Université de Techonologie de Compiègne

LO21

Rapport

Projet Trésorerie

Autores:
BENDYNA Lucas
GOURET Mathilde
RAMIREZ Brenda
XIE Sihan

Índice general

0.1. Résumé de l'application.

0.1.1. Opérations implémentes

Connexion

Connexion permet d'accéder aux informations de la base de données déconnecter l'utilisateur.

Comptes

Les comptes peuvent être classifiées en catégories:

- Actifs
- Passifs
- Recettes
- Dépenses

Chacune a la possibilité d'ajouter une compte, l'éditer, réaliser une transaction et d'afficher ces transactions. **Ajouter un compte** prends en compte:

- Nom
- Compte père
- S'il est virtuel ou pas
- Solde initial
- Compte de capitaux propre à utiliser

Ajouter une transaction contient:

- Réference
- Intitulé
- Comptes
- Crédit
- Débit

0.1.2. Opérations non implémentes

- LOGIN : Il n'est pas possible d'ajouter un compte dans l'application.
- COMPTES: L'application manipule une seule compte

Transactions

L'onglet Transactions affiche les transactions effectués. Il est implémenté avec deux boutons, une pour ajouter une transaction avec le même format de celui-ci des comptes et l'enregistrement pour actualiser l'affichage.

Rapports

Rapports représente le Bilan (d'actifs et passifs), et le Compte de résultat (). Bilan affiche les comptes et ses montants. D'autre côté le compte de résultat affiche le total recettes,le total dépenses et le perte d'aujourd'hui Cet onglet ne peut pas montrer l'option **clôture**

Clôture

La clôture remettre à zéro les comptes de recettes et dépenses par l'intermediaire d'un compte résultat et transfère la différence ver s le compte excédent ou déficit au contraire

Sauvegarder le contexte

Sauvegarder le contexte au démarrage de l'application, l'état de l'application, les paramétrages présents lors de la dernière exécution sont récupérés

Déconnexion

Déconnexion permet de déconnecter l'utilisateur

0.1.3. Description de l'architecture

- Base de données:Nous avons choisi une base de données plutôt qu'un table ou document à raison qu'ils sont bien plus efficaces lorsqu'il s'agit de retrouver des informations. Il est possible de les afficher sous forme de formulaires dans l'interface graphique, permettant d'accéder en toute liberté à tous les détails enregistrés.
- Qt Designer UI: Implémentation de UI pour représenter l'interface graphique de l'application à travers d'un éditeur visuel qui permet de positionner et ajuster les widgets plus facilement et comme nous le souhaitons
- Desing Patterns MVC : Le design pattern MVC permettre de séparer le modèle, la vue et le contrôleur. Le modèle : Il représente les données de l'application, en définissant définit aussi l'interaction avec la base de données et son traitement. La vue : Elle se contente d'afficher les données que lui fournit le modèle. Le contrôleur : Il effectue la synchronisation entre le modèle et les vues.
- Desing Patterns Singleton : Lors du développement de nos logiciels, nous souhaitons sans doute nousus assurer de n'avoir qu'une seule instance de certaines classes dans le cas du Singleton.

0.1.4. Argumentation de l'architecture permettant plus facilement les évolutions

0.1.5. Base de données

De plus, le choix de PostgreSQL a été parce qu'il est le plus complet aujourd'hui, permet de sauvegarder et restaurer des backups facilement, capable de manipuler gros volumes de données.

0.1.6. Qt Designer UI

Les fichiers d'interface utilisateur de Qt Designer représentent l'arborescence des widgets du formulaire au format XML. Les formulaires peuvent être traités:

Au moment de la compilation, ce qui signifie que les formulaires sont convertis en code C ++ qui peut être compilé. Au moment de l'exécution, ce qui signifie que les formulaires sont traités par la classe QUiLoader qui construit dynamiquement l'arborescence des widgets lors de l'analyse du fichier XML.

