RAPPORT PROJET BANQUE

Membres de L'équipe : Erwin DZIUBA, Gwenolé BINET, Hugo Weymiens, Romain Bardet

TABLE DES MATIERES

Dans le cadre de notre cours de Projet Applicatif en C++, nous avons entrepris le développement d'un système bancaire innovant nommé "BankStairs". Ce projet vise à créer une plateforme robuste et conviviale pour la gestion des comptes bancaires, les transactions sécurisées, ainsi que la communication avec d'autres banques.

"BankStairs" incarne notre ambition de concevoir un système bancaire moderne répondant aux exigences actuelles du secteur financier, tout en assurant la confidentialité et la fiabilité des opérations.

Dans ce rapport, nous détaillerons les objectifs spécifiques du projet, les fonctionnalités clés que nous avons d'implémentés, ainsi que les étapes de développement réalisées et envisagées pour concrétiser notre vision de "BankStairs".

Qu'est-ce qu'un Système Bancaire Moderne

Un système bancaire moderne est une plateforme informatique avancée conçue pour faciliter les opérations financières et la gestion des services bancaires. Il intègre des technologies innovantes pour offrir des fonctionnalités efficaces, sécurisées et conviviales aux clients et aux institutions financières.

Un système bancaire moderne s'appuie sur les avancées technologiques pour offrir des solutions financières innovantes, sécurisées et accessibles, répondant ainsi aux besoins changeants des clients et des institutions bancaires dans un environnement numérique en constante évolution.

Description du Projet

Objectifs spécifiques du Développement du Système Bancaire

Le projet "BankStairs" vise à développer un système bancaire en utilisant le langage C++ en plusieurs niveaux, en implémentant des fonctionnalités progressivement avancées et en respectant des concepts de programmation orientée objet (POO) ainsi que des bonnes pratiques de développement logiciel.

Niveau 1 : Compte Courant

Front Office

- Authentification : Permettre à l'utilisateur de saisir son identifiant (login) et son mot de passe pour accéder au système.
- Compte Bancaire : Permettre à l'utilisateur de consulter son compte courant.
- Opération Bancaire : Autoriser l'utilisateur à effectuer des crédits ou débits sur son compte courant.

Back Office

- Authentification : Permettre à l'administrateur de se connecter pour gérer les utilisateurs.
- Compte Bancaire : Permettre à l'administrateur de créer de nouveaux utilisateurs et d'associer un compte courant à un utilisateur existant.

Niveau 2 : Multi-Compte et IHM Graphique

Front Office

- Compte Bancaire : Permettre à l'utilisateur de consulter tous ses comptes (courant, livret C, PEL).
- Opération Bancaire : Autoriser l'utilisateur à effectuer des opérations sur ses comptes (crédit, débit, virement).
- Historique : Permettre à l'utilisateur de consulter l'historique de ses opérations.

Back Office

- Compte Bancaire : Permettre à l'administrateur de créer et d'associer des livrets C ou des PEL à un utilisateur existant.
- Historique : Permettre à l'administrateur de consulter l'historique des opérations d'un client.

Niveau 3 : Gestion des Rôles

Front Office

- Authentification : Gérer deux profils utilisateurs : "propriétaire" avec tous les droits, et "conjoint" avec des droits de consultation uniquement.

Back Office

- Compte Bancaire : Permettre à l'administrateur de créer et d'associer des profils (stagiaire, employé, directeur) à des clients.
- Authentification : Gérer trois profils d'authentification (stagiaire, employé, directeur) avec des droits spécifiques selon le profil.

Niveau 4 : La Banque Centrale

Front Office

- Opération Bancaire : Permettre au propriétaire d'effectuer des virements vers d'autres banques disponibles.
- Opération Bancaire : Permettre à la banque destinataire d'autoriser les opérations provenant d'autres banques.

Back Office

- Compte Bancaire : Permettre au directeur de référencer sa banque auprès de la banque centrale.
- Opération Bancaire : Autoriser des opérations interbancaires par le directeur ou un employé.

Ce projet implique l'utilisation avancée des concepts de programmation en C++, y compris l'utilisation des classes pour modéliser les entités, l'héritage et les classes abstraites pour les comptes bancaires, ainsi que l'implémentation d'une interface utilisateur graphique en plus de la version console. De plus, le projet comprend des fonctionnalités de communication avec une API bancaire externe en utilisant des sockets pour l'échange de données JSON.

Fonctionnalités Principales

Interface Utilisateur Console et Graphique : Le système offre aux utilisateurs la possibilité de choisir entre un mode console ou un mode graphique lors de l'exécution de l'application.

En mode console, l'utilisateur peut interagir avec le système via des commandes textuelles dans la console du terminal.

En mode graphique, l'interface utilisateur est basée sur wxWidgets, offrant une expérience visuelle conviviale avec des fonctionnalités interactives telles que des fenêtres, des boutons et des champs de saisie.

