

DOCUMENT QUESTIONS ET REFLEXION

1 – Le choix des technologies

Pour le Front j'ai choisi : Html, css, javascript, bootstrap.

Pour le Back j'ai choisi : Php 7.2 avec le framework Symfony 5.

Pour la Base de données j'ai choisi : Mysql + Doctrine.

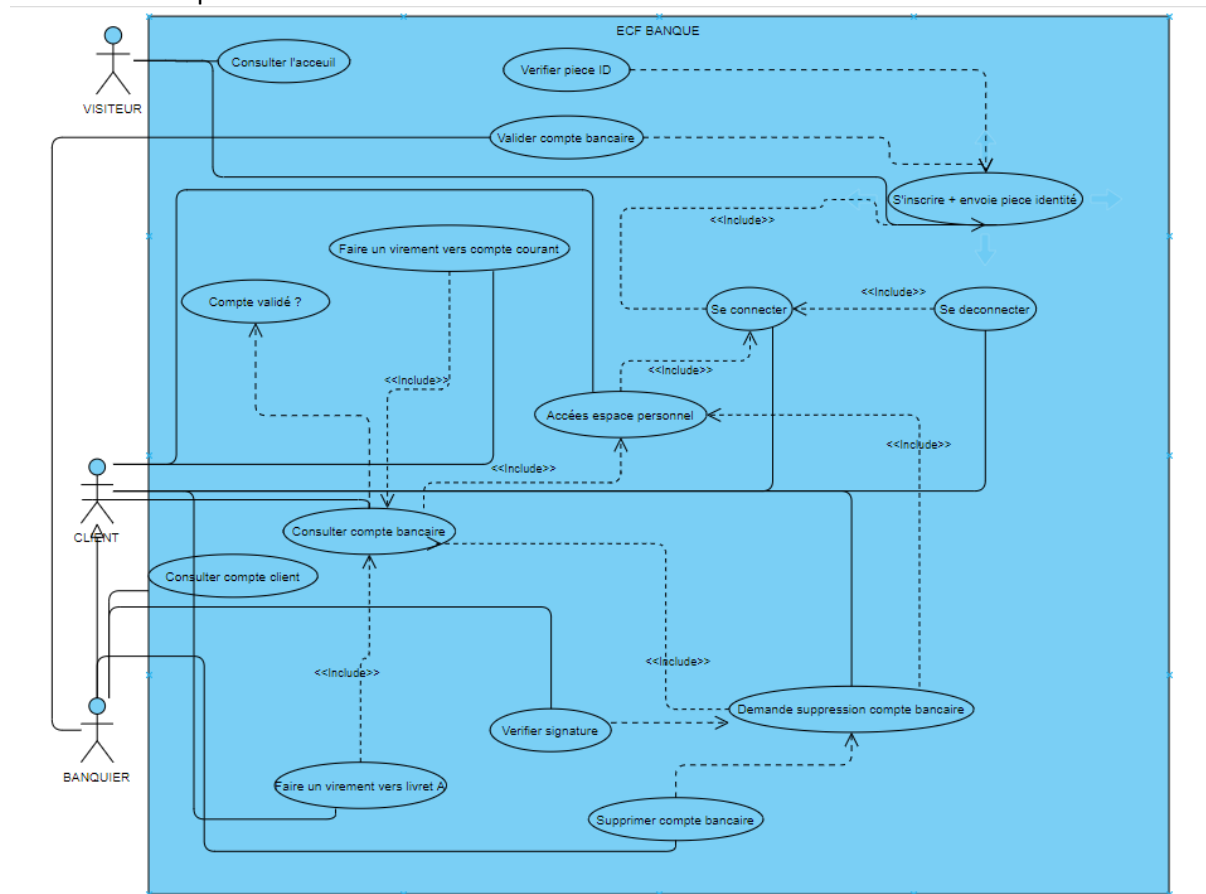
Mon IDE : PHPSTORM.

SERVER : WAMP.

Logiciel d'exploitation : WINDOWS 10.

2 – Les UML

J'ai maqueté un USE CASE avec VISUAL PARADIGME.

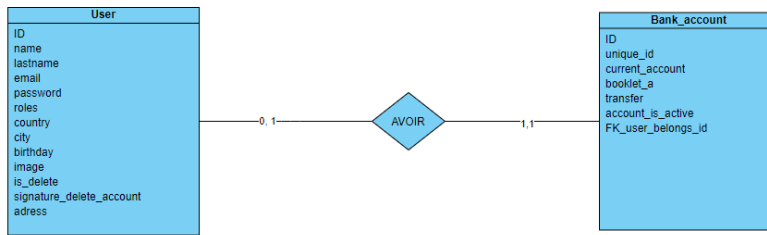


J'ai maqueté un diagramme de classe avec VISUAL PARADIGME.

