SafetyLine - Chiffrage

Rapport Final

Zhaojie LU

Chengyu YANG

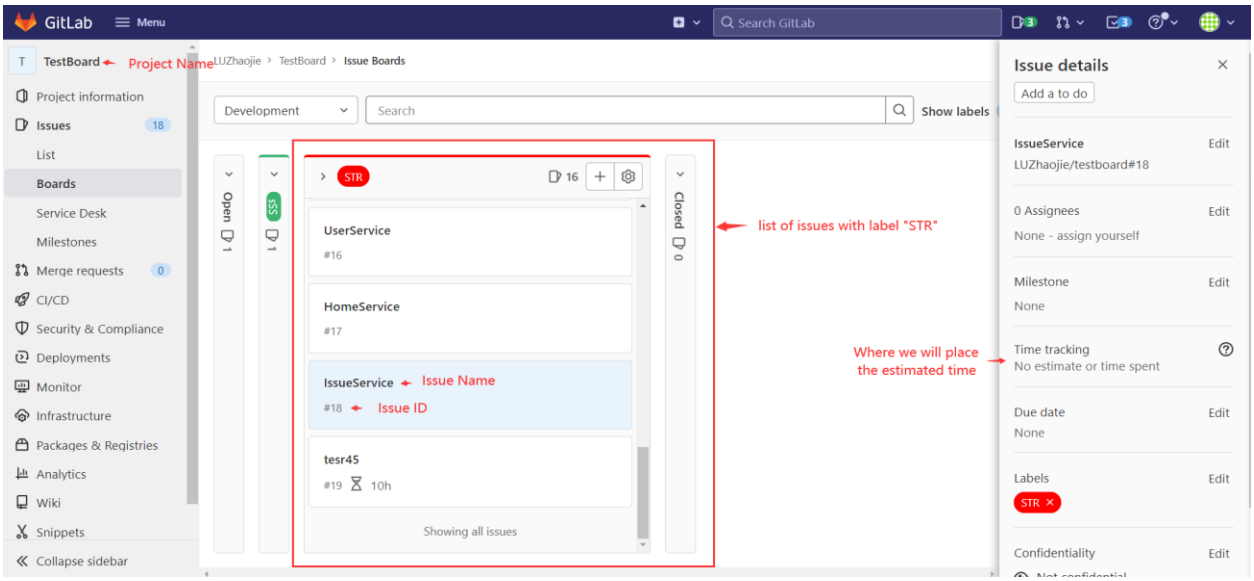
Problème

La cérémonie des “chiffrages” est une partie importante de la gestion de projet d'une entreprise. C’est le moment où les équipes se réunissent pour discuter des tâches à traiter lors du prochain sprint. Une fois les tâches acceptées, assignées par l’équipe de développement, chaque membre va planifier le temps d'achèvement prévu pour sa tâche et envoie l’ensemble de ses estimations par mail au chef de projet. Le chef de projet examine et approuve ces plans. Il va ensuite se connecter au GitLab, et ajouter les temps d'achèvement pour chaque tâche. Tout ce processus est assez lourd, le leader doit gérer les tâches de chaque membre et ne peut être ajouté à Gitlab que manuellement.

A la demande de l'équipe, nous allons construire une page d'application pour automatiser cette cérémonie de chiffrage.

