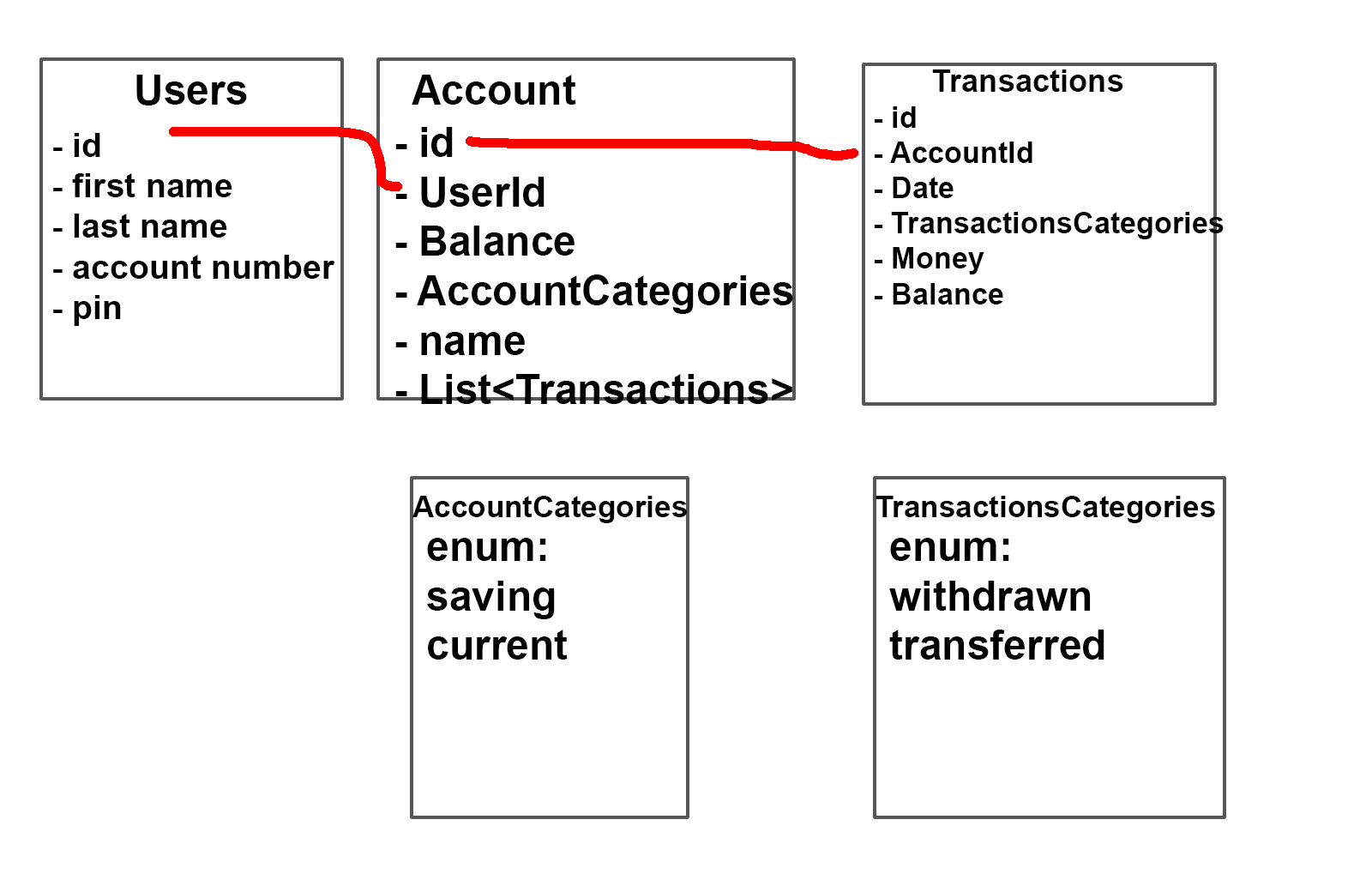
**Individual Assessment #1 - Banking Application**

1. **First step: The structure**

J’ai commencé le projet par suivre des tutoriels sur Youtube pour mieux comprendre comment faire un projet de ce type. Pendant ce temps de travail j’ai créé mes premiers models, controllers et views. J’ai fait les bases avec une classe : AppUser, Transaction et BankAccount ainsi que des enumérateurs pour les types de transactions et de compte bancaire.

