La banque, la vrai.

Gestionnaire de comptes

Par Vincent Gendron

Table des matières

[Présentation du projet 2](#_Toc520992456)

[Liste des compétences couvertes 2](#_Toc520992457)

[Cahier des charges 3](#_Toc520992458)

[Fichiers 3](#_Toc520992459)

[Base de données 4](#_Toc520992460)

[Dictionnaire de données 4](#_Toc520992461)

[MCD 6](#_Toc520992462)

[MLD 7](#_Toc520992463)

[Script SQL 8](#_Toc520992464)

[Visuel 9](#_Toc520992465)

[Déroulement du developpement 11](#_Toc520992466)

[Conclusion : 11](#_Toc520992467)

[Définitions : 12](#_Toc520992468)

## Présentation du projet

Le but de ce projet est de réaliser une application permettant de gérer son budget et son/ses compte(s) bancaire(s) caractérisé(s) par un titulaire du compte.

Pour ce faire il est demandé de pouvoir afficher la liste des comptes d’un utilisateur, la listes des opérations d’un utilisateur ainsi que 3 formulaires pour ajouter des opérations, des types et des catégories.

Ce projet a démarré le lundi 30/07 à 13h30 et est à faire avec un deadline donné qui est le jeudi 02/08 à 12h, ce qui nous donne 21h pour le développer avec 3h30 supplémentaire pour faire ce document ainsi qu’un PowerPoint de présentation.

## Liste des compétences couvertes

Pour cette application de gestion de comptes bancaires j’ai décidé :

- De développer la partie serveur en PHP uniquement, mes notions en symphonie étant trop faible.

- De développer la partie client en HTML/CSS, la partie visuelle étant peu importante, je ne me suis pas attardé sur des grandes parties de design.

- De gérer les données via une base de données SQL, afin d’avoir un stockage efficace et rapide.

- De gérer la sécurité au maximum de mes capacités, les fuites de données étant très problématique, surtout pour une application bancaire.

## Cahier des charges

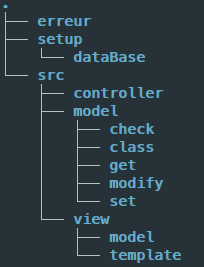
\* Faire une base de données gérant des comptes en banques et des opérations sur celui-ci :

- Un compte en banque doit être caractérisé par un titulaire de compte, un numéro de compte, une agence (qui a une adresse, un nom et une banque de référence).

- Une opération sur un compte doit être caractérisé par un montant, un type, une catégorie, une date, un libellé, un id, une régularité ou non, et par le fait que ce soit un débit ou un crédit.

\* Faire une application affichant les informations de la base de données et permettant d’ajouter des opérations ainsi que des types/catégories.

## Fichiers



L’arborescence des fichiers est basée sur l’architecture MVC mêlé avec des notions du MVT (vu sur le framework Django de python).

Le dossier model dans la vue met en place les variables, alors que le dossier template de la vue n’est que des fichiers html avec l’affichage des variables.

Un dossier setup a aussi été créée afin de pourvoir intégrer la base de donne sur le serveur du client.

Un dossier erreur est également présent pour référencer les bugs connus sur cette application.

## Base de données

### Dictionnaire de données

#### Structure de la table account :

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Valeur par défaut** |
| ***account\_id*** | int(11) | Non |  |
| name | varchar(50) | Non |  |
| balance | int(11) | Non |  |
| user\_id | int(11) | Non |  |
| agency\_id | int(11) | Non |  |

#### Structure de la table agency

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Valeur par défaut** |
| ***agency\_id*** | int(11) | Non |  |
| name | varchar(50) | Non |  |
| adress | varchar(50) | Non |  |
| bank\_id | int(11) | Non |  |

#### Structure de la table bank

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Valeur par défaut** |
| ***bank\_id*** | int(11) | Non |  |
| name | varchar(50) | Non |  |

#### Structure de la table category

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Valeur par défaut** |
| ***category\_id*** | int(11) | Non |  |
| name | varchar(50) | Non |  |
| user\_id | int(11) | Non |  |