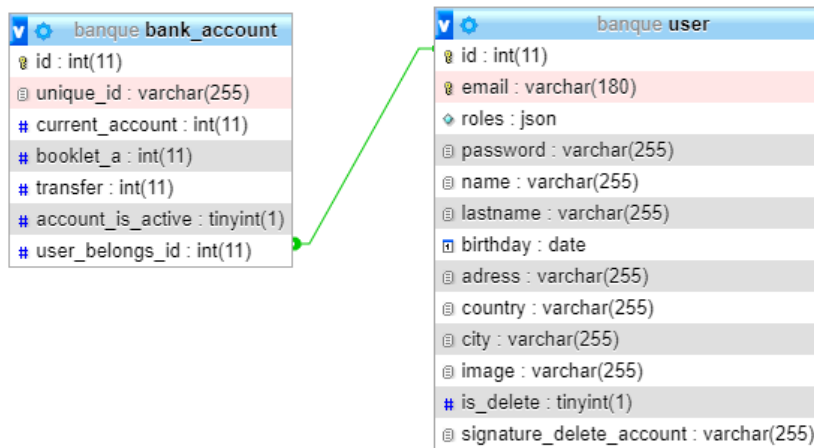
- Je me suis demandé comment un admin pourrait valider un compte ?
Pour cela j'ai établi une variable dans ma table Bank_account que j'ai nommée « account_is_active », qui sera un booléen, cela vaudra soit true ou false, grâce à cela je vais pouvoir établir une condition qui va afficher un compte bancaire que si cette variable est sur true.

Comment un utilisateur pourrait demander une demande de suppression de compte ?

- J'ai établi une variable dans ma table User que j'ai nommée « is_delete » qui sera un booléen et une autre que j'ai nommée « signature_delete_account » qui prendra une image de la signature du client.



Le Modèle physique de données.



3 – Les maquettes

J'ai fait le choix de réaliser le zoning, wireframe avec figma et le mock-up avec adobe xd.

ZONING

The image displays 10 wireframe screens for a banking application, arranged in two columns. Each screen is a gray rectangle with black text and shapes representing UI elements. The screens are numbered 1 through 10 in the top right corner.

- Screen 1:** Header: UTILISATEUR NON INSCRIT. Main content: A large rectangular area with a smaller rectangle inside. Labels: TITLE, TEXT.
- Screen 2:** Header: FORMULAIRE INSCRIPTION. Main content: A vertical stack of five rectangular input fields, followed by a larger rectangular button. Labels: TITLE FORM, form, BUTTON.
- Screen 3:** Header: INTERFACE USER COMPTE BANCAIRE NON VALIDER. Main content: A vertical stack of six rectangular input fields on the left, and a single rectangular button on the right. Labels: INFOS USER, COMPTE BANCAIRE EN COURS DE TRAITEMENT.
- Screen 4:** Header: FORMULAIRE INSCRIPTION. Main content: A vertical stack of five rectangular input fields, followed by a larger rectangular button. Labels: TITLE FORM, form, BUTTON.
- Screen 5:** Header: INTERFACE USER COMPTE BANCAIRE VALIDER. Main content: A vertical stack of six rectangular input fields on the left, and a single rectangular button on the right. Labels: INFOS, INFO COMPTE BANCAIRE.
- Screen 6:** Header: VIREMENT. Main content: A vertical stack of four rectangular input fields.
- Screen 7:** Header: INTERFACE USER COMPTE BANCAIRE NON VALIDER. Main content: A vertical stack of six rectangular input fields on the left, and a single rectangular button on the right. Labels: INFOS USER, COMPTE BANCAIRE EN COURS DE TRAITEMENT.
- Screen 8:** Header: FORMULAIRE INSCRIPTION. Main content: A vertical stack of five rectangular input fields, followed by a larger rectangular button. Labels: TITLE FORM, form, BUTTON.
- Screen 9:** Header: INTERFACE USER COMPTE BANCAIRE VALIDER. Main content: A vertical stack of six rectangular input fields on the left, and a single rectangular button on the right. Labels: INFOS, INFO COMPTE BANCAIRE.
- Screen 10:** Header: VIREMENT. Main content: A vertical stack of four rectangular input fields.