Authentification : Le système permet aux utilisateurs de s'authentifier en saisissant leur identifiant (login) et mot de passe, validés par une connexion sécurisée à la base de données.

Consultation des Comptes : Les utilisateurs peuvent visualiser les détails de leur compte courant ainsi que d'autres comptes associés tels que le livret C, PEL et plan boursier. Les informations sont extraites de manière sécurisée depuis la base de données.

Opérations Bancaires : Les utilisateurs peuvent effectuer des opérations financières telles que des crédits, des débits ou des virements entre leurs différents comptes. Les transactions sont gérées de manière sécurisée avec des contrôles d'autorisation en fonction de la solde et des limites.

Historique des Opérations : Les utilisateurs ont accès à un historique détaillé de leurs transactions passées, permettant ainsi de suivre l'évolution de leurs comptes et de leurs finances.

Gestion des Utilisateurs : L'administrateur peut créer de nouveaux utilisateurs, attribuer des profils (comme stagiaire, employé, directeur) et associer des comptes courants, livrets C ou PEL à chaque utilisateur. Cette fonctionnalité implique des interactions avec la base de données pour gérer les informations utilisateur de manière efficace.

Gestion des Profils d'Authentification : Le système prend en charge la définition de différents profils d'authentification (comme "propriétaire" avec tous les droits, "conjoint" avec des droits limités, ou des profils spécifiques pour les employés), avec des autorisations et des restrictions adaptées à chaque profil.

Gestion des Rôles : Les administrateurs peuvent attribuer des rôles spécifiques à chaque utilisateur, définissant ainsi les droits et les privilèges associés à chaque profil. Cela inclut la gestion fine des autorisations pour assurer la sécurité et la confidentialité des données.

CONTRAINTES TECHNIQUES

Choix du Framework GUI : WxWidgets

Pourquoi WxWidgets ?

Compatibilité Multiplateforme : WxWidgets offre une excellente compatibilité avec divers systèmes d'exploitation tels que Windows, Linux et macOS. Cette capacité multiplateforme est alignée avec notre objectif de rendre le NIDS accessible sur divers environnements.

Langage de Programmation : WxWidgets est conçu pour être utilisé avec C++, qui est le langage de programmation choisi pour notre projet. Cette compatibilité directe simplifie le processus de développement et d'intégration.

Personnalisation et Flexibilité : Le framework fournit une grande flexibilité en termes de personnalisation de l'interface utilisateur, ce qui est crucial pour créer une expérience utilisateur intuitive et répondre aux exigences spécifiques de notre projet.

Riche Ensemble de Fonctionnalités : WxWidgets offre un large éventail de widgets et d'outils qui facilitent le développement d'une interface utilisateur complète et fonctionnelle, essentielle pour la visualisation et la gestion des différentes fonctionnalités d'un système bancaire.

Performance : WxWidgets est connu pour son efficacité en termes de performances, un aspect critique pour notre projet bancaire qui doit fonctionner en temps réel sans consommer excessivement de ressources système.

Communauté et Support : WxWidgets possède une communauté active et un bon support, offrant une aide précieuse pour le dépannage et l'optimisation de notre application.

Comparaison avec d'autres Framework GUI

Qt : Bien que Qt soit également multiplateforme et populaire, il est moins orienté vers C++ natif par rapport à WxWidgets. De plus, Qt utilise un système de build QMake distinct, tandis que WxWidgets s'intègre plus naturellement avec CMake, utilisé dans notre projet.

GTK+ : GTK+ est une autre alternative populaire, surtout sur les systèmes basés sur Linux. Cependant, GTK+ n'offre pas le même niveau d'intégration native avec C++ que WxWidgets. De plus, GTK+ peut présenter des défis en termes de look uniforme et de convivialité sur différentes plateformes, contrairement à WxWidgets qui assure une apparence cohérente sur tous les systèmes d'exploitation.

DEVELOPPEMENT

Environnement et Outils de Développement

Le projet BankStairs a été développé en utilisant un ensemble d'outils et d'environnements de développement modernes et efficaces pour garantir la qualité et l'efficacité du processus de développement.

Visual Studio : Utilisé comme environnement de développement intégré (IDE) principal. Il a offert une interface puissante pour la codification, le débogage, et la gestion de version.

GitHub : Employé pour le contrôle de version et la collaboration au sein de l'équipe. Il a permis de suivre les modifications, de gérer les branches et de faciliter la révision du code entre les membres.

WxWidgets : Utilisé pour le développement de l'interface utilisateur. Ce framework a permis de créer une interface graphique multiplateforme et réactive.

SQLite : Utilisé pour le stockage des informations dans la base de données. SQLite est un système de gestion de base de données relationnelle embarqué, léger et efficace, idéal pour nos besoins de stockage de données dans l'application bancaire.