Pour cela j’ai créé plusieurs dossiers :

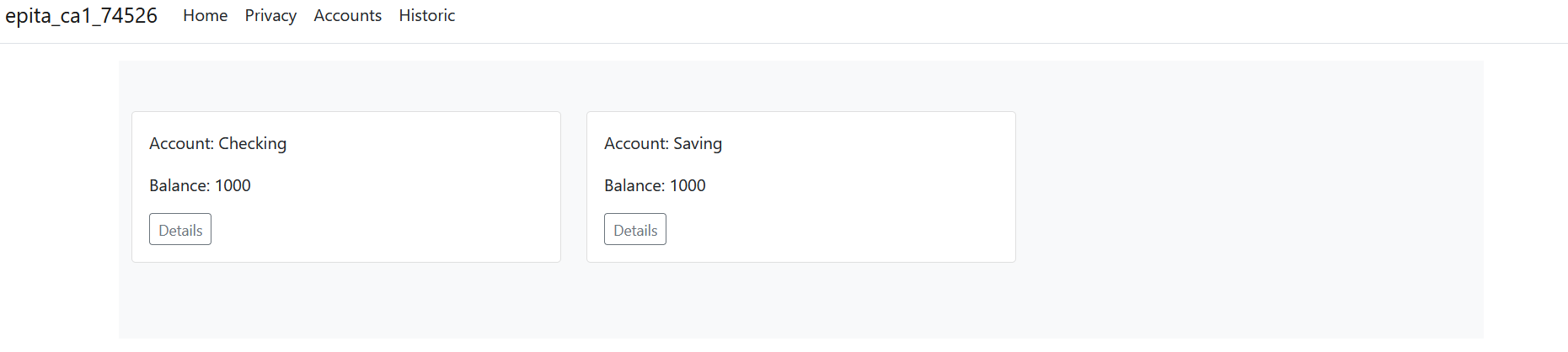
* Models qui contient tous les models
* Data avec un dossier Enum qui contient tous les enumérateurs

J’ai également créé le contexte de ma base de données et une seed pour celle-ci (ce qui m’a permis de tester plus simplement mon application) et je les ai rangés dans le dossier Data. J’ai aussi fait des schémas pour pouvoir avoir une idée de comment agencer les différentes classes, voilà donc comment je voyais l’agencement des 3 premières classes au départ : 

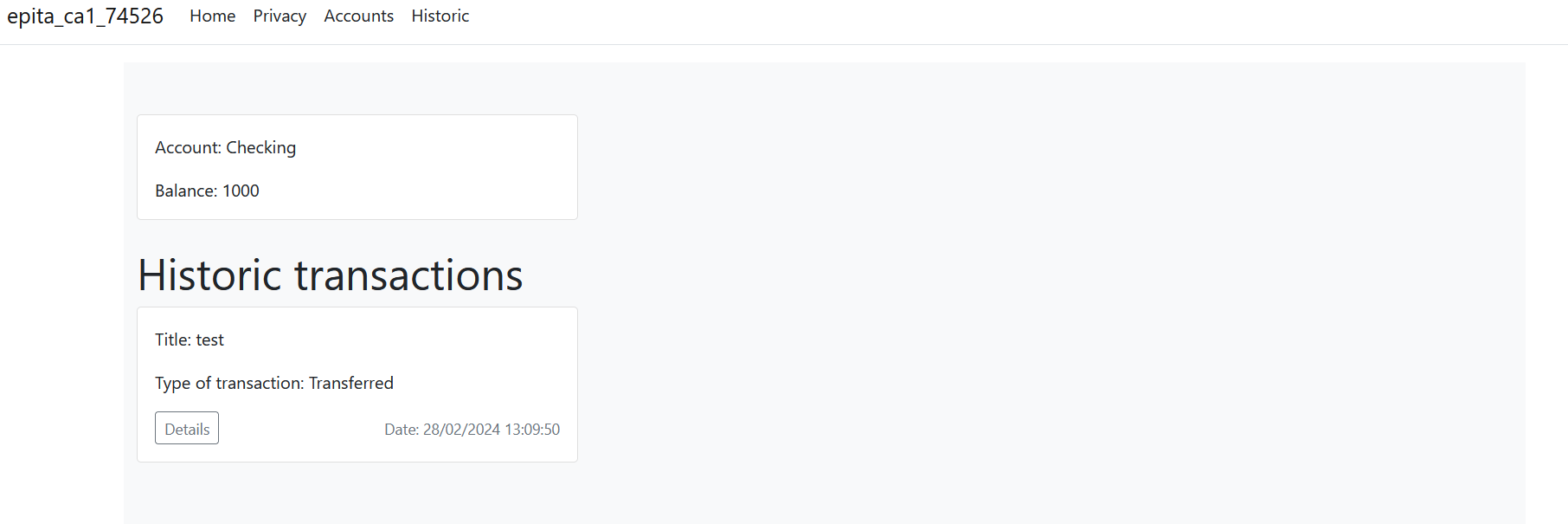
Au tout départ je n’ai pas implémenté le fait d’avoir des comptes (utilisateurs et administrateur). Je voulais d’abord me concentrer sur la structure c’est-à-dire créer des comptes bancaires et les afficher, de même pour les transactions.