#### Structure de la table operation

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Valeur par défaut** |
| ***operation\_id*** | int(11) | Non |  |
| date | datetime | Non |  |
| name | varchar(50) | Non |  |
| count | int(11) | Non |  |
| regular | tinyint(1) | Non |  |
| user\_id | int(11) | Non |  |
| account\_id | int(11) | Non |  |
| category\_id | int(11) | Non |  |
| type\_id | int(11) | Non |  |

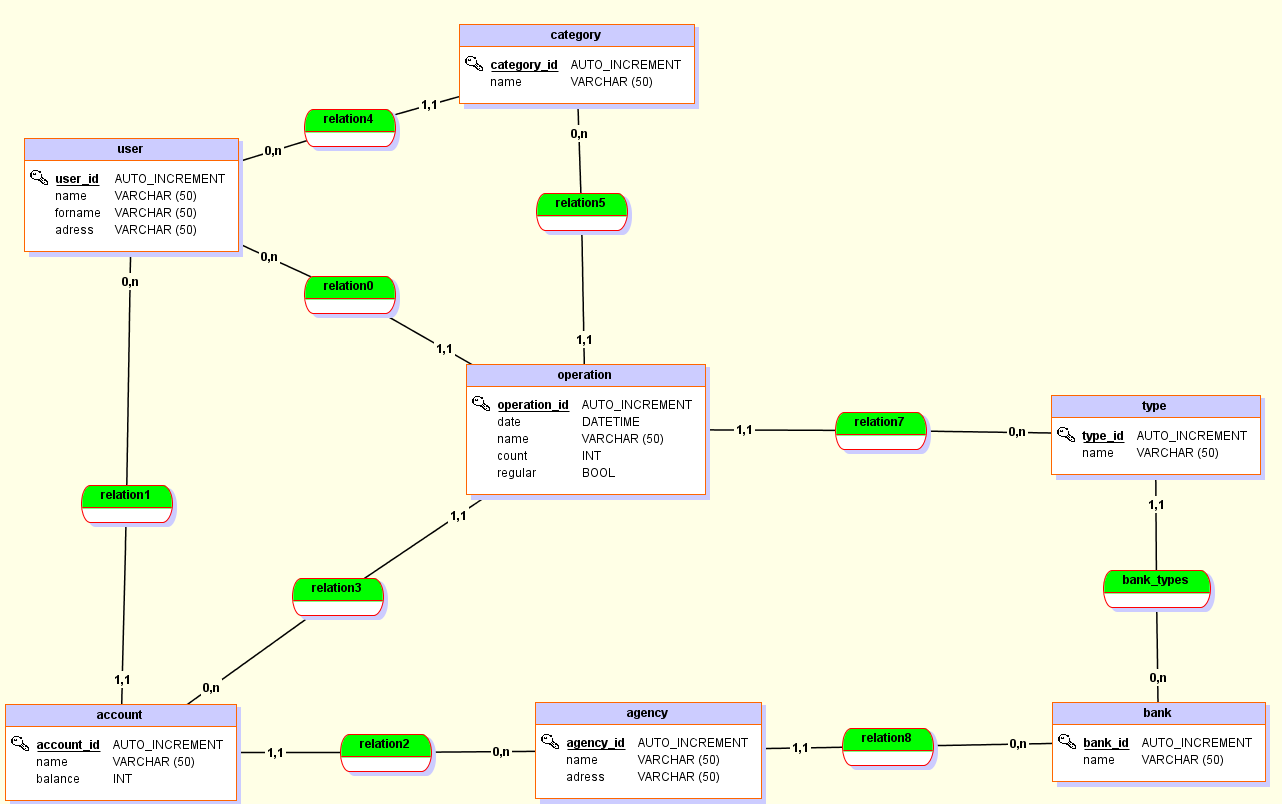
#### Structure de la table type

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Valeur par défaut** |
| ***type\_id*** | int(11) | Non |  |
| name | varchar(50) | Non |  |
| bank\_id | int(11) | Non |  |