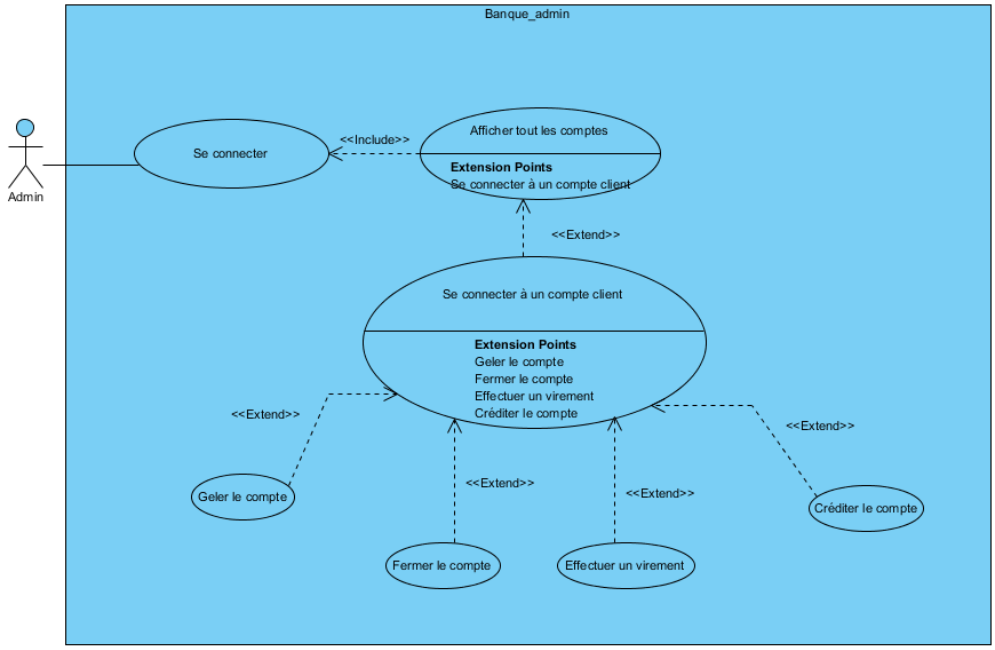
Documentation : La documentation du code et des fonctionnalités a été une partie essentielle du processus, assurant la clarté et la maintenabilité du projet.

ChatGPT : Utilisé comme outil d'assistance pour la génération rapide de code, la résolution de problèmes et l'obtention de conseils de développement.

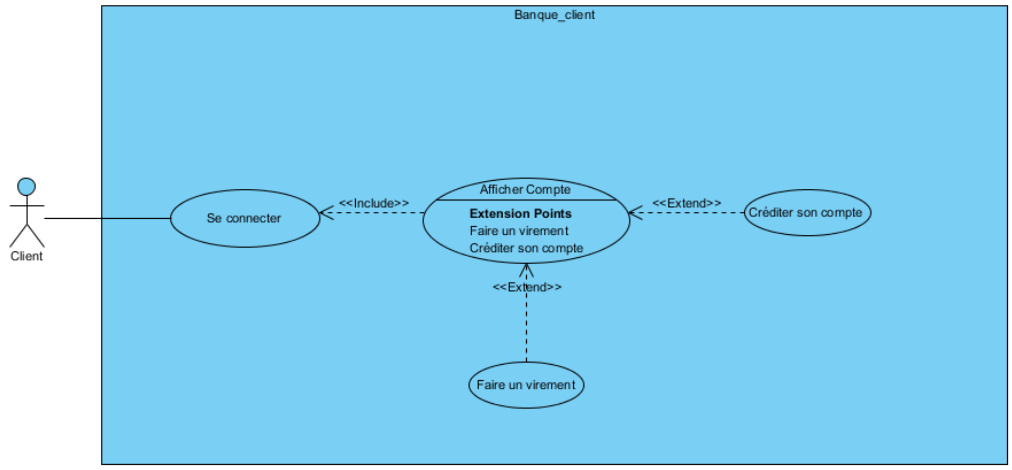
DOCUMENTATION TECHNIQUE ET LE GUIDE UTILISATEUR

ARCHITECHTURE :

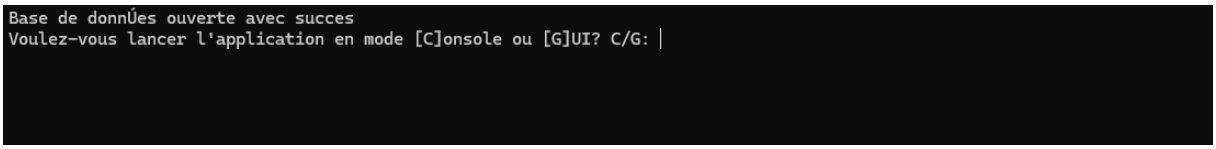
Uses case du back office :