0.1.7. Desing Patterns MVC

Le modèle de conception permet principalement deux choses :

- Le changement d'une couche sans altérer les autres. Nous pourrions aussi donc changer le modèle sans toucher à la vue et au contrôleur. Cela rend les modifications plus simples.
- La synchronisation des vues. Avec ce design pattern, toutes les vues qui montrent la même chose sont synchronisées.

0.1.8. Desing Patterns Singleton

La solution que nous avons trouver est d'utiliser le pattern singleton. En effet celui-ci va, via une de ses méthodes, nous permettre de récupérer l'unique instance de la classe.

0.2. Planning du projet

Le planning du projet a été planifié à travers d'un diagramme de GANTT réaliser pour tous les membres du groupe.

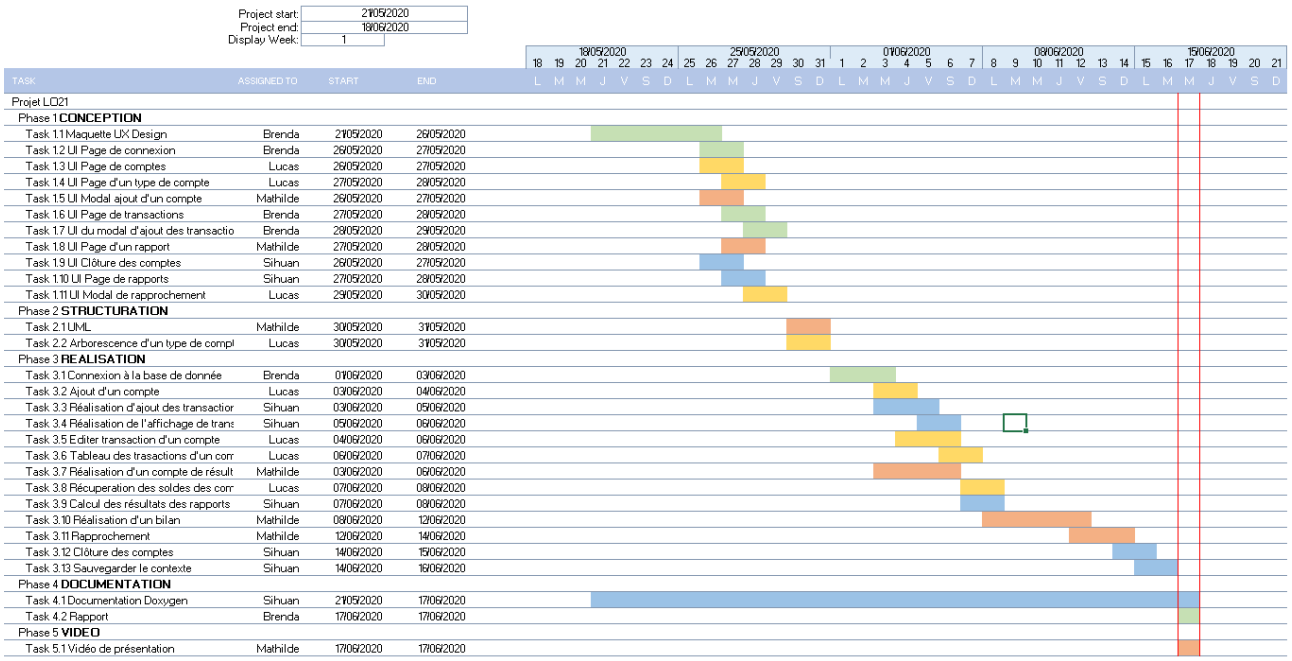

Figura 1: Diagramme GANTT

.

