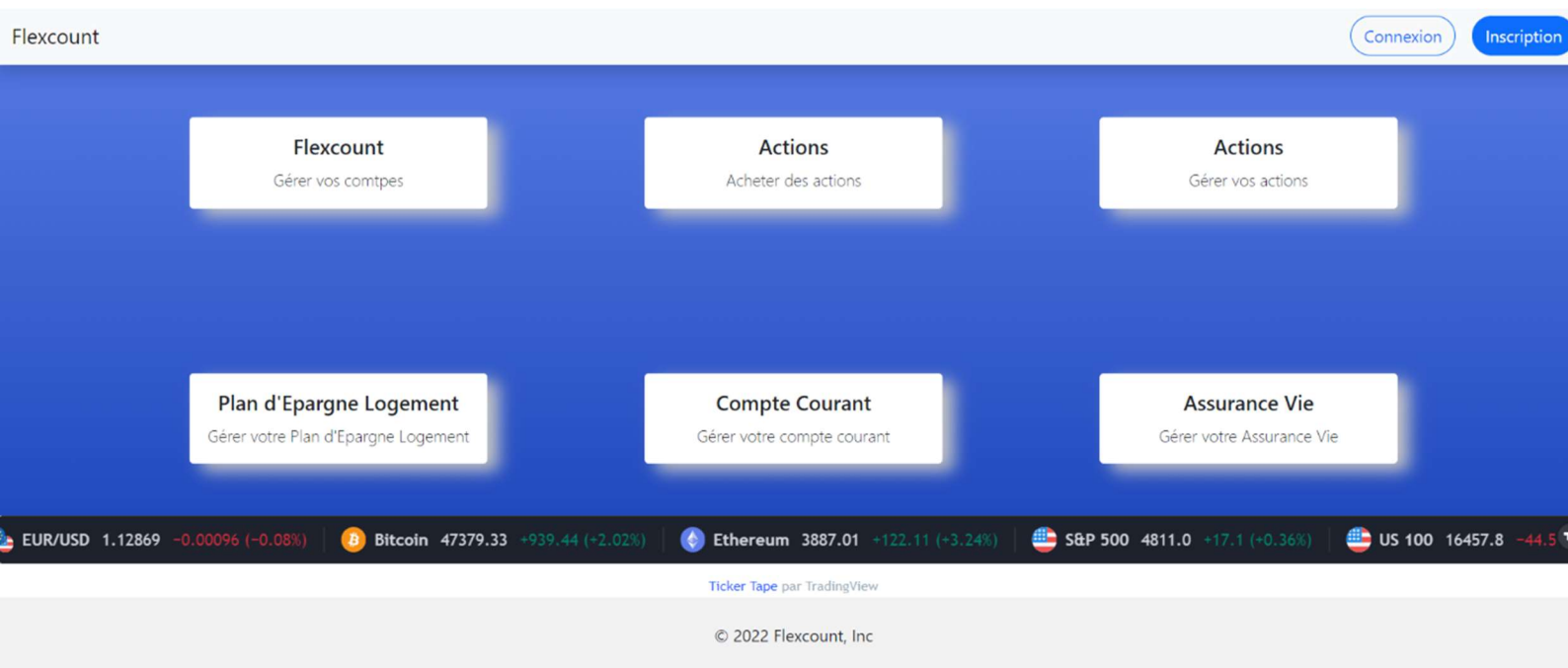
Diagramme de classe

Ensuite, nous avons structuré notre BDD de la manière suivante.