Uses cases front office :

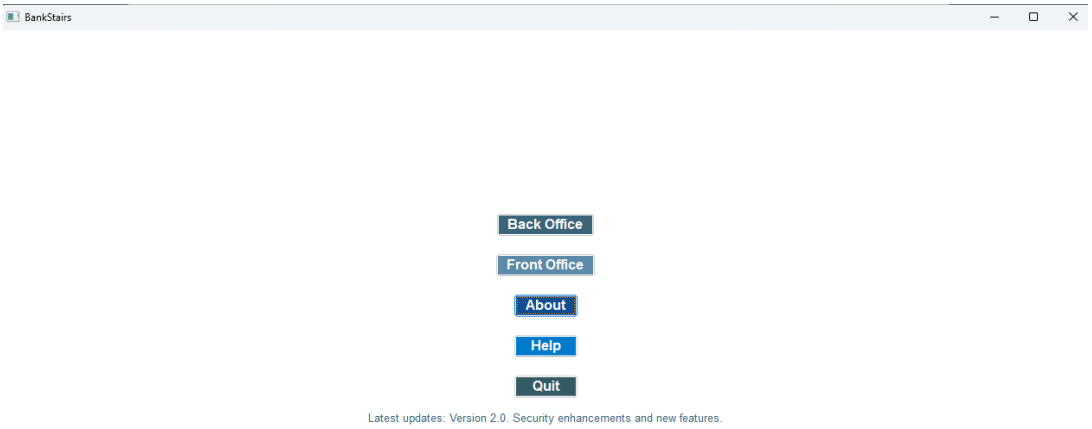


Choisir entre mode graphique et console. Cliquer sur « C » pour console et « G » pour le mode graphique.