0.3. Modèle UML

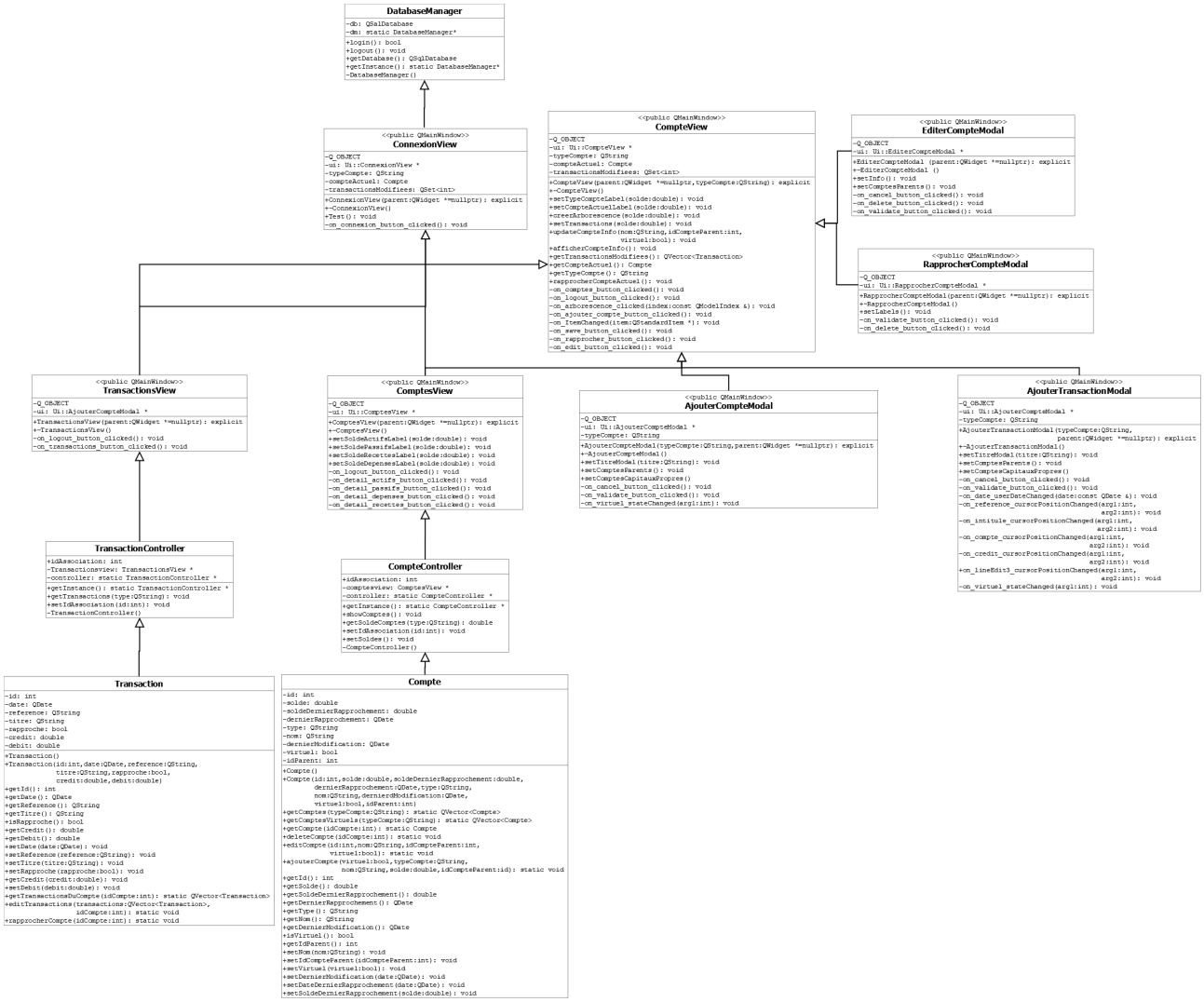

Figura 2: Modèle UML

.

0.4. Contribution personnelle des membres du groupe

0.4.1. BENDYNA Lucas

- UI Modal de rapprochement
- Rapprochement d'un compte
- UI Page de comptes
- Récuperation des soldes des comptes
- UI Page d'un type de compte
- Arborescence d'un type de compte

- Tableau des transactions d'un compte
- Ajout d'un compte
- Éditer Transaction d'un compte

0.4.2. GOURET Mathilde

- Réalisation d'un bilan
- Réalisation d'un compte de résultat
- UI Page d'un rapport
- UI Modal Ajout d'un compte
- UML
- Rapprochement
- Vidéo de présentation