L'analyse du contexte

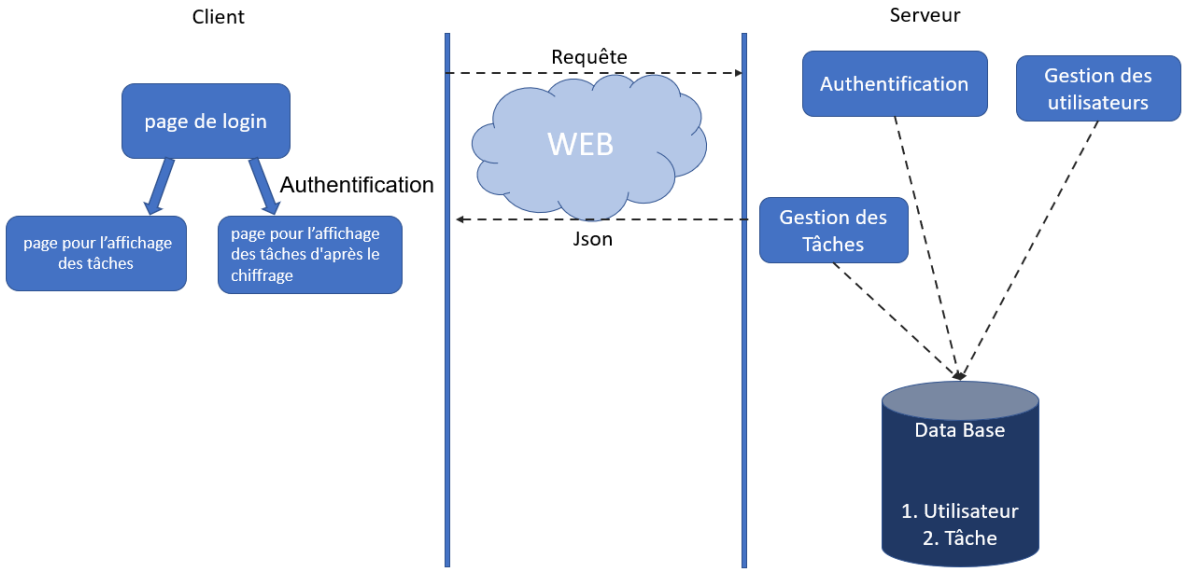
Les tâches dans notre question ci-dessus se trouvent toutes dans le projet Gitlab de l'entreprise. Ce que nous devons faire est d'extraire automatiquement ces tâches non traitées (pas encore ajouter le temps d'achèvement prévu) dans notre application, et de laisser les membres ajouter le temps d'achèvement prévu à cette application, et le chef peuvent voir toutes les tâches de chaque membre et valider ses tâches. Dès qu'il valide une tâche, l'application mettra automatiquement à jour les informations vers Gitlab. De plus, selon les exigences de l'équipe, notre application doit également un module pour la gestion des utilisateurs.



Solution apportée

Afin de créer une application avec les fonctions ci-dessus, nous avons d'abord besoin d'un Framework front-end et back-end.