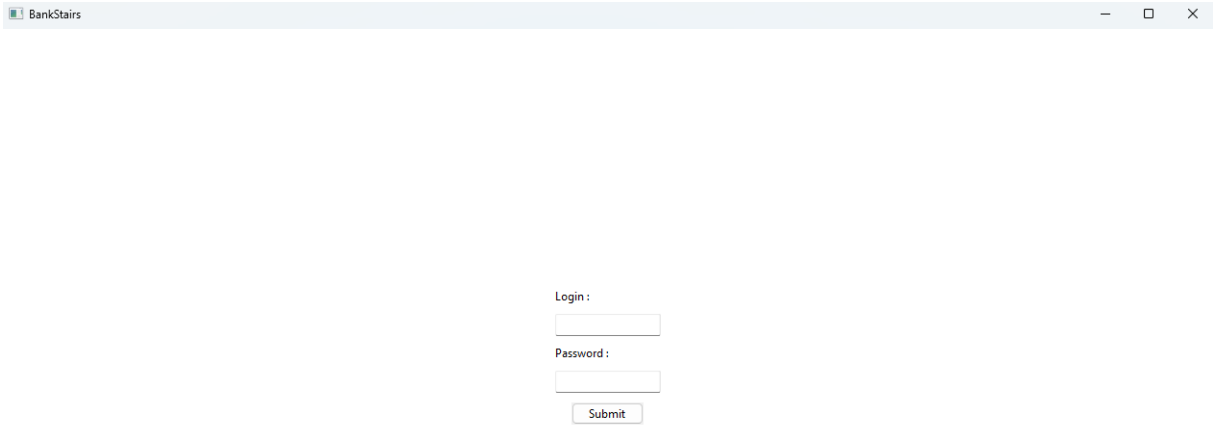


Mode graphique

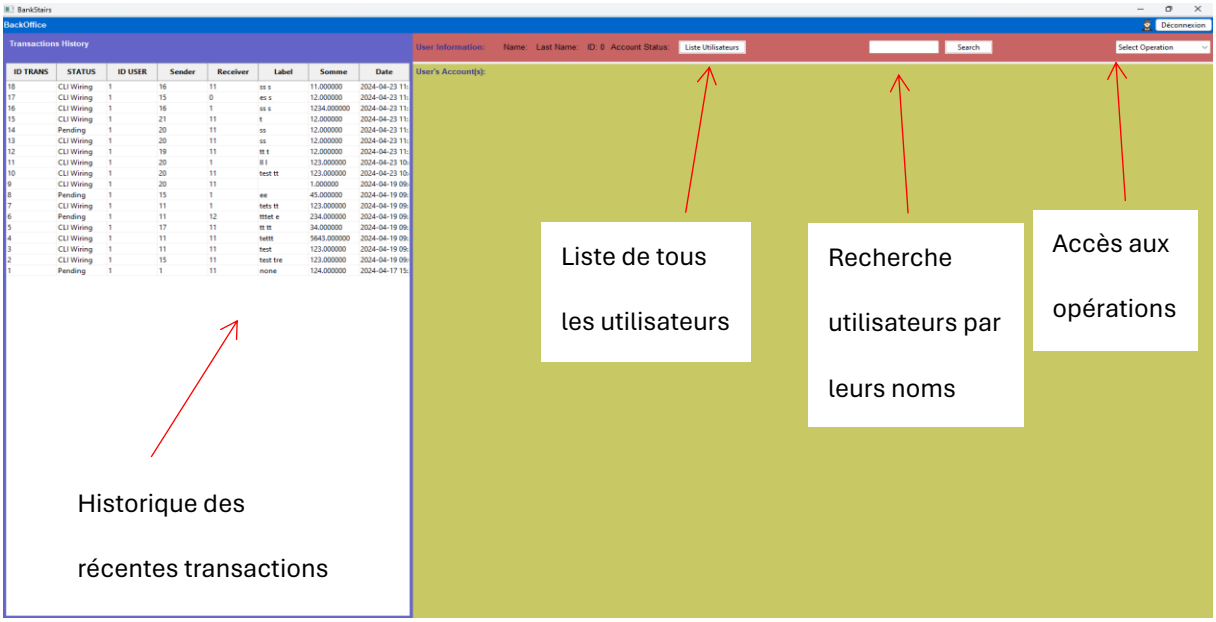
On dispose ici de 5 choix. « Back Office » nous permet de nous connecter au backoffice. Le choix « front office » nous permet de nous connecter au front office. « About » pour en savoir plus sur l'application. « Help » redirige vers un site html d'aide. « Quit » Pour quitter l'application.



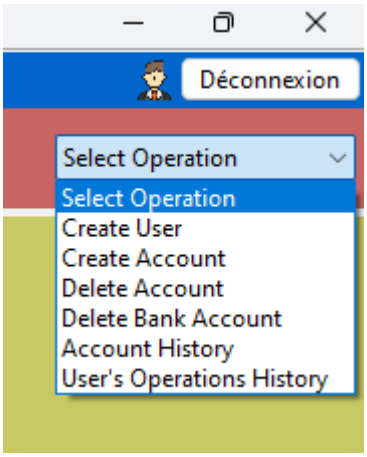
Après avoir cliqué soit sur front office ou back office, on arrive sur ce menu de connexion.

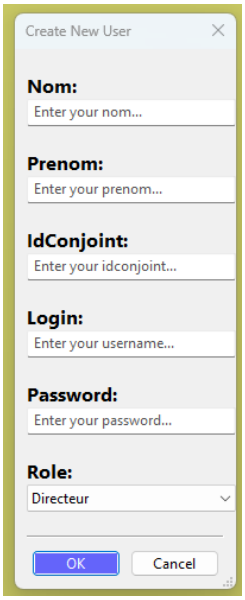


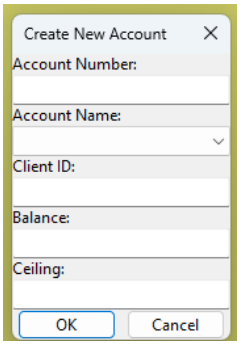
Backoffice : voici la page d'accueil du backoffice : (penser à bien mettre en plein écran)

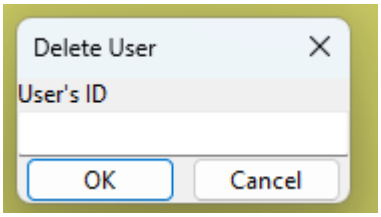


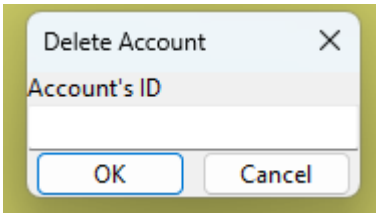
Les opérations :