0.4.3. RAMIREZ Brenda

- UI Page de connexion
- Connexion à la base de donnée
- Maquette UX Design
- UI du modal d'ajout des transactions
- UI Page de transactions
- Rapport

0.4.4. XIE Sihuan

- Documentation Doxygen
- UI Page des rapports
- Calcul des résultats des rapports à l'instant présente
- UI Clôture des comptes
- Clôture des comptes
- Réalisation d'ajouts des transactions
- Réalisation de l'affichage de transaction
- Sauvegarder le contexte

0.5. Annexes

0.5.1. Installation base de données

Instructions avant le démarrage du projet:

- Vous devrez installer PostgresSQL en suivant les instructions de ce lien Nous vous recommandons de choisir le mot de passe: **luluben08**. Au contraire vous devrez modifier le mot de passe directement dans le fichier **databasemanager.cpp** sur la partie du code **this -> db.setPassword** en ajoutant votre mot de passe. Après l'installation vous démarrerez PostgreSQL

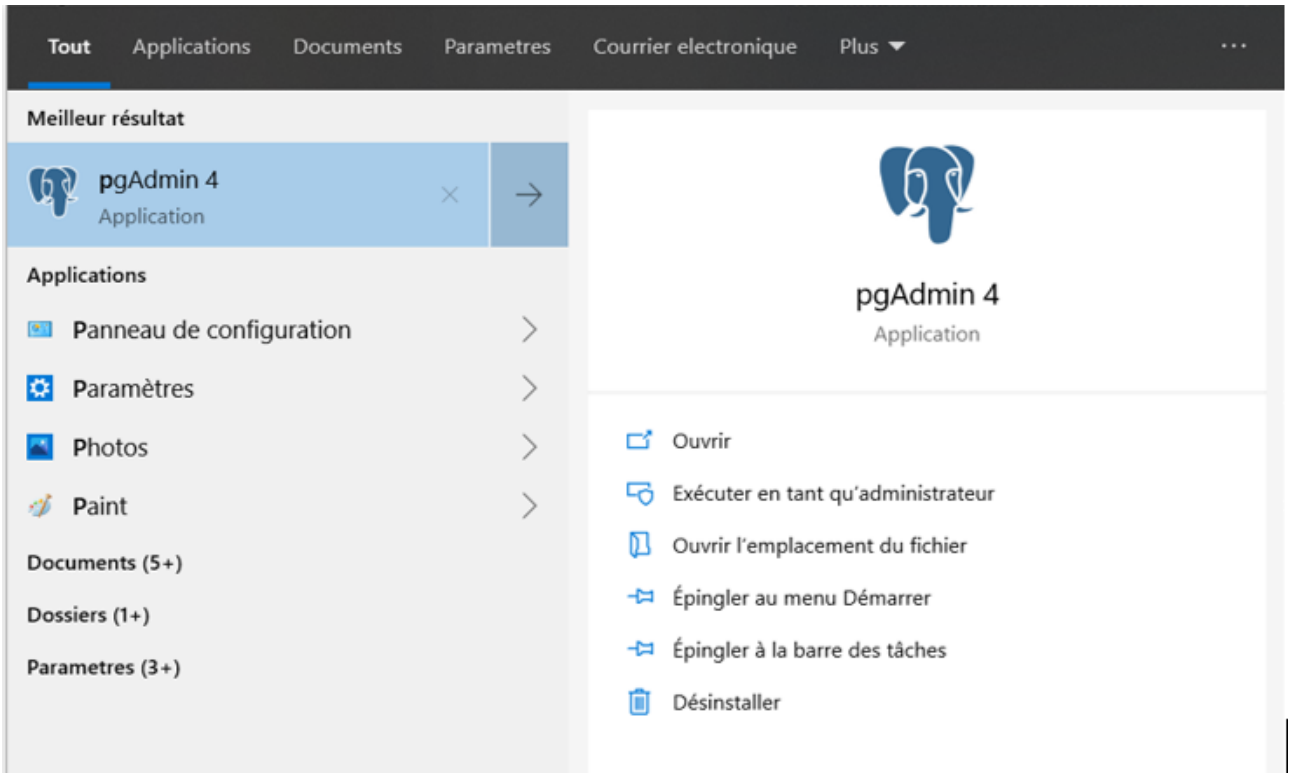

Figura 3: Démarrer Postgres

Vous introduirez le mot de passe et ensuite vous cliquerez sur Query Tool pour coller le code dans du fichier **script.sql**, de plus vous devrez ajouter l'instruction **INSERT INTO public.association (identifiant, mot_de_passe) VALUES ('user','password');** à la fin en substituant votre propres données de user et passeword. Finalement l'exécuter .

