Scala projet - ToDo

1. Introduction

Ce projet a été réalisé en 2019 pendant le cours “Programmation appliquée en Scala” à l'HEIG-VD d'Yverdon-les-bains. Ce projet a été réalisé par trois étudiants: **Caduff Max**, **Nanchen Lionel**, **Nicole soit Nicoulaz Olivier**.

Ce projet consiste en la création d’une application qui utilise Scala Play pour le back-end, Slick pour la base de données et est composée d’un front-end libre.

Nous avons créé une application de gestion de tâches. L'application permet à l'utilisateur de créer, terminer et supprimer des tâches.

Une tâche est composée d'un titre, d'une description (optionnelle), d'une date d'échéance (optionnelle) et d'un état qui définit si la tâche est en cours ou terminée.

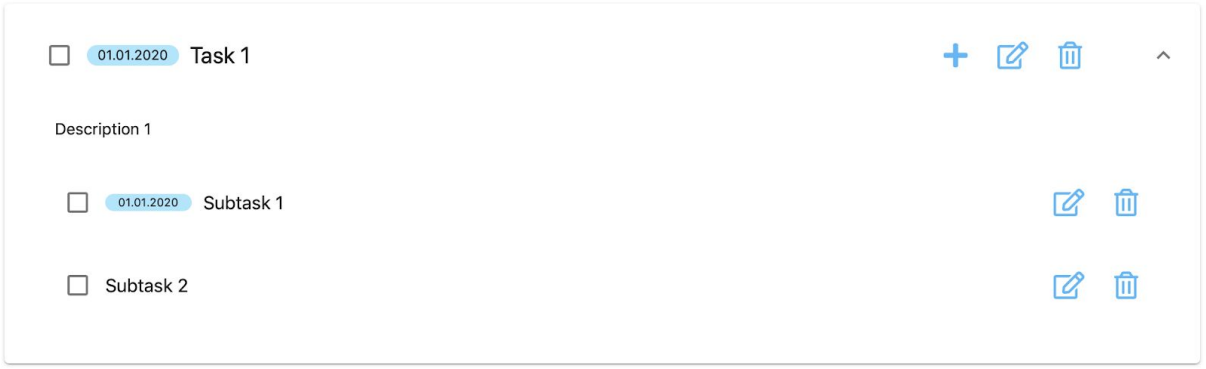
2. Implémentation

2.1. Front-end

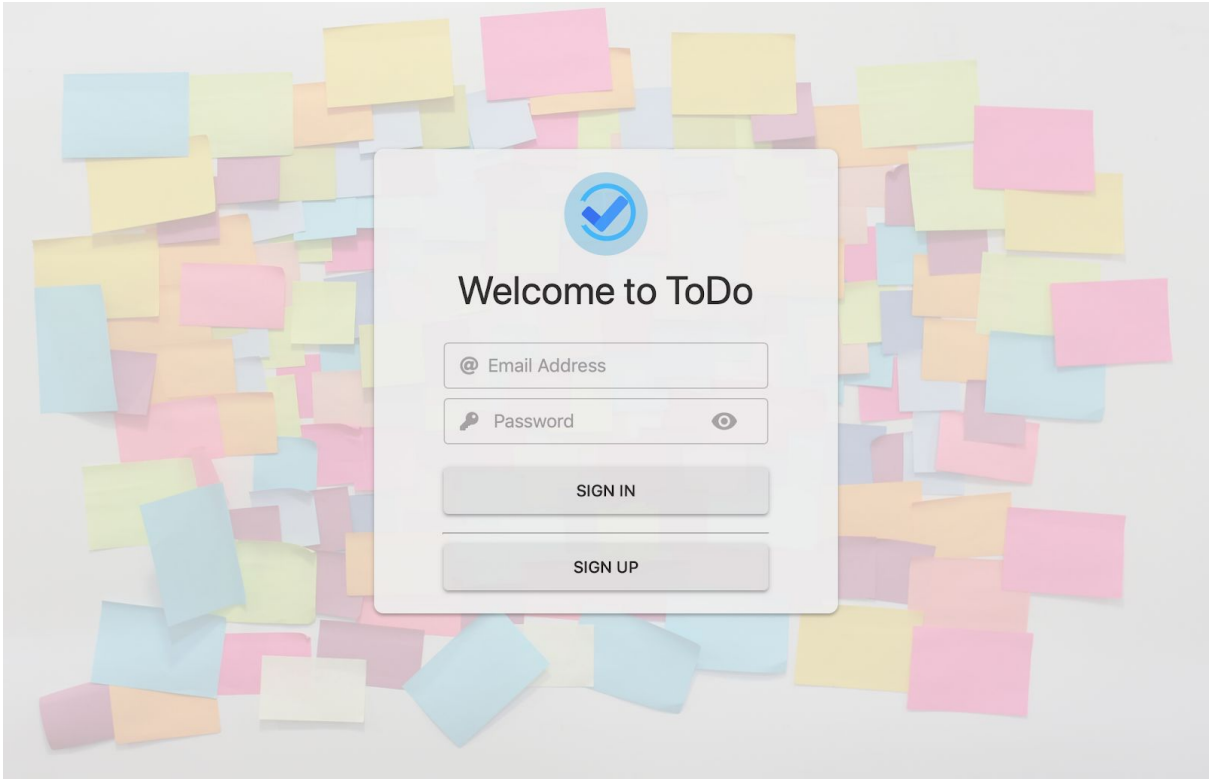
Le front-end a été réalisé en Javascript avec l’aide de la librairie React et l'utilisation de la librairie libre Material-UI.

Le front-end se veut très simple, claire et facile à utiliser. Il est constitué d’une page qui présente premièrement un outil qui permet de créer une nouvelle tâche. Ensuite, toutes les tâches courantes sont affichées. Finalement, les tâches terminées sont présentées.

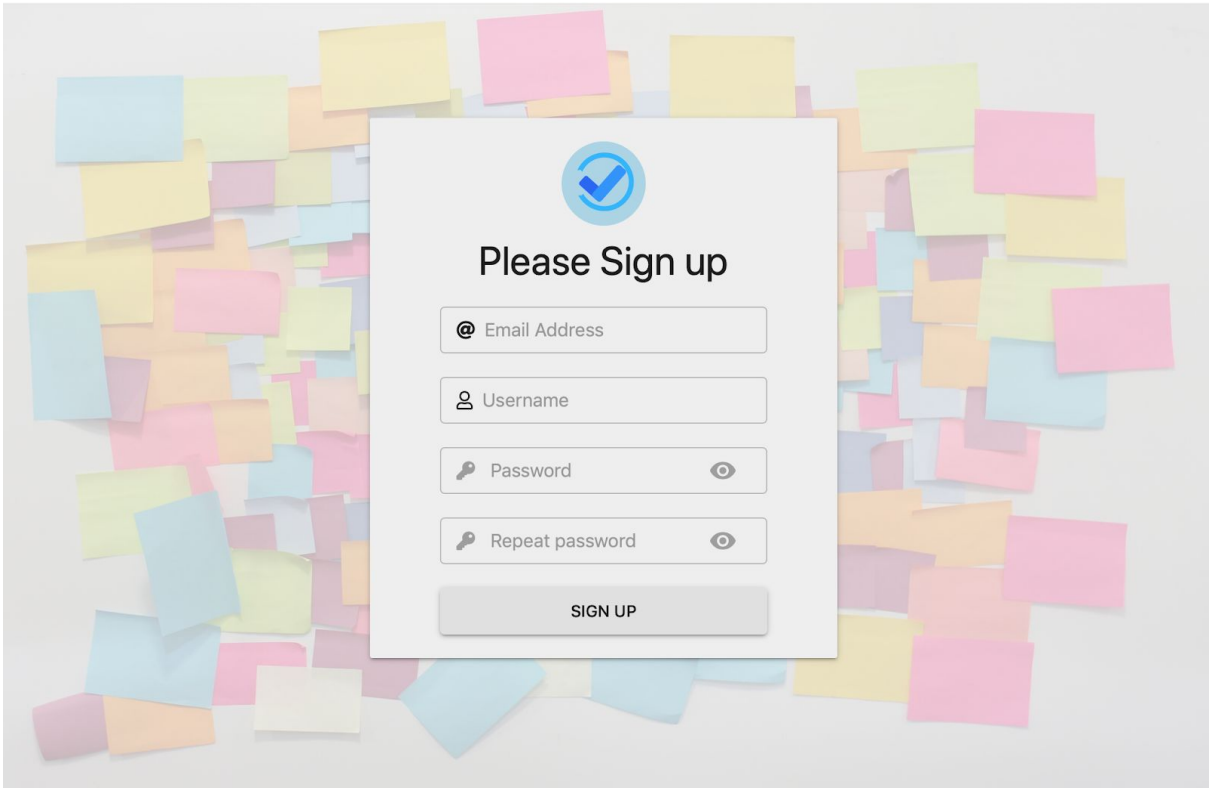
2.1.1. Une tâche contenant des sous-tâches