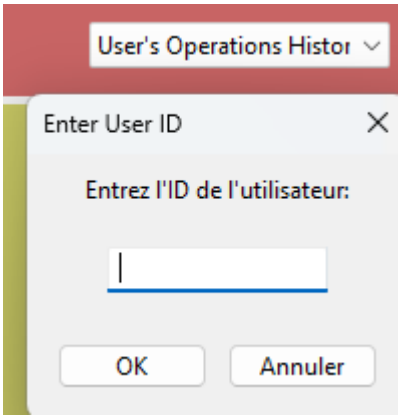


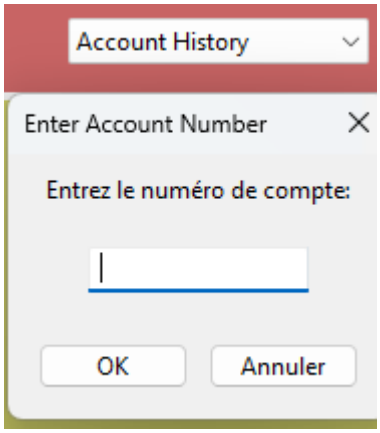




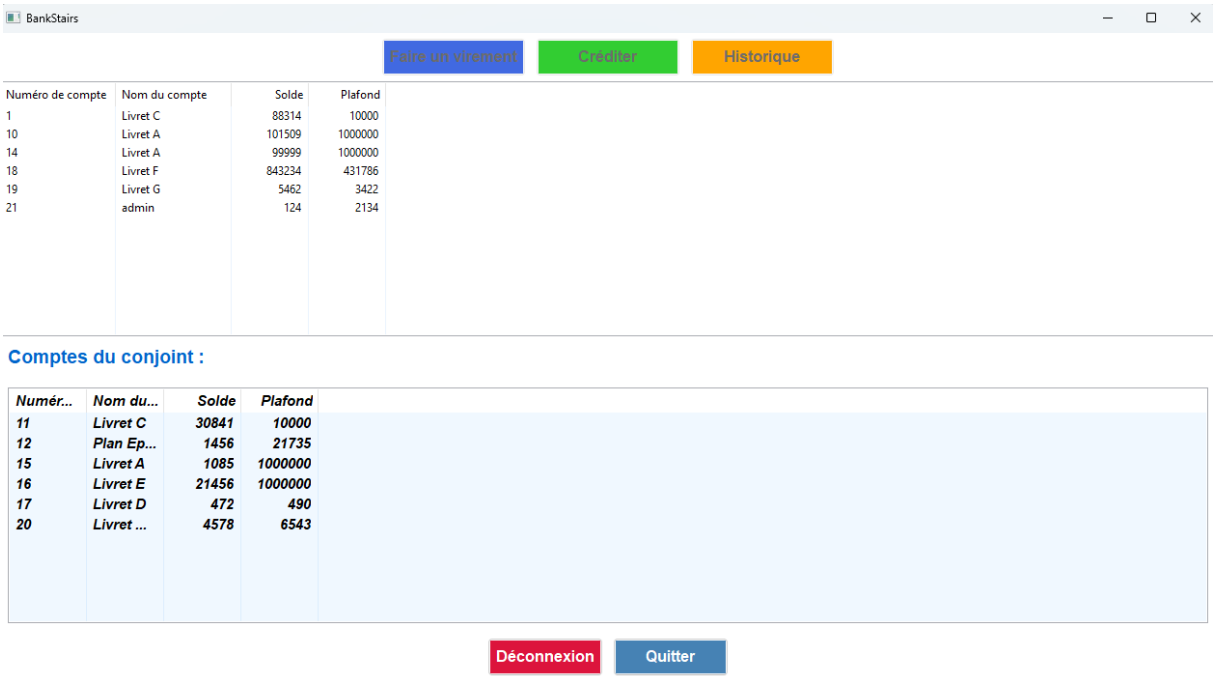




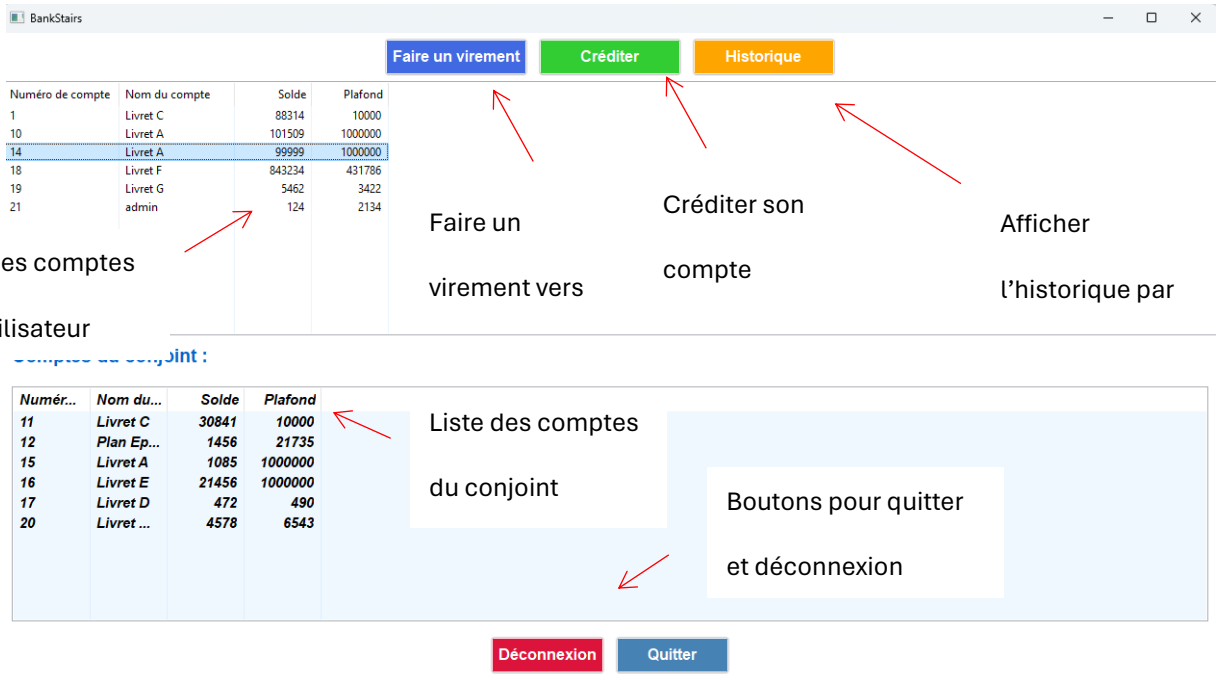




Front Office :



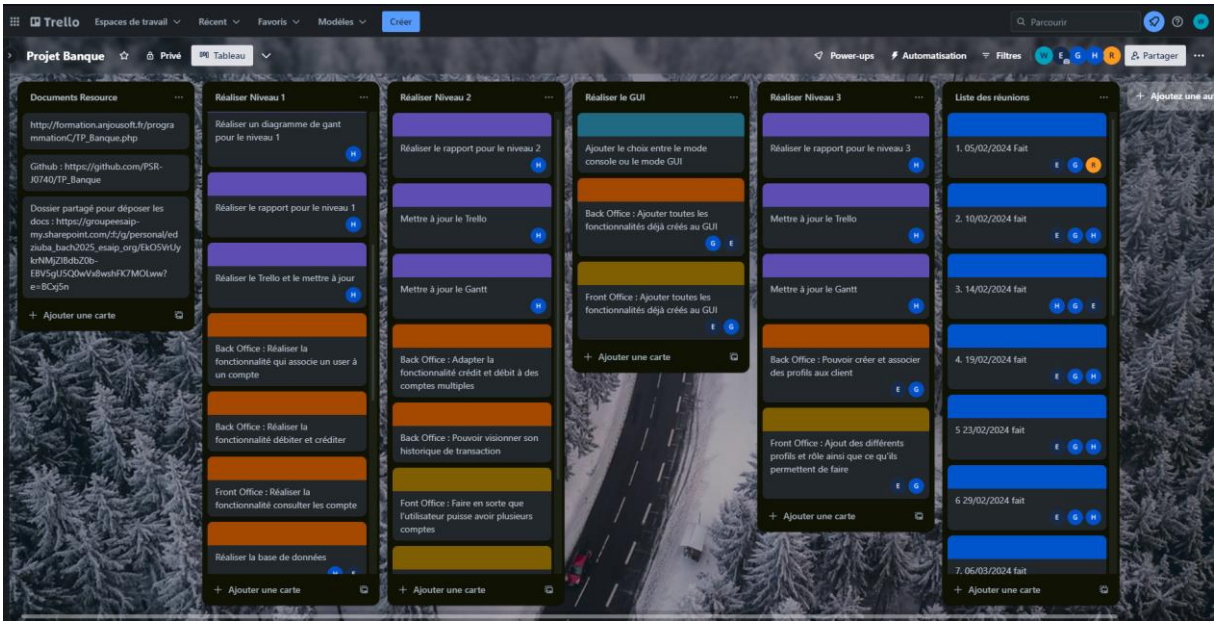
Fonctionnalités :



GESTION DE PROJET

Gestion de Projet avec Trello

Notre projet de développement d'un système bancaire a été rigoureusement planifié et géré à l'aide de l'outil Trello, un gestionnaire de tâches collaboratif en ligne qui nous a permis d'organiser les étapes du projet en différents tableaux et listes pour une meilleure visibilité et suivi des tâches.



Organisation et Planification

Nous avons décomposé le projet en phases clés et en différents niveau en lien avec les informations reçu, représentées par des colonnes distinctes dans Trello :

Tâches réalisées : Un historique des tâches complétées, assurant la traçabilité du travail accompli.

Tâches à faire : Ce qui reste à accomplir, y compris l'optimisation du code et la mise à jour de Trello.

Membres du groupe : Les rôles de chacun au sein de l'équipe, incluant le chef de projet Erwin et les développeurs, garantissent une répartition claire des responsabilités.

Les Réunions : Planification temporelle des séances de travail, aidant à coordonner les efforts de l'équipe et à respecter les délais.

Liens des documents : Centralisation des ressources, comme la présentation PowerPoint, pour un accès facile et partagé.

Répartitions des Tâches

Chaque carte sur le tableau Trello représente une tâche spécifique attribuée à un ou plusieurs membres de l'équipe, avec des détails sur la tâche et son statut actuel. Cette méthode nous a permis de suivre les progrès en temps réel et de réagir rapidement aux imprévus, tout en favorisant la collaboration et la transparence au sein de l'équipe.