Mon application à ce stade ressemblait à cela :

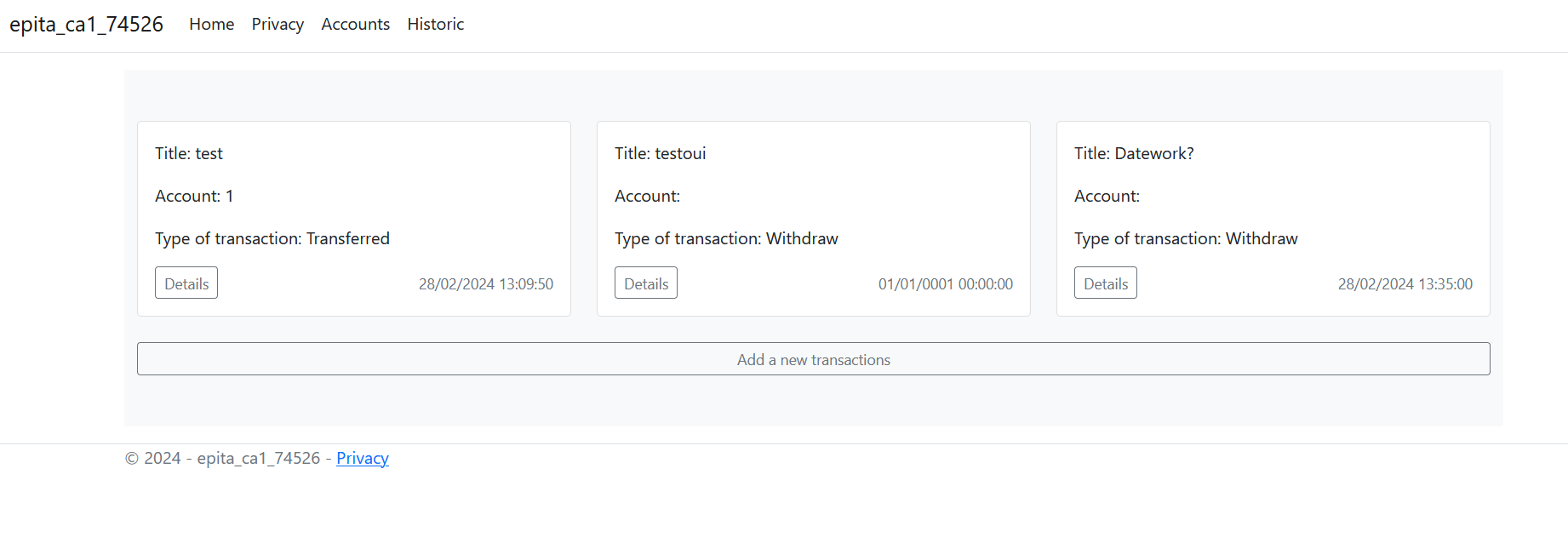
1. Une page accounts ou l’on pouvait accéder à tous les comptes bancaires créer.