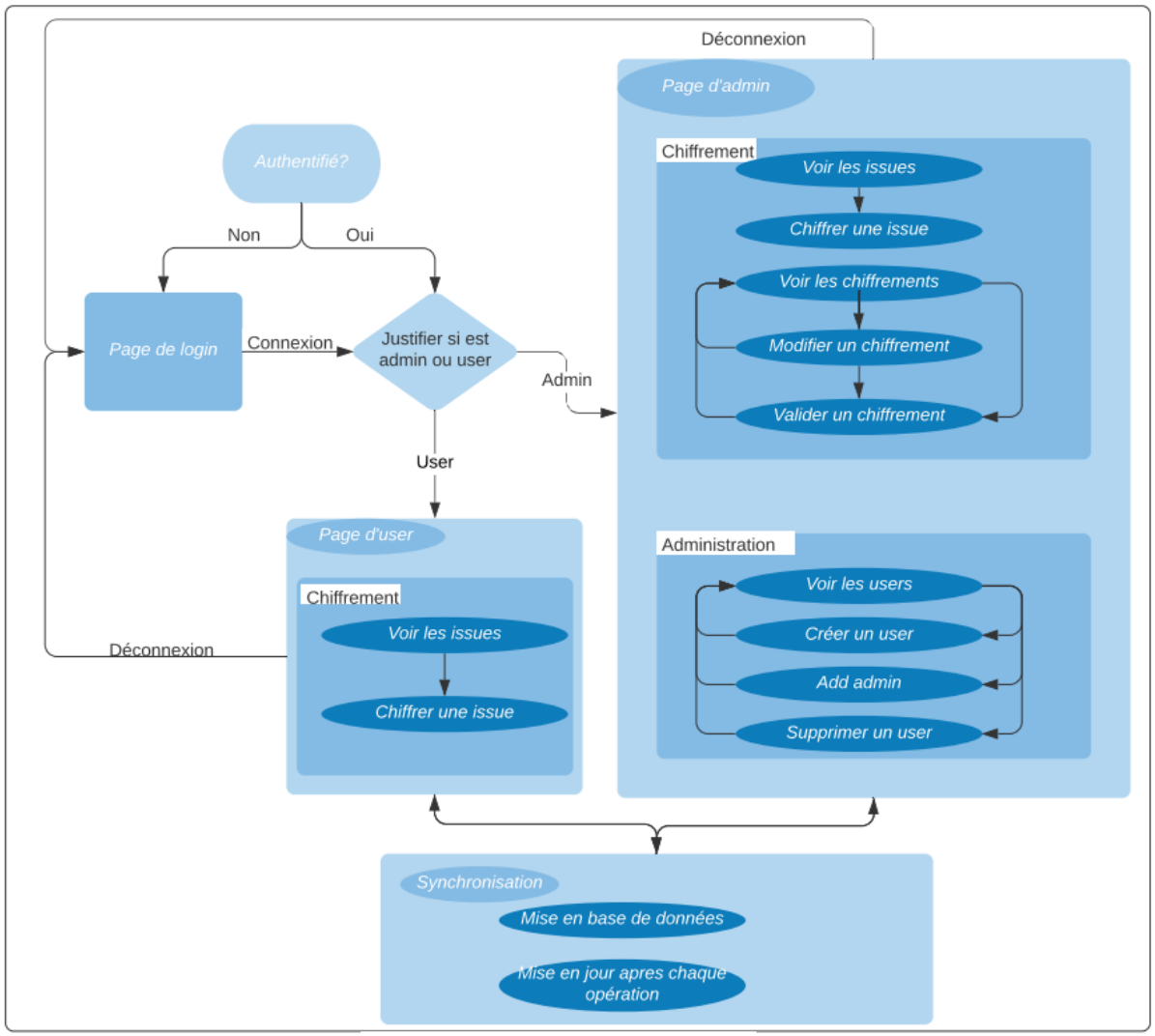
- La couche de données stockera les informations sur l'utilisateur et les informations sur les tâches.
- Le backend fournira les services nécessaires.
- Le frontend mettra en œuvre ces services



Plan détaillé

Dans notre projet, nous avons détaillé ensuite les composants en 3 parties principales.

- Page de Login
  -> distinguer le chef du projet et d'autres membres
- Page User
  -> voir les tâches non traitées et d'ajouter le temps d'achèvement prévu à leur tâche.
- Page Admin
  -> créer et supprimer des user, en plus il peut voir toutes les tâches chiffrées par user et les valider.



Application Safetyline

Réalisation

Couche de données :

Pour que nos données dans l'application soient modifiées par les opérations des différents utilisateurs, nous devons disposer d'une couche de données pour stocker ces données. Nous avons choisi MySQL comme base de données. Pour nos données légères, c'est un très bon choix, il peut récupérer rapidement le contenu dont nous avons besoin et pratique pour installer.

Dans la base de données, nous devons créer deux tableaux. L'une est utilisée pour stocker des données utilisateur, une autre est utilisée pour stocker des informations sur la tâche.