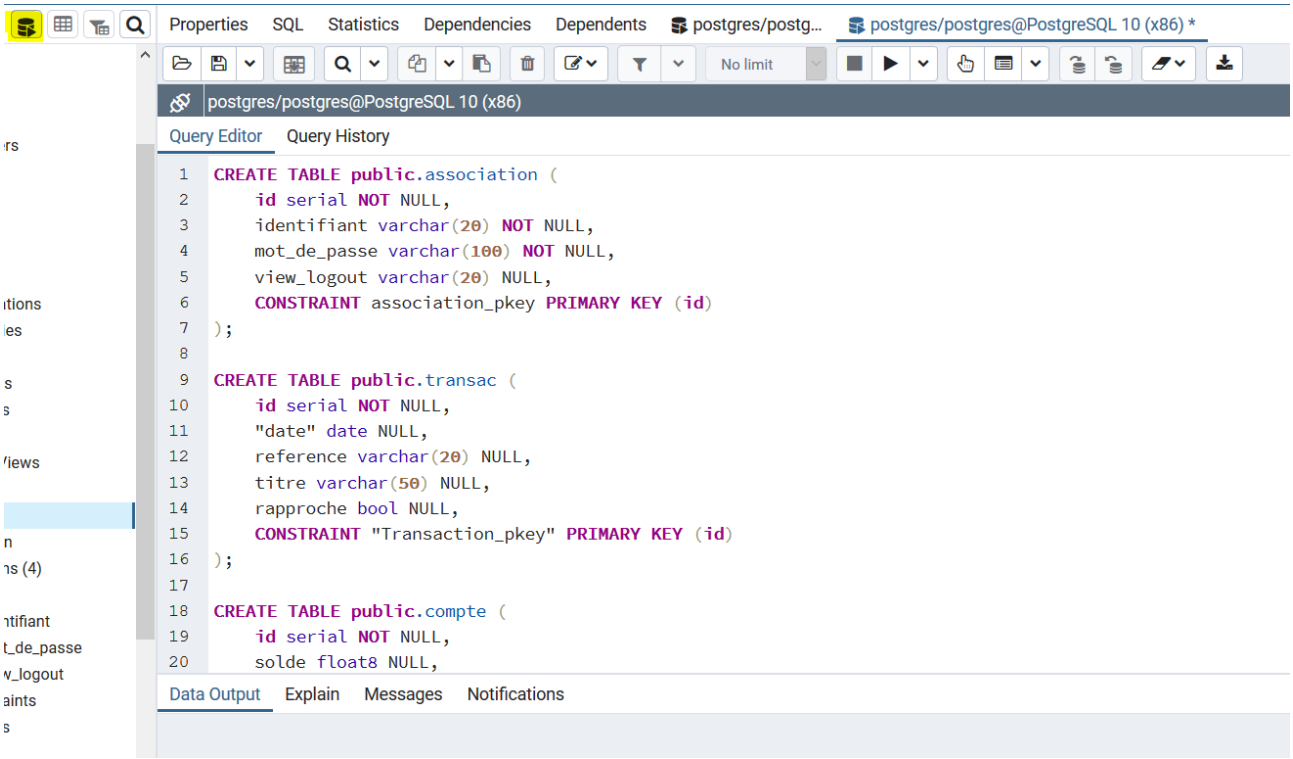

Figura 4: Ajouter base de données

- Vous devrez vérifier la connexion entre Qt et PostgreSQL en cliquant sur l'onglet **Path** de **Environnement de Compilation**. Path doit contenir où se localise le fichier bin et lib de PostgresSQL? pour exemple: **C:\ Program Files (x86)\ PostgreSQL\ 10 \ bin**