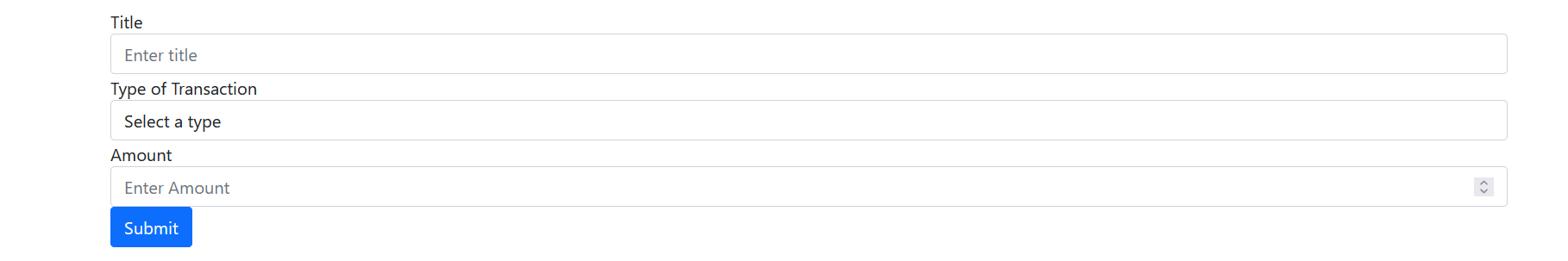
1. Le bouton détail de la page Accounts menait vers plus de précision sur le compte en question avec un historique des transactions du compte choisi.



1. La page Historic permettait de voir toutes les transactions faites peu importe le compte concerné.



1. Et enfin il était aussi possible d’ajouter une nouvelle transaction depuis la page Historic.



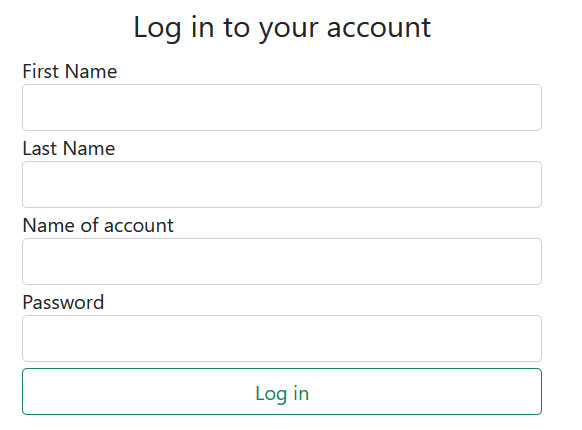
Pour faciliter la récupérations et l’ajout des comptes bancaires et des transactions j’ai mis en place des interface IAccountRepository et IBankAccountRepository rangé dans le dossier interfaces avec les repository héritant de ces classes rangé dans le dossier repository.

Après avoir fait ça j’avais accompli la première partie de mon application et j’avais poser les bases pour la suite. Il était temps d’entamer la deuxième étape qui étaient celle de la création des comptes utilisateurs et administrateurs.

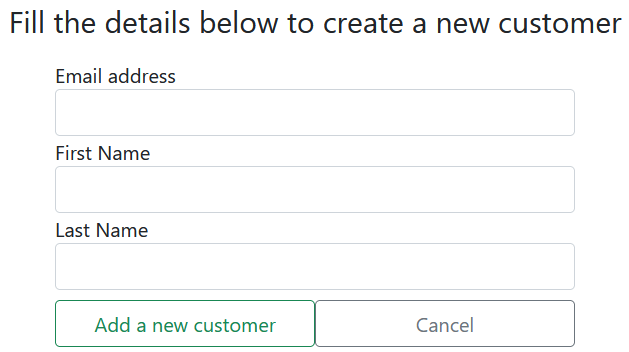
1. **Second step: Connected the authentication to all the precedent method**

Pour cela j’ai utilisé IdentityFramework. J’ai donc créé une classe UserRoles que j’ai placé dans data avec deux rôles : User et Admin. J’ai créer deux nouvelle pages « vue », login et register. J’ai découvert en faisant ça les viewModels, au tout départ je n’avais pas trop saisi leurs intérêts mais par la suite j’ai trouvé ça très pratique et les utilisés dans quasiment toutes mes futurs pages.