1° Utilisateurs :

Nom de Tableaux	Utilisateurs
Attributs	Username Password Email Rôle (0 = user , 1 = admin)

Exemple dans MySQL : Admin (rôle = 1), User (rôle = 0)

id	username	password	email	role
1	Zhaojie	123456	x1136995140@gmail.com	1
2	Tom	1136995140	tom@gmail.com	1
3	Chef	123456	chef@gmail.com	0

2° Tâches :

Nom de Tableaux	Taches
Attributs	Title Description HumantimeEstimate Updated (1 = validé) Time (Dernière modification) Editor

Exemple dans MySQL :

id	title	humantimeestimate	description	assignee	updated	time	editor
12	MonTest3	2d10h	(NULL)	(NULL)	1	2022-01-14 09:51:48.84	vincent
13	Frontend	2w5d	(NULL)	(NULL)	0	2022-01-14 09:34:23.04	John
14	Backend	2w4d	(NULL)	(NULL)	0	2022-01-20 20:25:57.798	Zhaojie
15	LoginService	(NULL)	(NULL)	(NULL)	0	(NULL)	(NULL)
16	UserService	(NULL)	(NULL)	(NULL)	0	(NULL)	(NULL)

Backend :

Afin de mettre en œuvre les fonctions de notre application, nous avons besoin des services backend qui peuvent faire des opérations avec notre base de données. Ici, Nous avons utilisé la combinaison de SpringBoot et Mybatis, 2 Framework pour construire les fonctionnalités de notre backend.

Pour l'environnement de développement actuel, Spring boot peut être considéré comme le framework JAVA le plus pratique aujourd'hui, c'est une déclinaison du Framework classique de Spring qui permet essentiellement de réaliser des micro services. Il automatise le processus de démarrage complexe du Spring. Il permet donc de créer une API de services très simplement. Ensuite, notre projet doit interagir avec la base de données (MySQL), nous avons donc également besoin d'un moyen pratique pour le réaliser. Nous avons donc choisi le framework MyBatis-Plus. MyBatis-Plus est un puissant outil amélioré de MyBatis. Il fournit de nombreuses opérations efficaces. MyBatis-Plus peut injecter automatiquement des fragments SQL de base, disposer d'un wrapper de condition puissant et flexible, son utilisation peut vous faire gagner beaucoup de temps de développement. Il peut être facilement ajouté à Spring boot via Maven.

Nous utilisons également une bibliothèque qui s'appelle GitLab4J qui est utilisé pour nous permettre spécifiquement de faire fonctionner des éléments de Gitlab avec du code java.

GitLab4J™ API (gitlab4j-api)
Java Client Library for the GitLab REST API

GitLab4J™ API (gitlab4j-api) provides a full featured and easy to consume Java library for working with GitLab repositories via the GitLab REST API. Additionally, full support for working with GitLab webhooks and system hooks is also provided.

Service réalisé :

Nom de Service	Définition
Enregistrer/Créer User	Nous pouvons enregistrer un user dans notre base de données en donnant les informations nécessaires (name, email…)
Supprimer User	Supprimer un user dans notre base de données
Recherche User par Nom/ID	Récupérer les informations de l'utilisateur en saisissant le nom de l'utilisateur/ ID
Récupérer rôle	Récupérer le rôle d'un compte utilisateur en saisissant le nom de l'utilisateur
Login	L'exactitude du nom d'utilisateur et du mot de passe peut être vérifiée par rapport à la base de données
Changer Password	Changer le mot de passe de l'utilisateur
Mettre à jour	Obtenir des tâches avec une balise spécifique à partir d'un projet gitlab
Valider Tâche	Téléchargez les tâches qui ont été ajoutées avec le temps estimé aux tâches correspondantes sur gitlab
Ajouter Temps Estimé	Ajouter/modifier du temps estimé aux tâches
Taches Triés	Retourner les taches triés par heure de dernière modification

Frontend :

Selon les besoins de l'équipe, nous utiliserons le Framework Angular pour construire la page frontend qui est une plateforme pour la création d'une web application au niveau de l'entreprise. Il s'agit d'un cadre frontal relativement complet, comprenant des services, des modèles, une modularisation, un routage etc.

En utilisant ce Framework, nous avons construit avec succès notre application selon le plan et les services que nous avons créés précédemment.