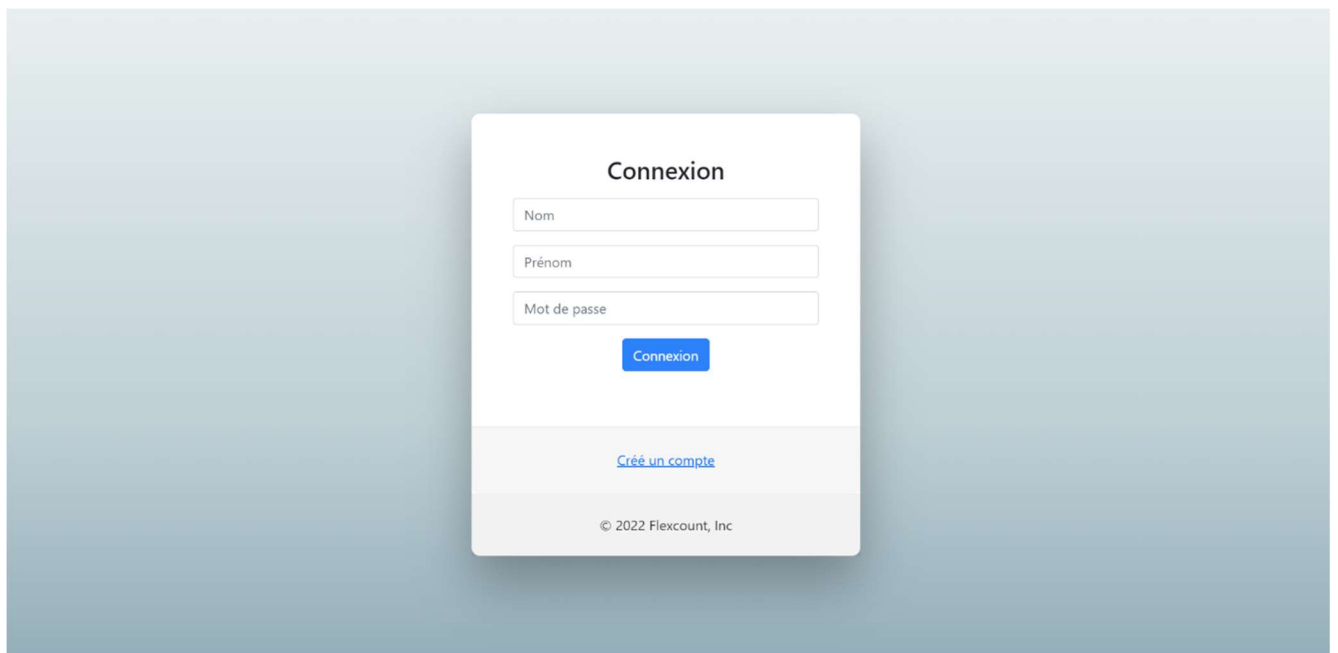


MCD de la BDD

Vues



Racine du site, page d'accueil.