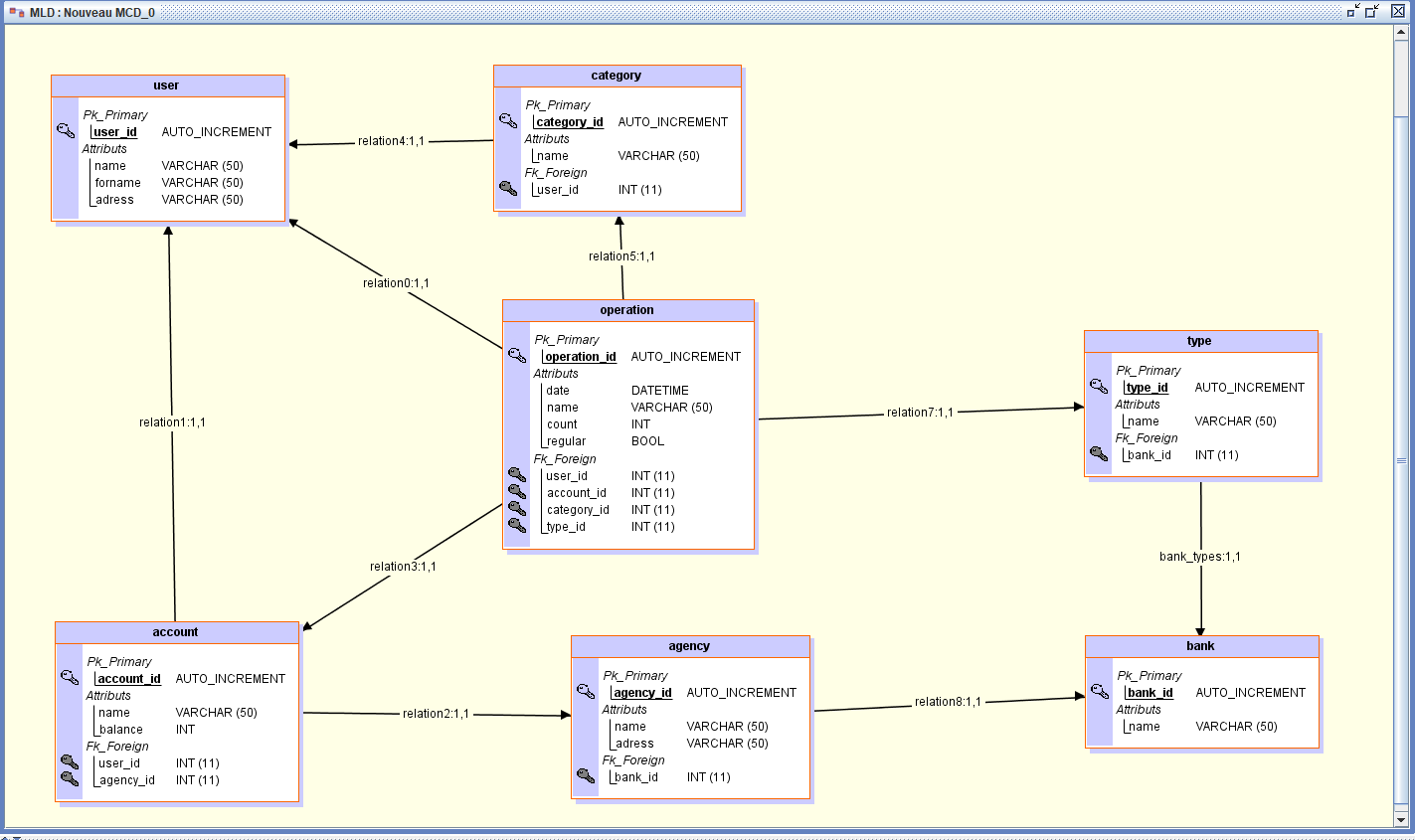
#### Structure de la table user

|  |  |  |  |
| --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Valeur par défaut** |
| ***user\_id*** | int(11) | Non |  |
| name | varchar(50) | Non |  |
| forname | varchar(50) | Oui | NULL |
| adress | varchar(50) | Oui | NULL |

