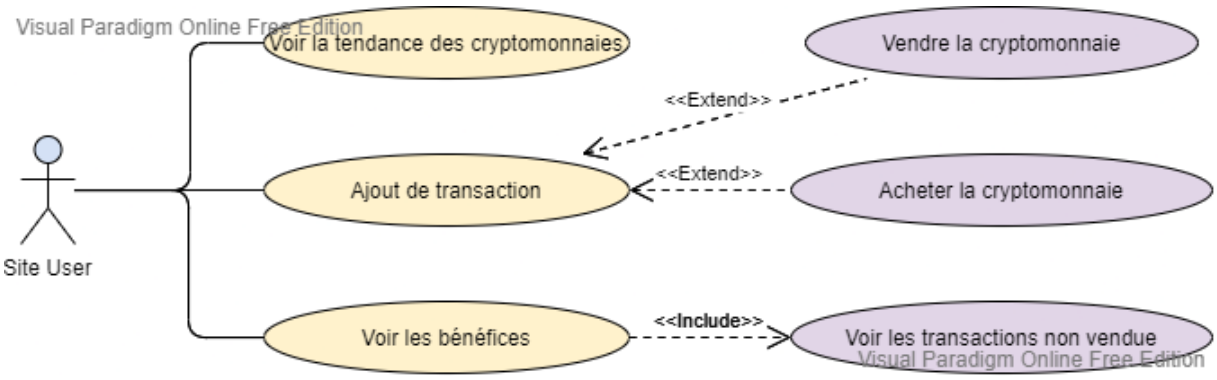
Maxime
Caparros

Rapport pour l’examen certifiant du bloc – Le développement d'une solution digitale

Table des matières

1. Présentation des fonctionnalités

1.1 Diagramme Use Case UML



L'utilisateur du site pourra voir les tendances des différentes cryptomonnaies (10 cryptomonnaie maximum)

Ensuite il pourra ajouter des transactions en ajoutant des cryptomonnaies dans son portefeuille ou en vendant les cryptomonnaie de son portefeuille.

Il pourra ensuite voir les bénéfices de son portefeuille avant qu'il vende pour savoir quand il devra vendre ses cryptomonnaies.

1.2 User stories

Julien Kurtin

Âge
Entre 25 et 34 ans

Niveau d'études
Master ou diplôme équivalent

Secteur d'activité
Vente

Taille de l'entreprise
201 à 500 salariés

Outils nécessaires au quotidien

Téléphone

Objectifs

Faire du bénéfices en achetant de la cryptomonnaie

Principaux défis

Ne pas vendre lorsqu'il n'est pas rentable

Julien souhaite mettre de l'argent dans la cryptomonnaie

- En tant qu'utilisateur, je souhaite voir la tendance des différentes cryptomonnaies afin de pouvoir acheter au meilleur moment.
- En tant qu'utilisateur, je souhaite voir la rentabilité de mon portefeuille afin de pouvoir vendre au meilleur moment.
- En tant qu'utilisateur, je souhaite acheter de la cryptomonnaie afin de pouvoir la vendre plus tard.
- En tant qu'utilisateur, je souhaite vendre ma cryptomonnaie afin de faire des bénéfices.
- En tant qu'utilisateur, je souhaite voir les transactions que j'ai effectué afin de pouvoir me faire une idée sur mes bénéfices total.

1.3 Arborescence fonctionnelle

Sur la page d'accueil nous afficherons les différentes cryptomonnaies (10 maximum) et on affichera la rentabilité au-dessus de la liste. Pour afficher le graphique il suffira de cliquer sur l'affichage de la rentabilité.

Nous pourrons naviguer via les boutons en haut à droite le crayon permettra de visiter la page des transactions avec 2 listes : une pour les transactions non vendues et l'autre pour celles vendues.

Nous pouvons vendre les transactions en cliquant sur vendre a côté des transactions.

Sur le bouton « + » en haut a droite nous pourrons acheter des cryptomonnaies.

2. Les fonctionnalités principales

2.1 Les différentes fonctions principales

- Afficher les différentes cryptomonnaies (10 maximum)
- Acheter de la cryptomonnaie
- Vendre sa cryptomonnaie
- Voir ces transactions
- Voir ces bénéfices sur un graphique

2.2 Les tests

Nous commençons par créer des entités pour tester les getter et setter

```php
public function testRenta()
{
    $rentabilite = new Rentabilite();
    $date= new \DateTime();
    $benefice = 25000;

    $rentabilite->setDate($date)
                ->setBenefice($benefice);
    $this->assertEquals($benefice, $rentabilite->getBenefice());
    $this->assertEquals($date, $rentabilite->getDate());
}
```

Nous faisons pareil pour l'entité transactions.

Ensuite nous testons nos fonctions ici se sera searchKey() qui sert à retourner la cryptomonnaie ayant l'id envoyé en paramètre.

```php
$crypto=$this->searchKey( key: 1);
$this->assertEquals( expected: 'Bitcoin', $crypto[1]);
```

Puis nous vérifions le bon fonctionnement de la base de données.

Tout d'abord on va ajouter des informations dans une base de données test pour pouvoir les manipulés.

```php
class TransaFixture extends Fixture
{
    public function load(ObjectManager $manager): void
    {
        $transa = new Transactions();
        $transa->setName( name: 'BitCoin')
        ->setPrice( price: 1520)
        ->setCreatedAt(new \DateTime())
        ->setQuantity( quantity: 10)
        ->setSolded( solded: false);

        $manager->persist($transa);

        $manager->flush();
    }
}
```

Grâce au bundle Fixture nous pouvons ajouter des données dans une base de données.

```
$ php bin/console doctrine:fixtures:load --env=test

 Careful, database "bddcryto_test" will be purged. Do you want to continue? (yes/no) [no]:
 > yes

   > purging database
   > loading App\DataFixtures\AppFixtures
   > loading App\DataFixtures\TransaFixture
```

Ensuite nous faisons ça aussi pour la table Rentabilité.

Nous testons après la réception des données en comptant le nombre de donnée dans le tableau.

```php
public function testDonneeTransa(): void
{
    $kernel = self::bootKernel();
    $this->assertSame( expected: 'test', $kernel->getEnvironment());
    $routerService = static::getContainer()->get( id: TransactionsRepository::class)->count([]);

    $this->assertEquals( expected: 1, $routerService);
}
```

Ici nous regardons si notre transaction était dans la base de données.

```
OK (5 tests, 10 assertions)
```

3. Choix techniques

Développement sur le Framework Symfony en utilisant la version de PHP 8.1.4.

Ensuite l'utilisation de Bootstrap 5.1.3 pour rendre notre application responsive.

4. Organisation

Tout d'abord j'ai commencé à étudier l'API de coinmarket afin de ressortir les différentes informations nécessaires pour l'application. Ensuite j'ai regardé les informations que l'on devait stocker dans l'application pour mieux structurer ma base de données.

Quant à Magalie elle a créé une maquette de l'application web. Avec une charte graphique.

Ensuite nous avons tout les deux commencé a développer l'application Symfony, j'ai gérer la partie base de données et la réception des données de l'API coinmarket pendant que Magalie initialisé le Template de base avec Bootstrap.

Puis j'ai continué à gérer les données avec des formulaires et Magalie afficha ces données sur les différentes pages.

5. Fonctionnalités manquantes

Il manque l’achat réel de cryptomonnaie, les tests et l’hébergement.