Page de connexion

Inscription

Inscription

© 2022 Flexcount, Inc

Page d'inscription

FlexCount

Dashboard

Déconnexion

Erwan Bouvart

Mon Compte

Nom : Bouvart

Prénom : Erwan

Mot de passe : 123

modifier le mot de passe


Souscrire a une nouvelle assurance vie/PEA

intitulé

Souscrire a une nouvelle assurance vie

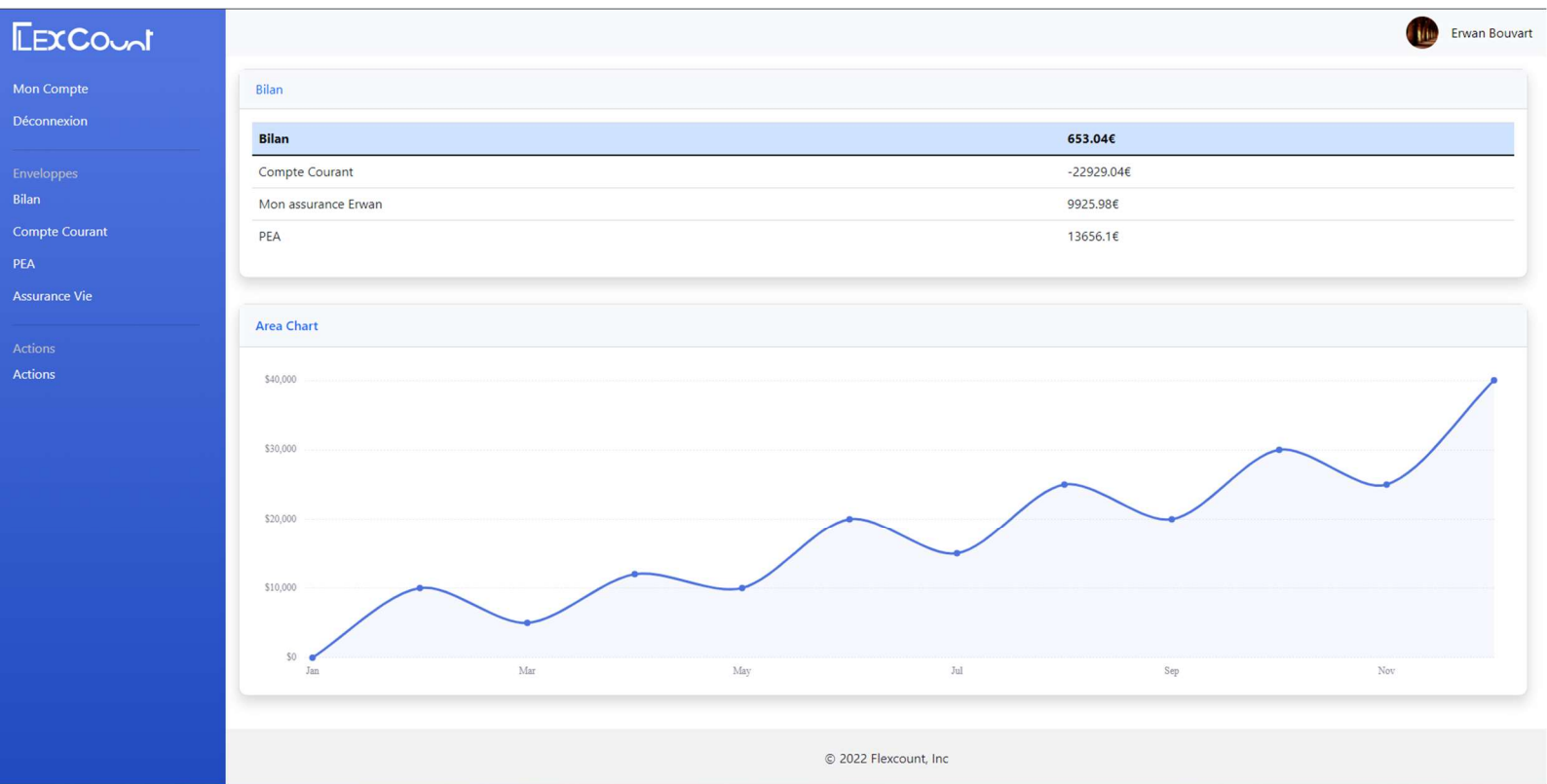
Vous avez déjà un PEA

Profil



© 2022 Flexcount, Inc

Espace compte



Dashboard

FlexCount

Mon Compte

Déconnexion