ZONING MOBILE



WIREFRAME

ACCUEILInscriptionConnexion

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

FORMULAIRE INSCRIPTION

ACCUEILInscriptionConnexion

EMAIL

MOT DE PASSE

NOM

CONFIRMER MDP

PRENOM

VILLE

PAYS

DATE DE NAISSANCE

ADRESSE

PIECE D'IDENTITE

VALIDER

INTERFACE USER COMPTE NON VALIDER

ACCUEILCOMPTE DECONNECTION

SAHOULI

BILLEL

FR

SAINT-PIREST

18 RUE DE TOTO

COMPTE BANCAIRE EN COURS DE TRAITEMENT
Cela pourrait prendre quelques heures

INTERFACE USER COMPTE VALIDER

ACCUEILCOMPTE DECONNECTION

SAHOULI

BILLEL

FR

SAINT-PIREST

18 RUE DE TOTO

b.sahoul@hotmail.fr

ID : Bee54e5zaS

COMPTE COURANT : 3000 EURO
Click pour virement du Compte Courant vers Livret A

LIVRET A : 1500 EURO
Click pour virement du Livret A vers Compte Courant

SITUATION DU COMPTE : ACTIF

DEMANDE DE SUPPRESSION DU COMPTE BANCAIRE

SUPPRIMER☐

VALIDER

VIREMENT

Menu

COMPTE COURANT

3000 EURO

CLICK BLOQUER

LIVRET A

1500 EURO

CLICK BLOQUER

TRANSFERE

VALIDER

BANKFORCE

Inscription

Email

ex : MrJack@outlook.fr

Mot de passe

6 caractères minimum

Confirmez votre mot de passe

6 caractères minimum

Prénom

Prénom

Nom

Nom

Date de naissance

01

BANKFORCE

Connexion

E-mail

Mot de passe

Connexion

4 – Le choix de l'architecture et configuration

J'ai fait le choix de suivre une architecture MVC (Modèle Vue Contrôleur).

Le principe est simple, dans mon cas j'utilise Doctrine, Doctrine va générer les entités de mes tables qui vont représenter le Modèle. Le contrôleur lui va être le chef d'orchestre il va non seulement se servir du Modèle pour enregistrer et envoyer de la data dans ma base de données mais il va aussi renvoyer des données à la Vue, et la Vue va se charger d'afficher les données à l'utilisateur.

Par la suite dans le fichier «.env » je peux passer en environnement de dev ou de prod.

```
###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=842f92e4b05a6db98df4fc2c7a95bbdb
```

Toujours dans le même fichier je peux créer ma base de données et la configurer en mettant le dbUser le mot de passe et le nom de la base de données que je veux lui donner.

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/banque?serverVersion=5.7"
```

5 – Les bonnes pratiques de sécurité

Dans le security.yaml il y a la gestion du hachage du mot de passe, il va prendre le meilleur type de hachage possible, dans mon cas il a sélectionné le hachage en « argon ».

```

security:
  encoders:
    App\Entity\User:
      algorithm: auto

```

Là je peux gérer aisément la sécurité des connexions, lorsqu'un utilisateur s'inscrit son rôle est ROLE_USER, ainsi pour déclarer qu'un user est un admin je doit le modifier dans ma table et lui insérez le ROLE_ADMIN.

```

access_control:
  - { path: ^/admin, roles: ROLE_ADMIN }
  - { path: ^/compte, roles: ROLE_USER }

```

	id	email	roles	password	name	lastnam
1	4	b.sahouli@hotmail.fr	["ROLE_ADMIN"]	\$argon2id\$v=19\$m=65536,t=4,p=1\$MzN1UG10RFU4Tk...	Billel	Sahouli
2	5	edede@hotmail.fr	[""]	\$argon2id\$v=19\$m=65536,t=4,p=1\$UVJBVHYuc1p0Mm...	Billel	Sahouli
3	6	dddd@hotmail.fr	[""]	\$argon2id\$v=19\$m=65536,t=4,p=1\$0UpLTVRo0FdCVH...	Billel	Sahouli
4	11	testDate@hotmail.fr	[""]	\$argon2id\$v=19\$m=65536,t=4,p=1\$VzJJakhJdkhRN2...	Billel	Sahouli

Un exemple de mes routes sécurisé.

```

/**
 * @Route("/compte/virement-compte-courant/{id}", name="account_current")
 */
public function transferCurrentAccount($id, Request $request): Response

```

```

/**
 * @Route("/admin/delete/{id}", name="admin_delete_account")
 */

```