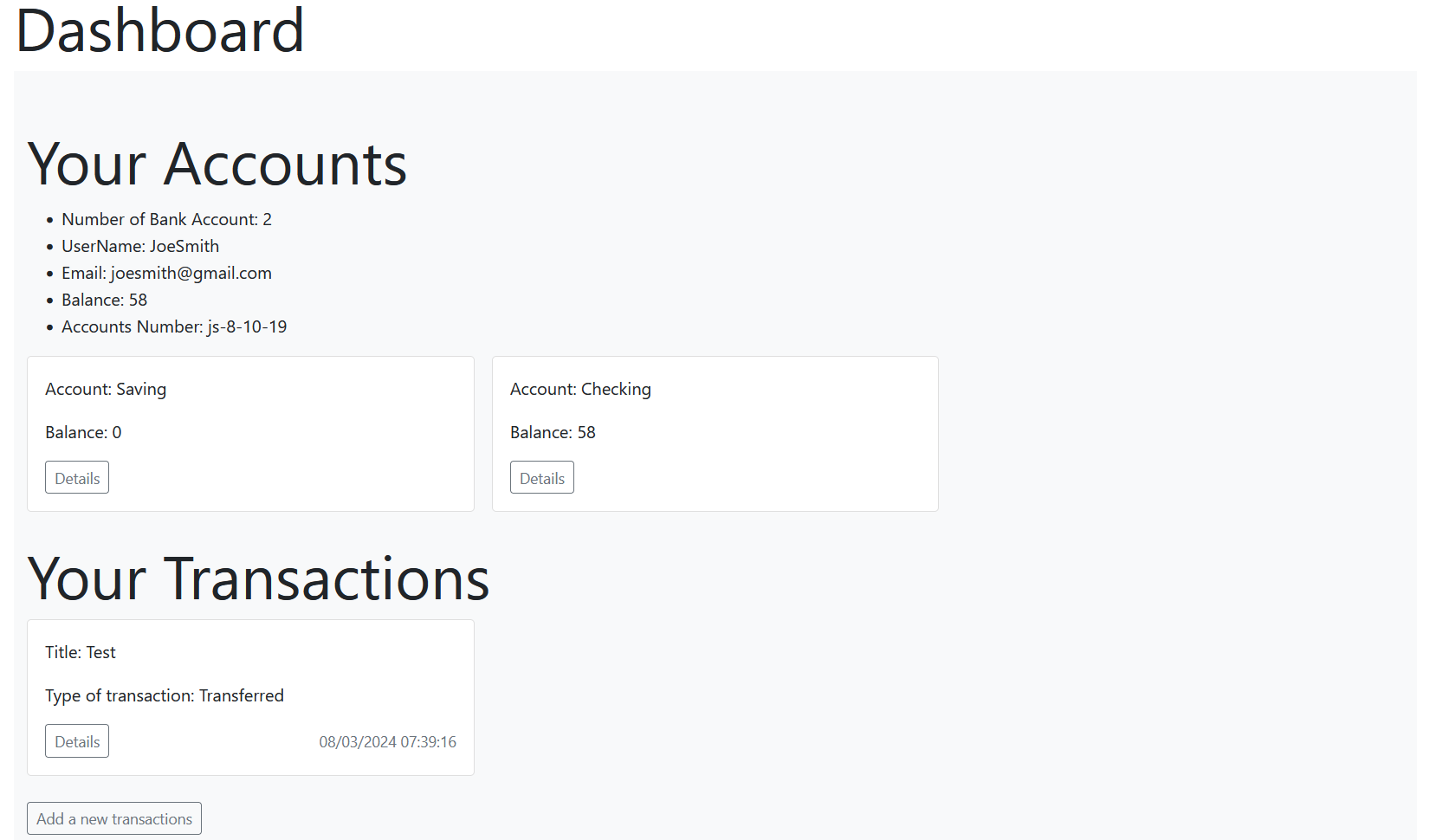
Quand j’ai créé ma page login je n’ai tout d’abord pas pris en compte le fait de pouvoir choisir entre customer et bank employee donc voilà à quoi ressemblait la page aux premiers abords :