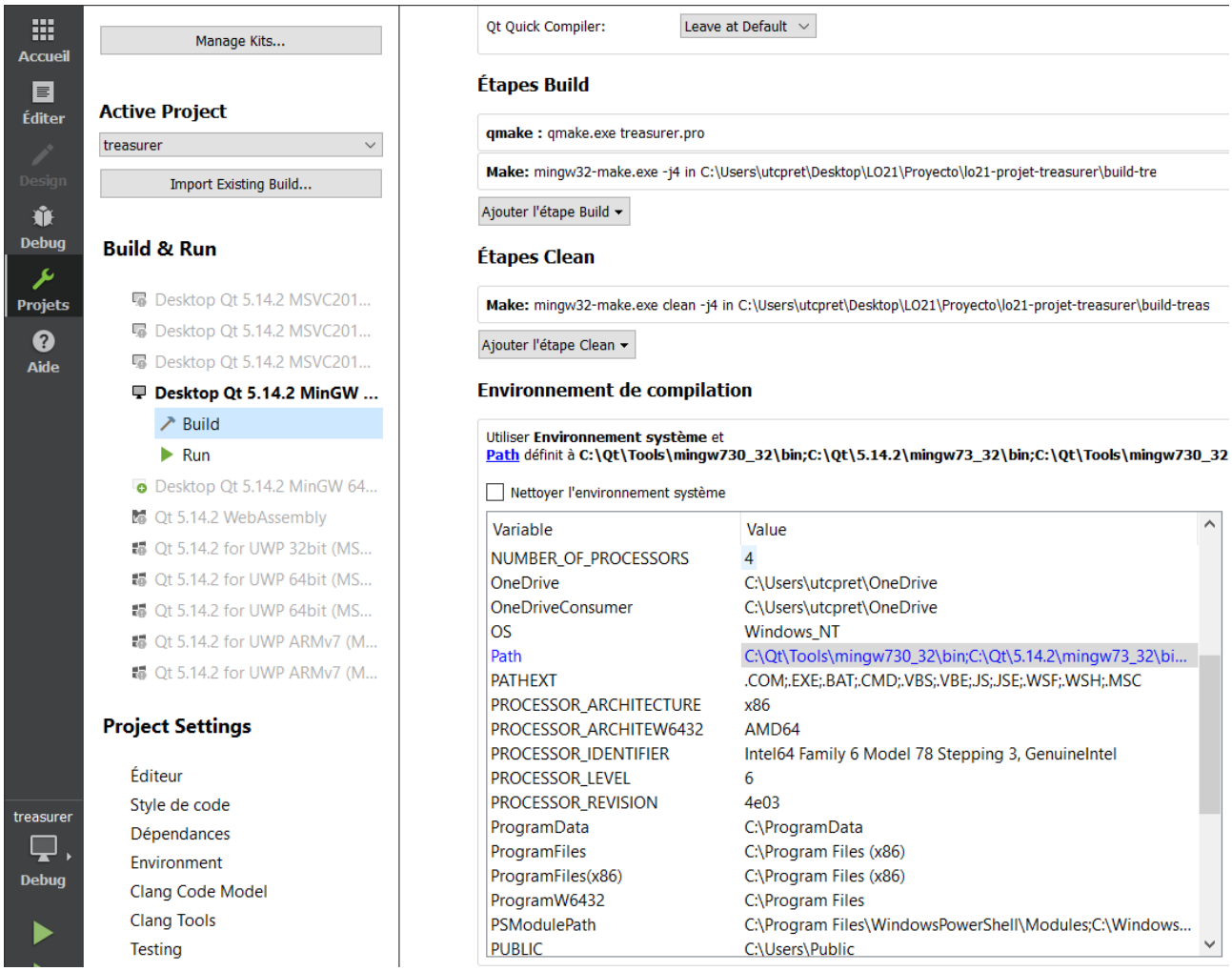

Figura 5: Configuration Build