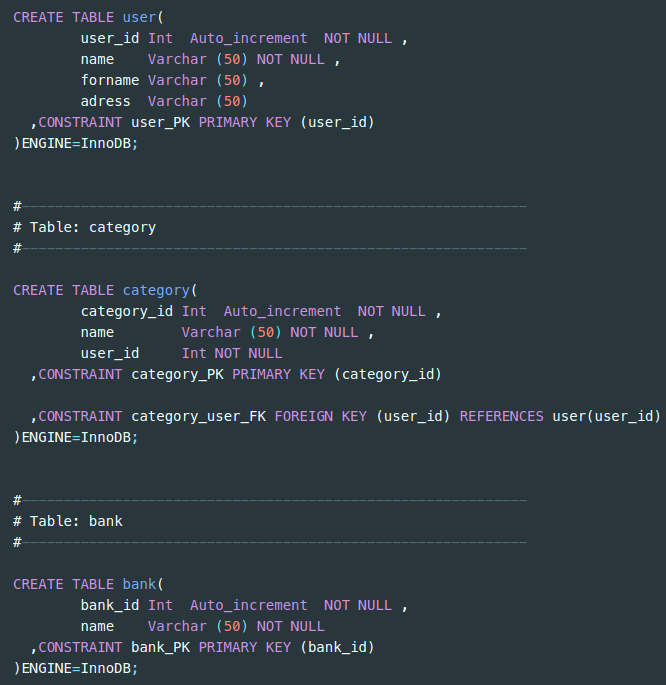
### MCD

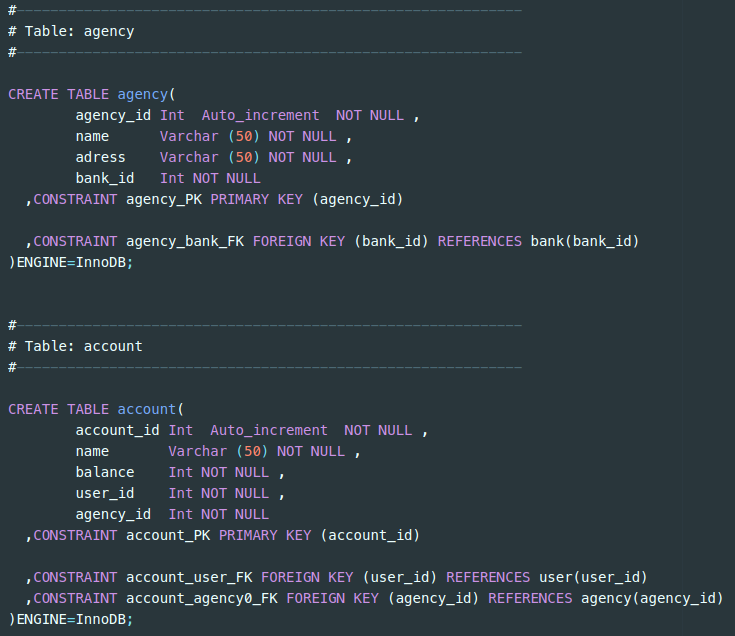
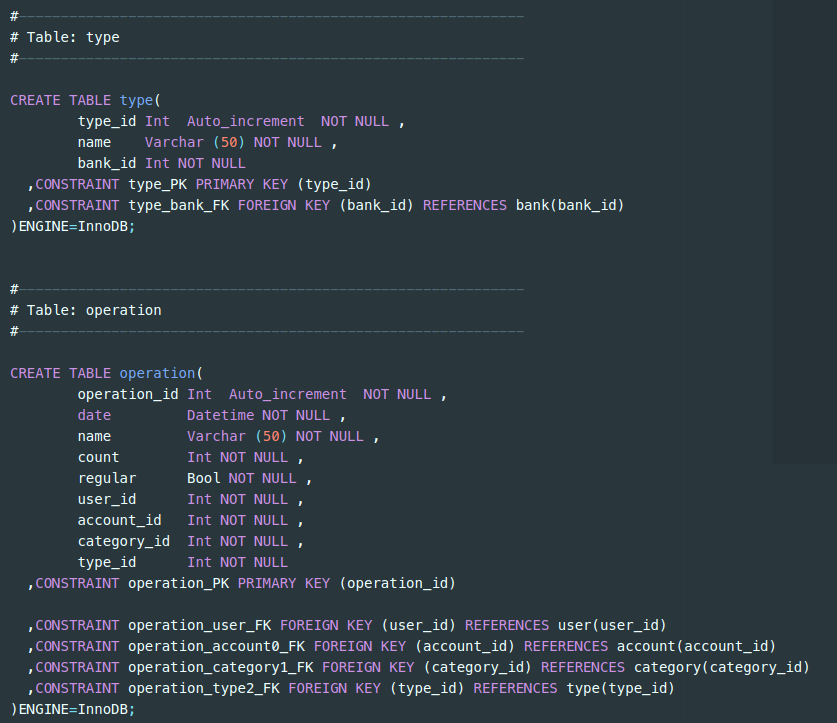