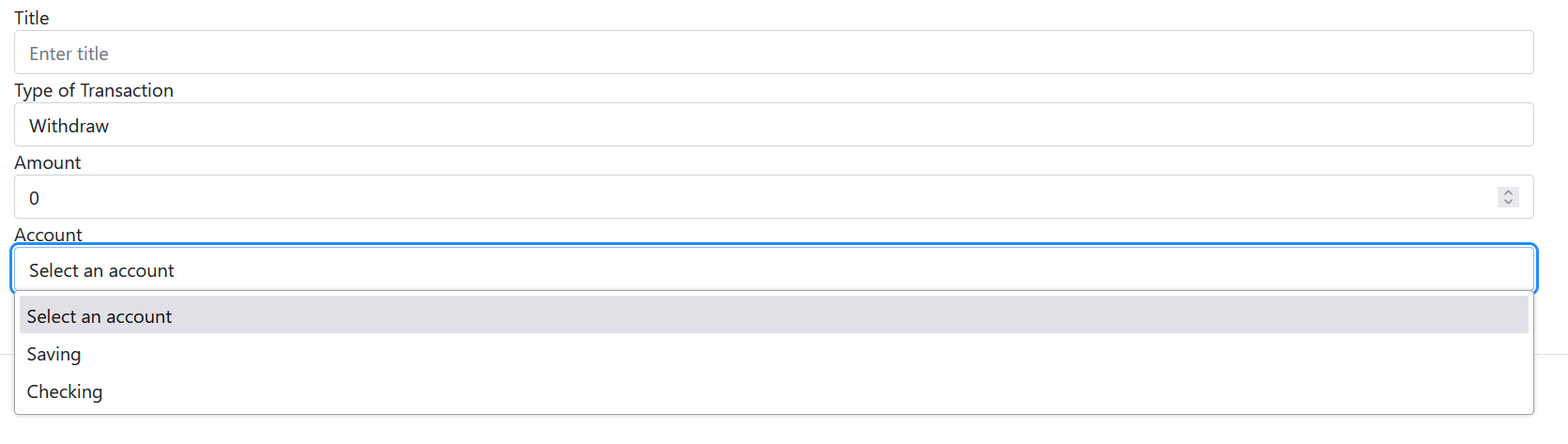
La page register permettait d’enregistrer un nouveau compte peu importe si nous étions déjà connectés ou non.



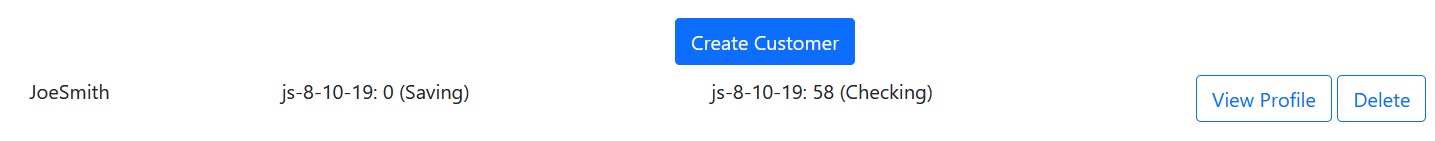
Une fois la logique de connexion mise en place avec la page login, j’ai construit une page appelée dashboard qui permet à l’utilisateur de lister toutes ces informations. Cette page reprend les mêmes méthodes que précédemment avec en plus l’informations de l’id de l’utilisateur pour récupérer uniquement que ce qui lui concerne.