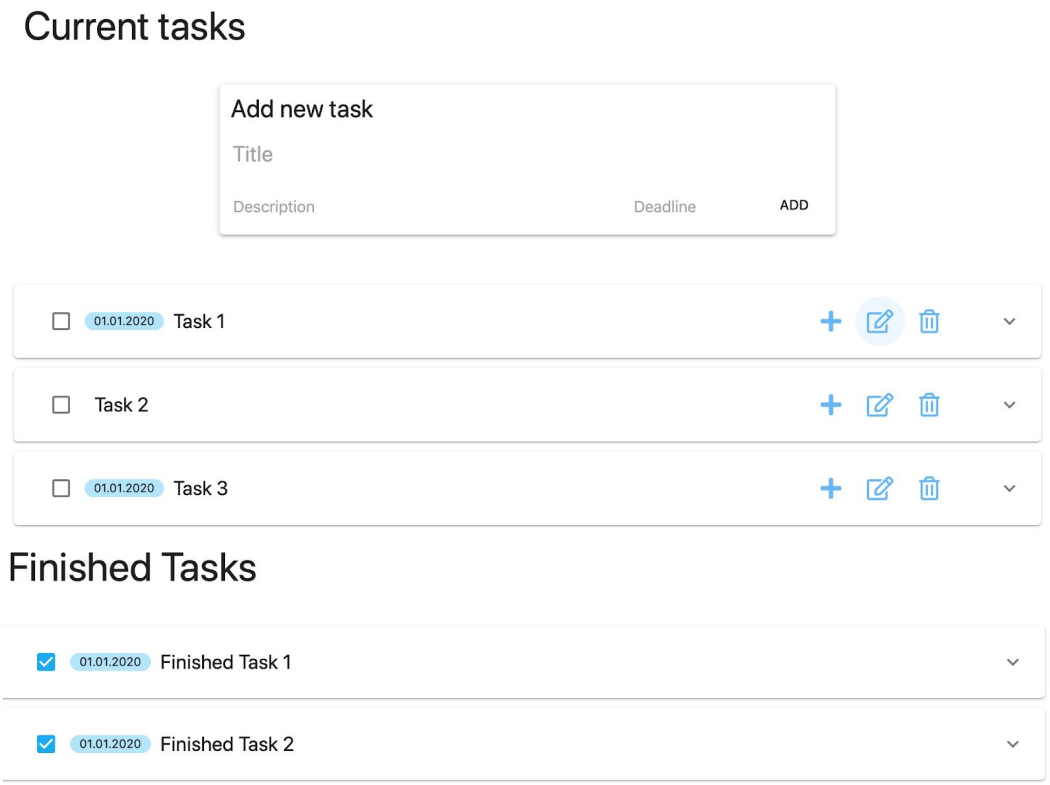
2.1.2. La page d'identification



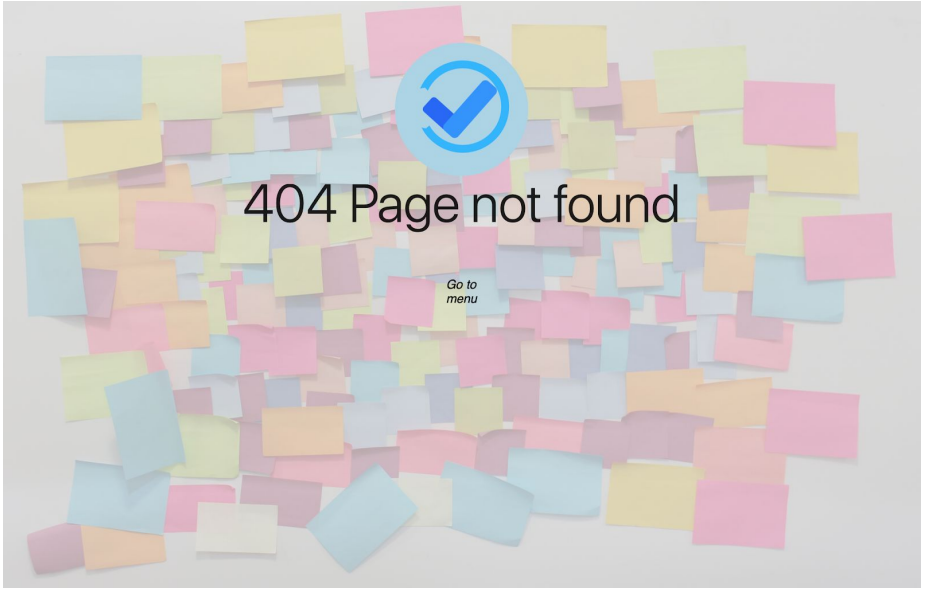
2.1.3. La page d'enregistrement



2.1.4. La page présentant les tâches



2.1.5. La page d'erreur



2.2. Back-end

Notre API REST nous permet de réaliser les tâches suivantes:

Pour une user:

Méthode	Chemin d'accès	Commentaire
GET	/users	Récupération de tous les utilisateurs
PUT	/users/:id	Modification d'un utilisateur donné (:id représente l'id de l'utilisateur)
POST	/users	Création d'un nouvel utilisateur
DELETE	/users/:id	Suppression d'un utilisateur donné (:id représente l'id de l'utilisateur)

Pour une tâche:

Méthode	Chemin d'accès	Commentaire
GET	/tasks	Récupération de toutes les tâches
GET	/users/:id/tasks	Récupération de toutes les tâches d'un utilisateur donné (:id représente l'id de l'utilisateur)
PUT	/tasks/:id	Modification d'une tâche donnée (:id représente l'id de la tâche).
POST	/tasks	Ajout d'une nouvelle tâche
DELETE	/tasks/:id	Suppression d'une tâche donnée (:id représente l'id de la tâche

Toute la partie de gestion des sous-tâches, est fonctionnelle sur l'API REST et sur le react mais nous avons malheureusement manqué de temps pour la liaison de cette partie.

2.3. Base de données

Notre base de données permet de stocker des utilisateurs, des tâches et des sous-tâches.