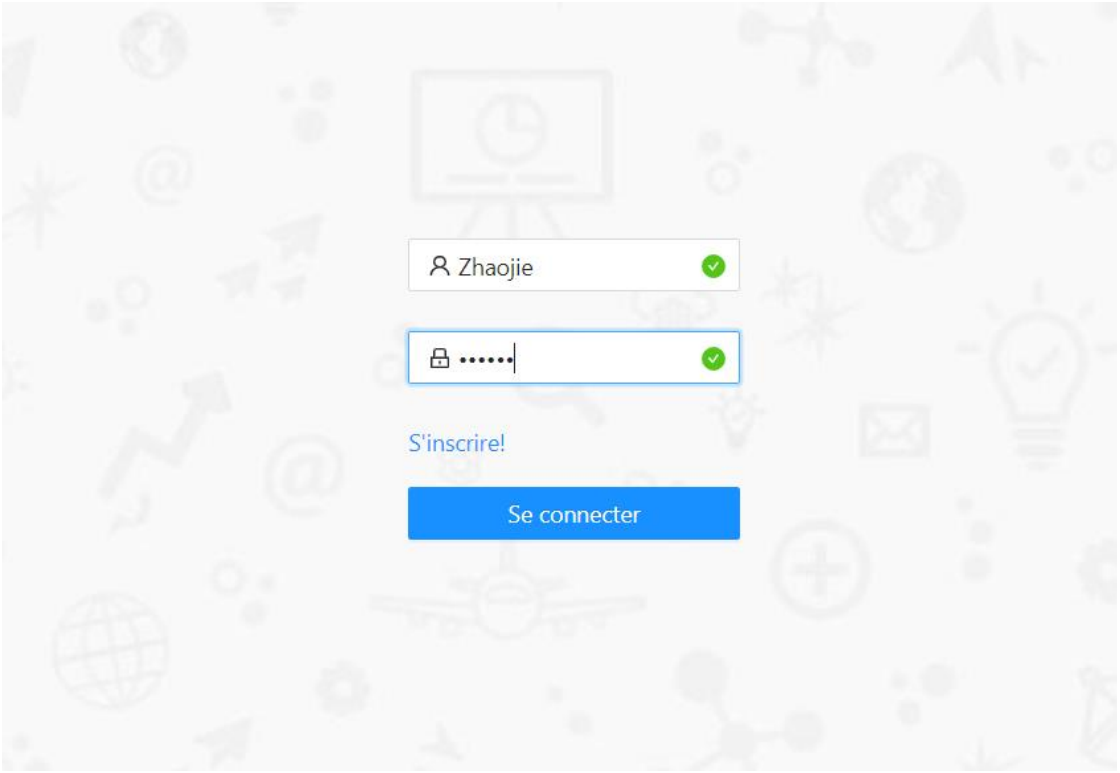


Figure 1: Page de Login

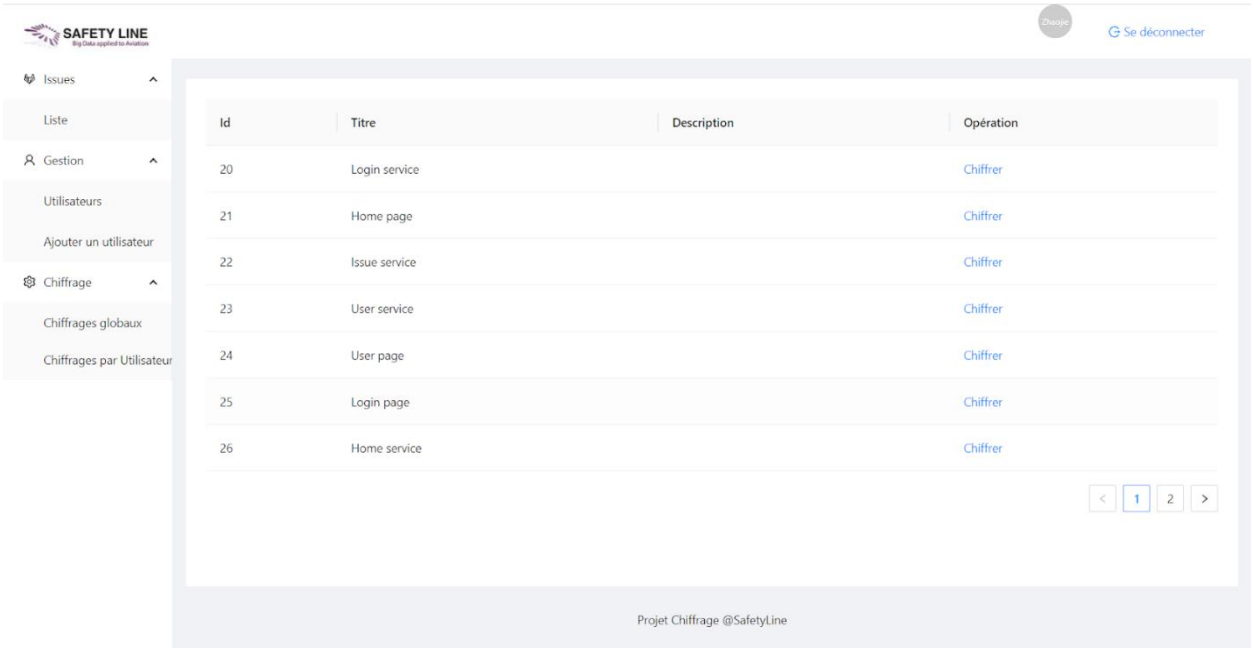


Figure 2 : Page d'administrateur

A l'aide de <localStorage>, nous pouvons enregistrer les informations dans le navigateur. Dans notre application, nous avons <user-token> pour vérifier l'identité, <role-token> pour déterminer son niveau d'accès, <id-token> pour exécuter les opérations sur les utilisateurs, <username-token> pour afficher le nom d'utilisateur à la page d'accueil.

Dans le module de connexion, nous avons utilisé le garde de la route. Avec son aide, seul l'utilisateur qui a déjà s'inscrit peut accéder à la page d'accueil parce qu'il va vérifier les données dans <localStorage>. Si les données sont valides, l'utilisateur peut accéder à toutes les pages auxquelles il a accès. Sinon, il va toujours rester à la page de connexion. Une fois que l'utilisateur ferme la page, toutes les données seront effacées et il doit se reconnecter.

Et pour éviter l'erreur d'authentification, nous ajoutons un module d'intercepteur qui va vérifier l'identité d'utilisateur à chaque opération. S'il y a une erreur, il va supprimer les

données dans <localStorage> et se navigate à la page de connexion. L'utilisateur doit se reconnecter pour continuer leur opération.

En détectant les données dans <localStorage>, nous pouvons déterminer s'il y a un utilisateur connecté. Notre application ne permet pas de se connecter à deux comptes en même temps. Si le deuxième utilisateur veut se connecter dans le même navigateur en même temps, il faut strictement déconnecter le premier compte, sinon le deuxième utilisateur ne peut pas se connecter.

Pour simplifier l'utilisation de notre application, si l'utilisateur oublie de se déconnecter avant de quitter la page, notre application va automatiquement effacer ses données dans le navigateur.

Conclusion

Après les tests en utilisant Postman (outil pour test des APIs) et les utilisations réelles sur notre application, nous somme prête à l'emploi. Les modules de gestion des utilisateurs et des tâches fonctionnent bien d'après avoir exécuté le front-end et le back-end. L'application peut automatiquement mettre à jour les tâches qui doivent être chargées en fonction du contenu du projet Gitlab. Et les membres peuvent également ajouter du temps estimé à ces tâches conformément aux principes de Gitlab.

Le seul problème pourrait être une mauvaise portabilité. Le projet et les étiquettes de tâche du Gitlab auxquelles notre application accède sont fixes. Si nous voulons changer le projet, nous devons changer le code backend.