### MLD



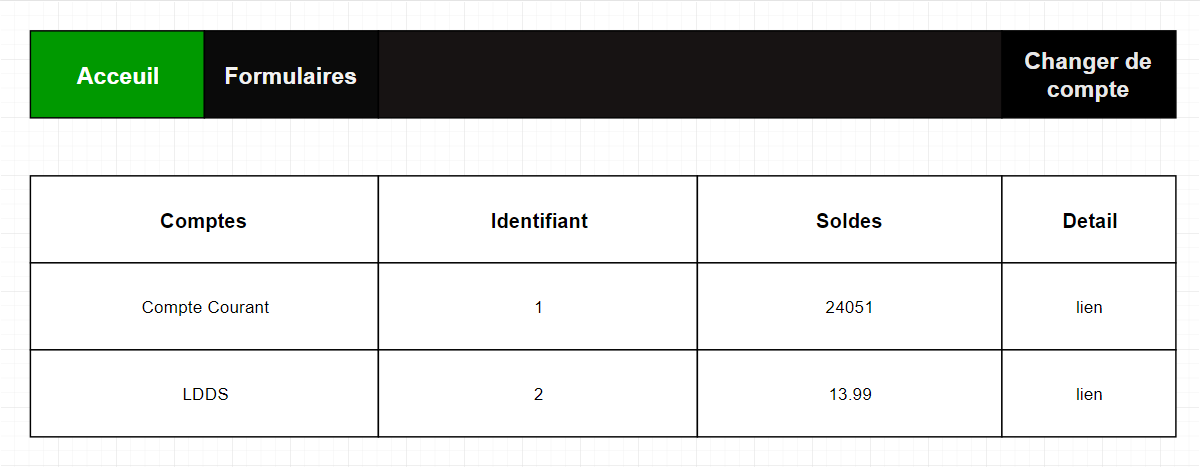
### Script SQL



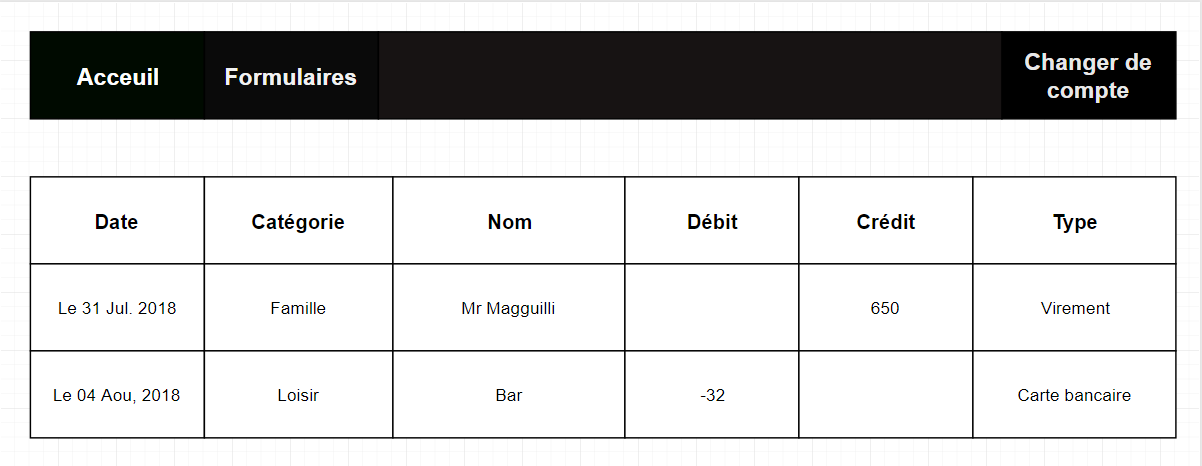


## Visuel

#### Vue accueil pour un utilisateur connecté :



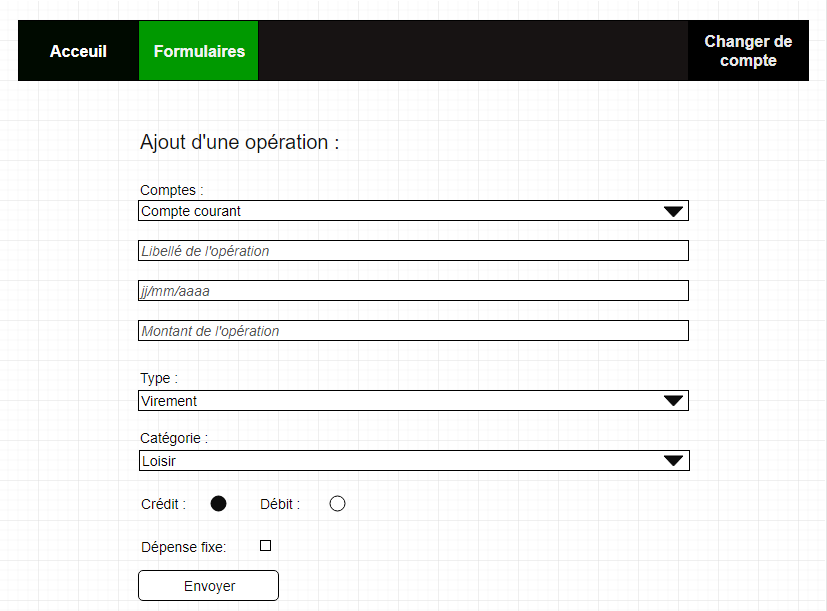
#### Vue pour les détails d’un compte :



#### Vue pour ajouter des catégories/Types :



#### Vue pour ajouter des operations :



## Déroulement du developpement

Via ce projet, il a fallu faire un choix dans sa manière de développer.  
  
Le premier étant de se préparer au projet puis d’essayer de le faire calmement. Le second était d’agir directement et implémenter le plus de fonctionnalités possible. Je code lentement j’ai donc passé choisis la seconde méthode et mis beaucoup de temps à implémenter le plus possible. Malheureusement je ne suis pas satisfait du résultat.

Ayant fait divers tests de manière de developper lors du dernier projet, le code source n’était pas reutilisable. Ce qui a été un reel frein pour mon avancée dans ce projet.

## Conclusion :

Presque la totalitee de ce qui est défini par le sujet a été réalisé.

Mais, le sujet étant vague sur le contexte, le type de paiement est défini dans ma base comme une table lie à la banque dans lequel l’utilisateur est.

Hormis cette fonctionnalité manquante, le design de l’application est à revoir.

C’était mon premier projet où je me suis occupé du front end, et j’ai beaucoup appris via ce projet.

## Définitions :

HTML : Langage de balisage utilisé pour la création de pages web.

CSS: Le CSS est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers [HTML](http://glossaire.infowebmaster.fr/html/).

PHP : Ce langage est principalement utilisé pour produire un site web dynamique.

PYTHON : Le python est un langage conçu pour augmenter la productivité des développeurs.

MVC : (Modèle Vue Controlleur): Modèle d'architecture qui cherche à séparer nettement les couches de présentation, métier et d'accès aux données.

MCD : Modèle conceptuel de données.

MLD : Modèle logique de données.

SQL : Langage permettant de communiquer avec une base de donnée.