Enveloppes

Bilan

Compte Courant

PEA

Assurance Vie

Actions

Actions

Erwan Bouvart

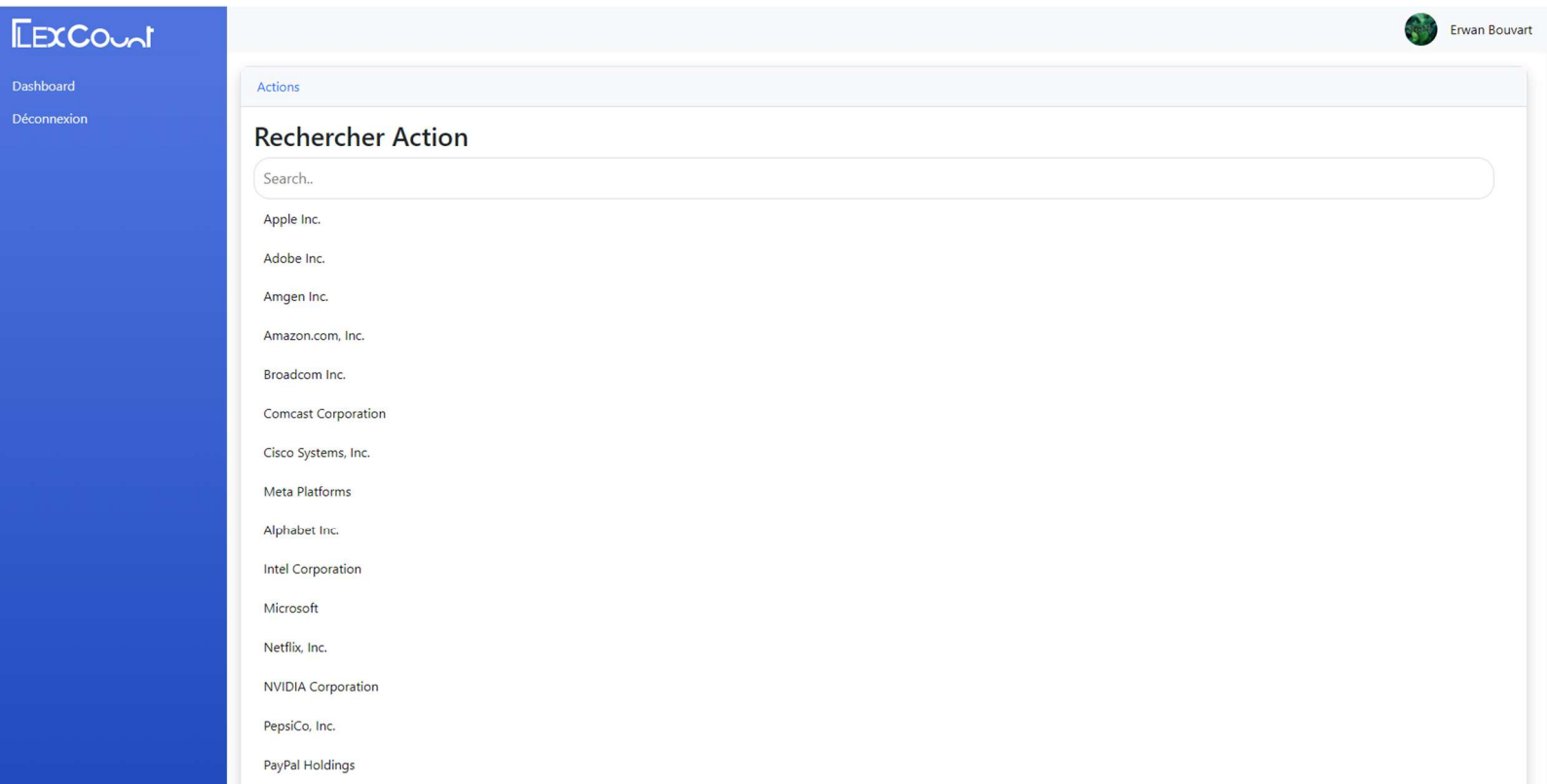
Bilan

PEA					Total :	-2.1299999999999955€	13656.07€
Date d'achat	Nom	Quantité	Prix Achat	Prix Actuel	Rendement	Plus value +/-	Total
2022-01-03	Apple Inc.	2	177.57€	176.505€	-0.5997634735597188%	-2.1299999999999955€	353.01€
2022-01-05	Amazon.com, Inc.	4	3310.6€	3310.6€	0.0%	0.0€	13242.4€
2022-01-05	Cisco Systems, Inc.	1	60.66€	60.66€	0.0%	0.0€	60.66€