Résumer pour chaque membre du groupe ce qu'il a fait :

Erwin (chef) :

- Organisation du groupe, décision quels outils utilisé, répartir les tâches etc
- développement de la partie graphique wxwidgets
- développement fonctions
- conception de la base de données
- rédaction rapport
- Powerpoint

Gwenolé :

- développement de la partie console
- développement fonctions
- conception de la base de données

Hugo :

- création des schéma UML
- maintenance du trello
- création du gant
- rédaction du rapport
- aide dans le développement

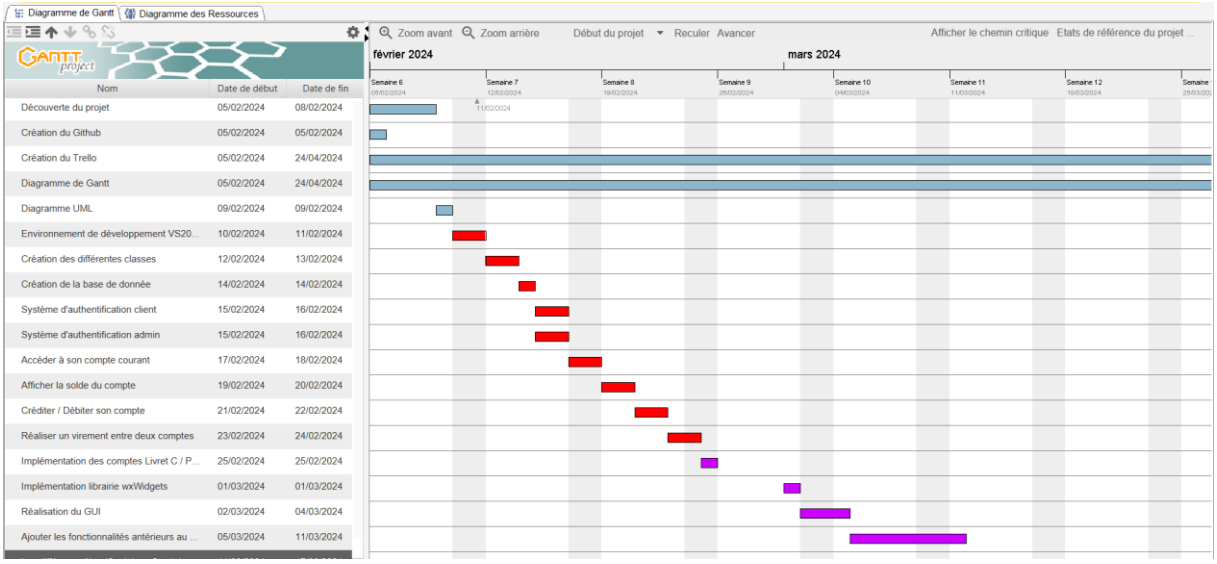
Romain :

- rien

(Jamais présent)

Gestion de Projet avec Gant

Un Gant a également été créé afin de s'organiser et d'avoir une trace de toutes les étapes réalisées dans le temps.



Résumé des Réalisations et Résultats Obtenus

Durant ce projet nous avons pu concevoir et développer une application fonctionnelle qui répond à la majorité des attentes du cahier des charges. Nous avons beaucoup appris durant ce projet, que ce soit dans la gestion de projet ou du développement en c++. Nous vous invitons à consulter la démonstration, Voir le dossier demo dans le dossier pour accéder à la démonstration (fichier .exe)

GITHUB

Description du Dépôt :

Le code source de notre projet est hébergé sur GitHub, une plateforme de gestion de développement logiciel qui facilite la collaboration et le partage de code. Le dépôt contient l'intégralité du code source du projet, les fichiers de configuration, les documentations techniques, ainsi que les guides d'utilisation.

https://github.com/PSR-J0740/TP_Banque

RESSOURCES SUPPLEMENTAIRES ET REFERENCES

Documentation Officielle

WxWidgets Documentation : Accédez à la documentation officielle pour comprendre en profondeur les composants WxWidgets utilisés dans le projet. https://docs.wxwidgets.org/3.0/

Tutoriels et Guides

WxWidgets Programming Tutorial : Des tutoriels pour commencer avec la programmation WxWidgets. https://www.wxwidgets.org/docs/tutorials/

Mettre en place l'environnement wxwidgets sur visual studio : https://www.youtube.com/watch?v=1fZL13jIbFQ

Forums et Communautés

Stack Overflow ; une plateforme de questions-réponses pour les développeurs : https://stackoverflow.com/

WxWidgets Forum ; Un forum pour les développeurs WxWidgets pour obtenir de l'aide et partager des connaissances. https://forums.wxwidgets.org/