Voici le schéma que nous avons utilisé:

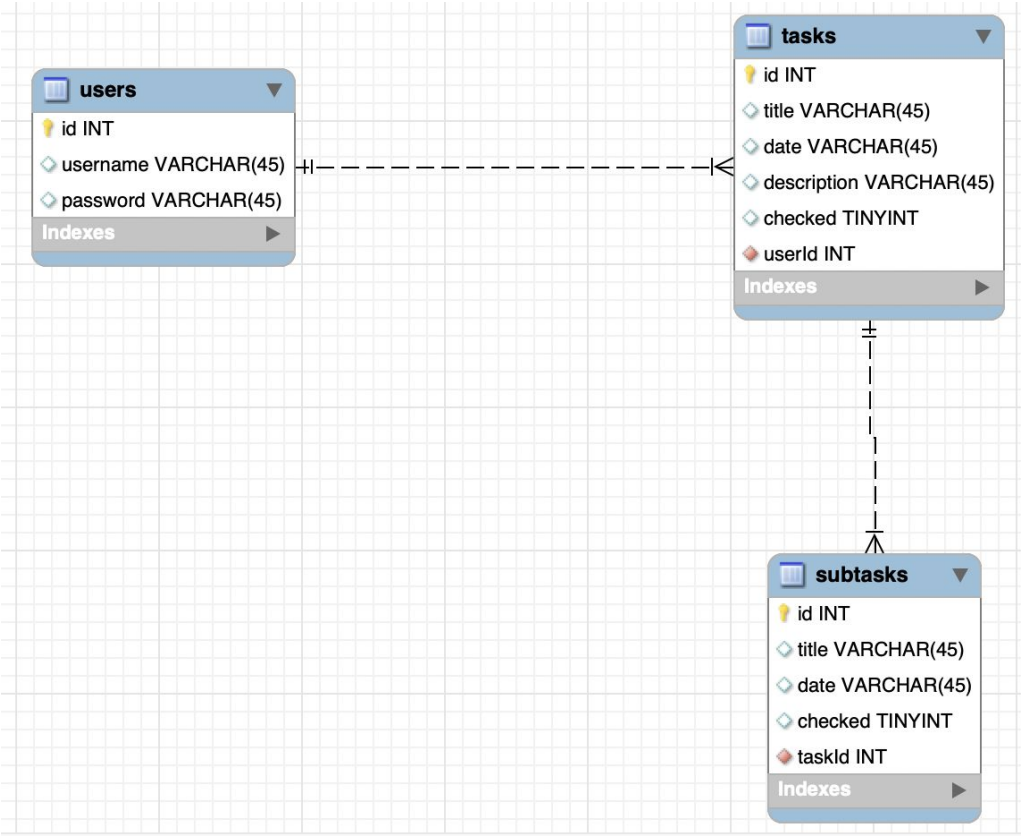


Schéma 1: MLD base de données

Description des tables:

- **users**: contient les informations utilisateur ce qui nous permet de les authentifier.
- **tasks**: contient les informations de chaque tâches liées à chaque utilisateur. Une tâche ne peut être liée qu'à un seul utilisateur mais un utilisateur peut avoir plusieurs tâches.
- **subtask**: contient les informations de chaque sous-tâches liées à des tâches. Une sous-tâche peut être liée à une seule tâche mais une tâche peut avoir plusieurs sous-tâches.

Pour nos tests, nous avons hébergé une base de données MySQL gratuitement sur https://www.freemysqlhosting.net/.