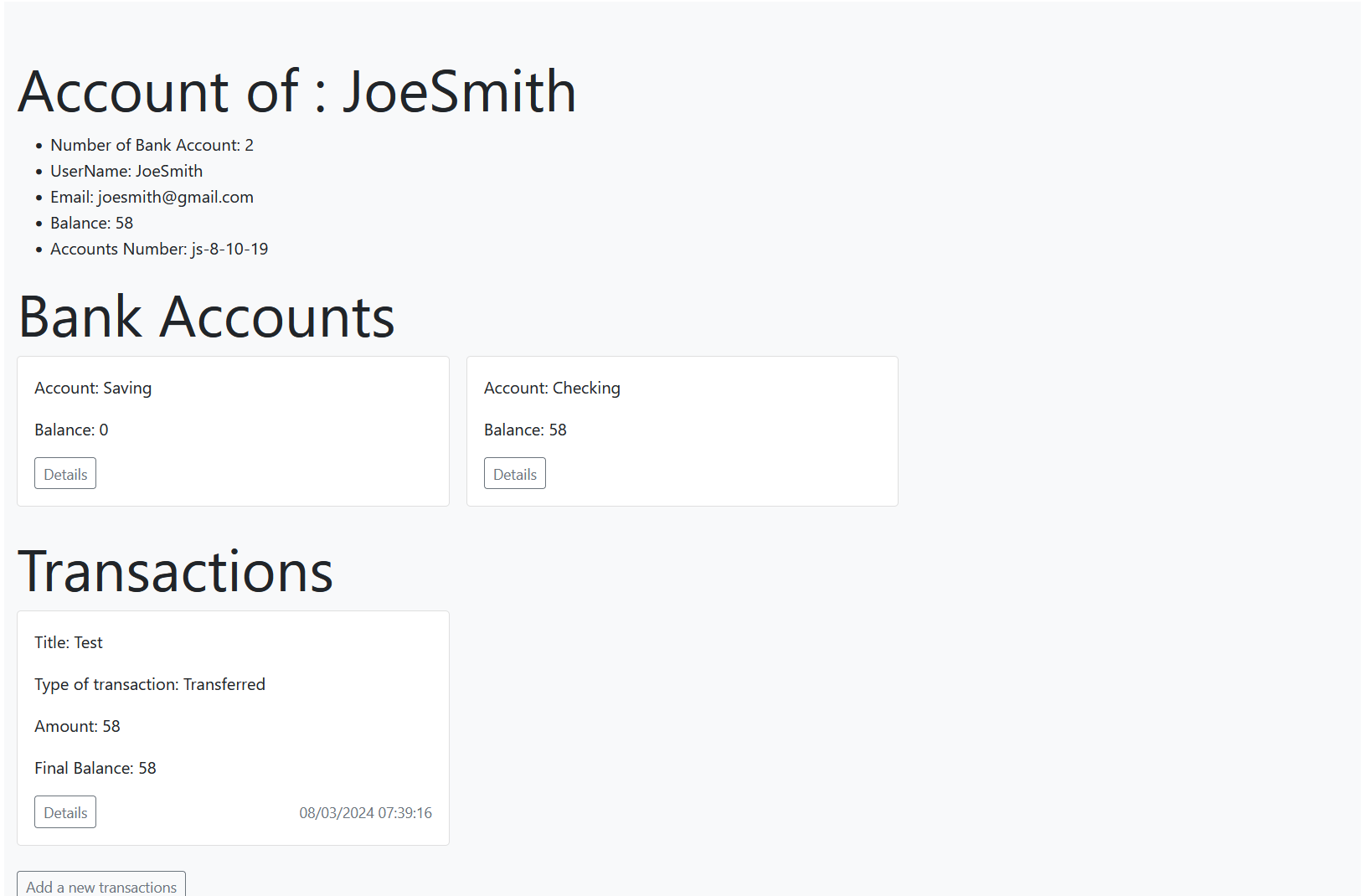
J’ai relié cette page avec cette faite précédemment (Details pour les transactions et compte bancaires et ajouter une nouvelle transaction). J’ai d’ailleurs modifié la page pour ajouter une nouvelle transaction pour que l’utilisateur puisse choisir sur lequel de ces comptes il veut faire la transaction.