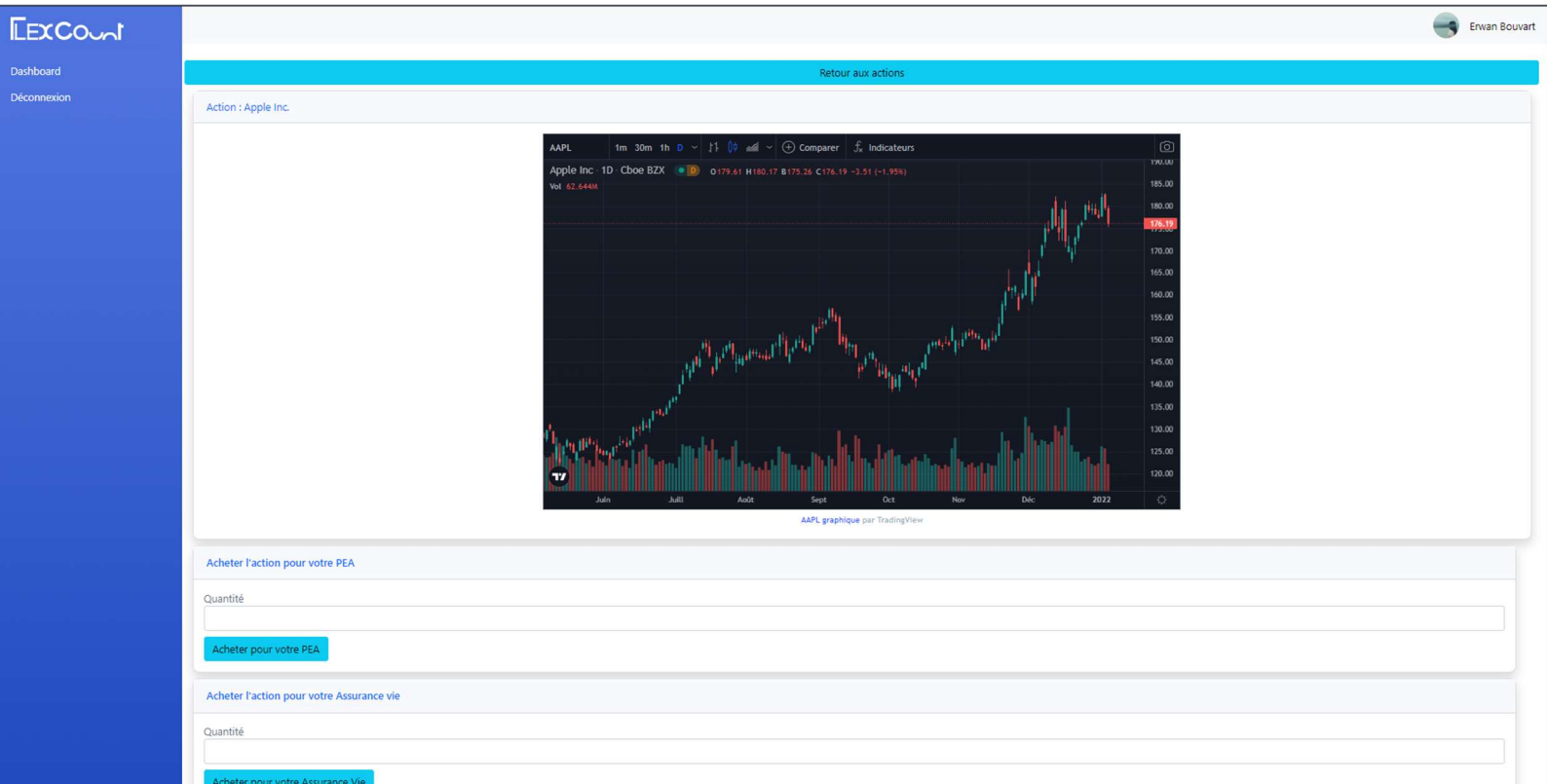
Acheter des actions

© 2022 Flexcount, Inc

Espace PEA



Espace sélection d'actions



Espace Achat d'action