2.4. Autre framework

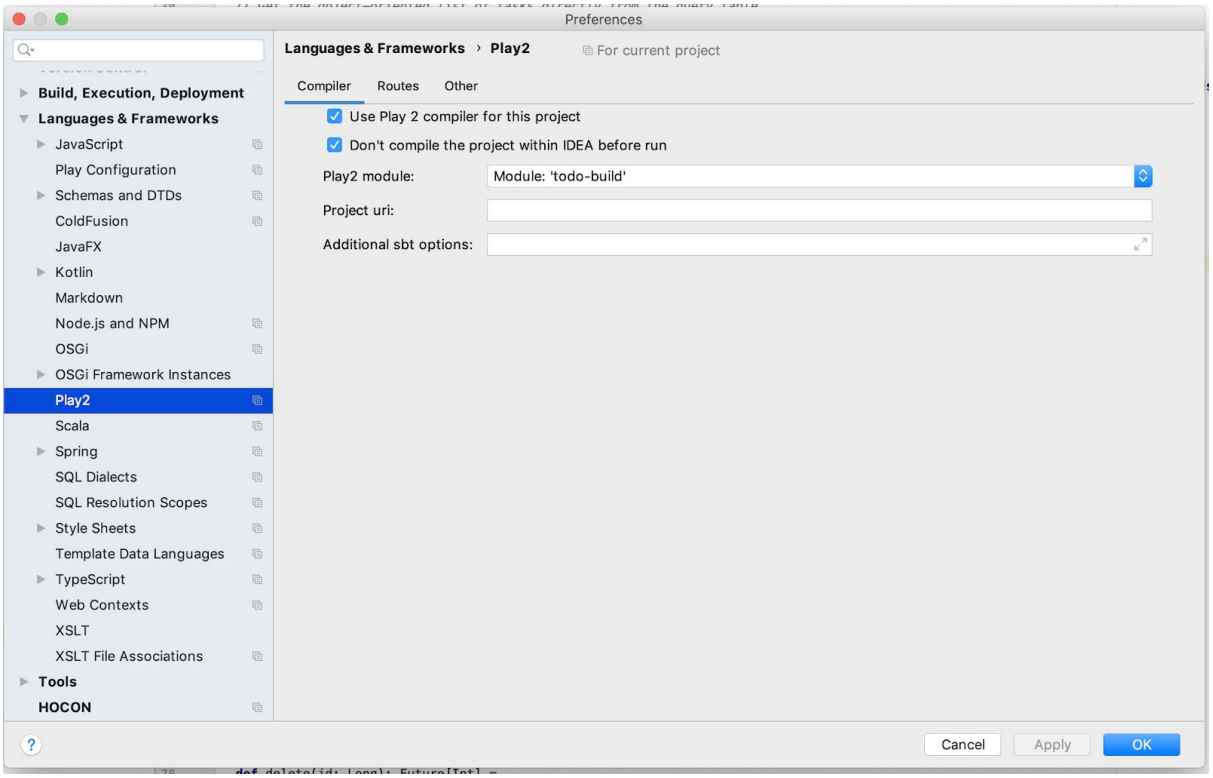
Nous n'avons pas utilisé d'autres frameworks dans le cadre de ce projet.

3. Problèmes

3.1. Backend

3.1.1. Importation du projet démo

Le premier problème que nous avons rencontré a été l'importation du projet démo fourni comme exemple. L'application ne reconnaissait pas les routes, afin de régler de problème, nous avons modifié les préférences dans IntelliJ IDEA puis coché **"Use Play 2 compiler for this project"** :



3.1.2. Foreign keys

Plus tard dans le projet, nous avons eu des problèmes avec les foreign keys qui ont été résolus par les exemples mis en ligne sur Github: https://github.com/c-meier/Teaching-HEIGVD-SCALA-Play-Framework-Examples/tree/fb-play-slick-app.

3.1.3. Cors filter

Nous avons eu des problèmes lors du travail en local lorsque l'on a essayé de connecter le frontend à l'API REST du Cors Filter. Afin de corriger ce problème, nous avons suivi le tutoriel suivant: https://www.playframework.com/documentation/2.7.x/CorsFilter.

3.2. Base de données

Concernant la base de données, nous n'avions pas tous les mêmes versions de MySQL installées en local ce qui pose problème avec les versions du connecteur que nous souhaitons utiliser. Nous avons donc eu plusieurs solutions qui s'offraient à nous, soit une base de données en ligne, soit un container docker. Nous avons décidé d'héberger notre base de données en ligne gratuitement sur https://www.freemysqlhosting.net/.

Tâches non terminées

Les sous-tâches

Les sous-tâches ont été implémentées du côté du front-end. Concernant le back-end, les routes n'ont pas été implémentée et malheureusement, nous n'avons pas eu le temps d'implémenter ces routes et la communication entre le front-end et le back-end.

Identification et enregistrement

L'interface d'identification et d'enregistrement sont créées et fonctionne correctement. Son implémentation du côté du back-end est inexistante.