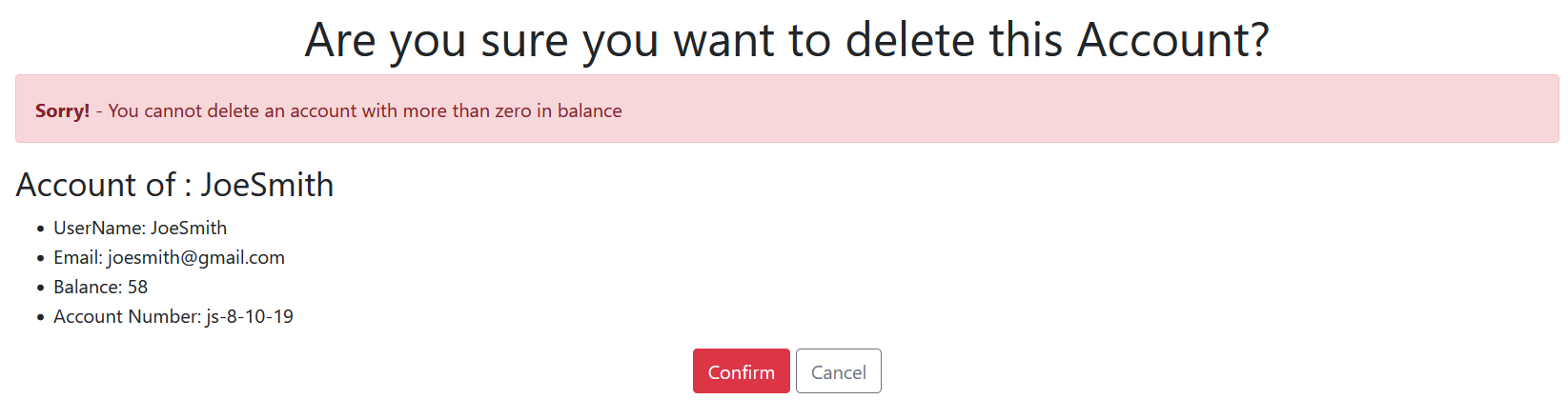
Après avoir fini la vue du customer j’ai construis celle du bank employee. Pour cela j’ai fait une page User Lists sur lequel il peut voir l’username et les comptes de l’utilisateurs avec les balances respectives. Il peut également accéder à son profile où le supprimer.



En appuyant sur view profile il obtiendra presque la même chose que l’utilisateur obtient dans son dashboard :

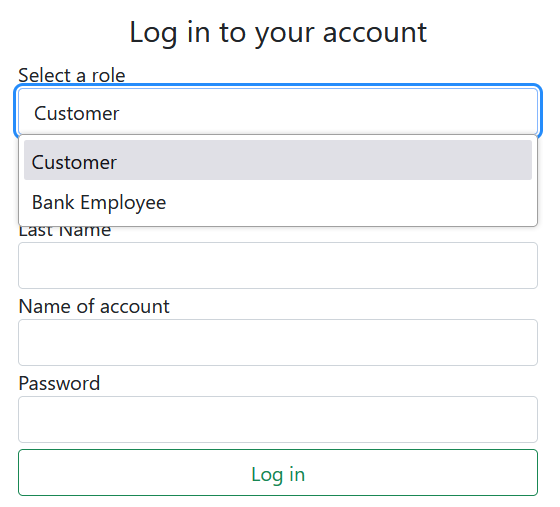


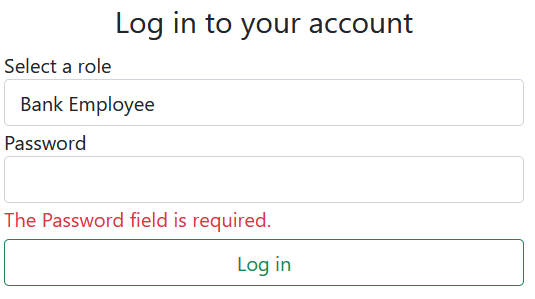
S’il appuie sur delete et confirm, et que l’utilisateur concerné n’a pas zero de monnaie alors une erreur va s’afficher :



Pour faciliter la mise en place de ces pages j’ai comme précédemment fais des interfaces et reposository. Ici un pour les Users et un pour le dashboard. Les ViewModel m’ont également été très utiles pour encapsuler différentes donnés et afficher tout ce que j’avais besoin.

A ce stade l’application était donc presque finie ! J’ai ajouté la possibilité de choisir entre bank employee ou customer lors du login :





Pour finir j’ai ajouté des protections pour ne pas qu’un utilisateur non connecté ou un non employee de bank puisse accéder à des pages qu’il ne devrait pas. L’application était donc fin prête !

Au final voici à quoi ressemble l’uml de mes models :