Université de Lille

PROJET WEB
THE DA2I TRICOUNT

WEB-BDD
LP DA2I
L'HÔTE BASIL
PAREE HUGO
2022 - 2023

I/ Déploiement

i)Clonage git

Ouvrez un terminal et naviguez jusqu'au répertoire où vous souhaitez cloner le projet. Utilisez la commande suivante pour cloner le projet à partir de Git :

git clone git@gitlab-ssh.univ-lille.fr:basil.lhote.etu/RembourseMoi.git
git clone https://gitlab.univ-lille.fr/basil.lhote.etu/RembourseMoi.git

ii)Dépendances

Assurez-vous que vous disposez de Java et Maven installés sur votre système. Si vous ne les avez pas installés, vous pouvez les télécharger depuis les sites web respectifs.

iii)Exécution

Dans le répertoire du projet, exécutez la commande suivante pour faire tourner le serveur localement.

```
mvn spring-boot:run
```

iv)Accès à l'application

Maintenant, accédez à l'application en ouvrant un navigateur web et en saisissant l'URL suivante : https://localhost:4383

II/Synthèse

i) Objectif de travail

Une entreprise souhaite créer une application de gestion de comptes entre amis similaire à Tricount, Pumpkin, Splitwise, Flooz, Splid ou Lydia. L'application permet aux utilisateurs enregistrés de créer un événement et d'ajouter des entrées de dépenses pour enregistrer qui a payé quoi et pour qui. Les autres participants sont identifiés par un pseudo. L'application permet de calculer les dépenses cumulées et le nombre minimal de transactions nécessaires pour équilibrer les comptes entre tous les participants.

ii) Ce qui a été fait

Les demandes du client et leur statut sont listées ci-dessous :

- ☑ Conception d'objets "Métier" réutilisables hors web.
- ☑ Spring ou au minimum organisation MVC "à la main"
- ☑ JPA ou au minimum organisation DAO "à la main"
- ☑ Feuilles de styles externalisées (CSS, Bootstrap, Pure, ...)
- ☑ Upload ... charger sa photo sur son profil par exemple.
- ☑ "responsive design" : Fonctionnement sur téléphone portable aussi bien que sur station de travail
- ☑ Entêtes mises en commun (fichiers includes, les logos DA2I et Lille1 doivent figurer sur toutes les pages ...)
- ☑ Système de trace d'activité du serveur (Logger, Valve...)
- ☑ Ergonomie générale du site,
- ☐ Code coté client vérifiant autant que possible les informations saisies (JavaScipt)
- ☐ Validation HTML des pages par Tidy ou par le W3C. Présence de l'icône de test sur chaque page.
- ☑ Eviter l'injection SQL
- ☑ Système d'authentification automatique(Spring Security , Realm, ...)
- ☐ Accès à la base via un pool de connexions paramétrable
- ☑ Cryptage des informations via un serveur sécurisé (SSL,...)
- ☑ Centralisation de tout ce qui pourrait changer (logo, nom de la formation, base de données etc .. web.xml ou application.properties)

Nous avons aussi choisis de rajouter une fonctionnalité permettant de rendre le site accessible en plusieurs langues pour permettre des événements internationaux.

iii) Les améliorations possibles

a)Ajout d'un système de mail

Il pourrait être intéressant de rajouter un système de mail pour être notifié d'une dépense nous concernant, des dates des événements ou bien pour récupérer son mot de passe. (Ce système a été ajouté et testé après la date de rendu, il est fonctionnel mais non disponible)

iv) En quoi le projet était difficile

La difficulté principale du projet résidait dans l'algorithme de remboursement, c'est un algorithme difficile à prendre en main qui doit recalculer à chaque dépense combien chaque personne se doit afin d'optimiser au maximum les transactions de remboursement.

III/ Conclusion

i) Description technique du projet

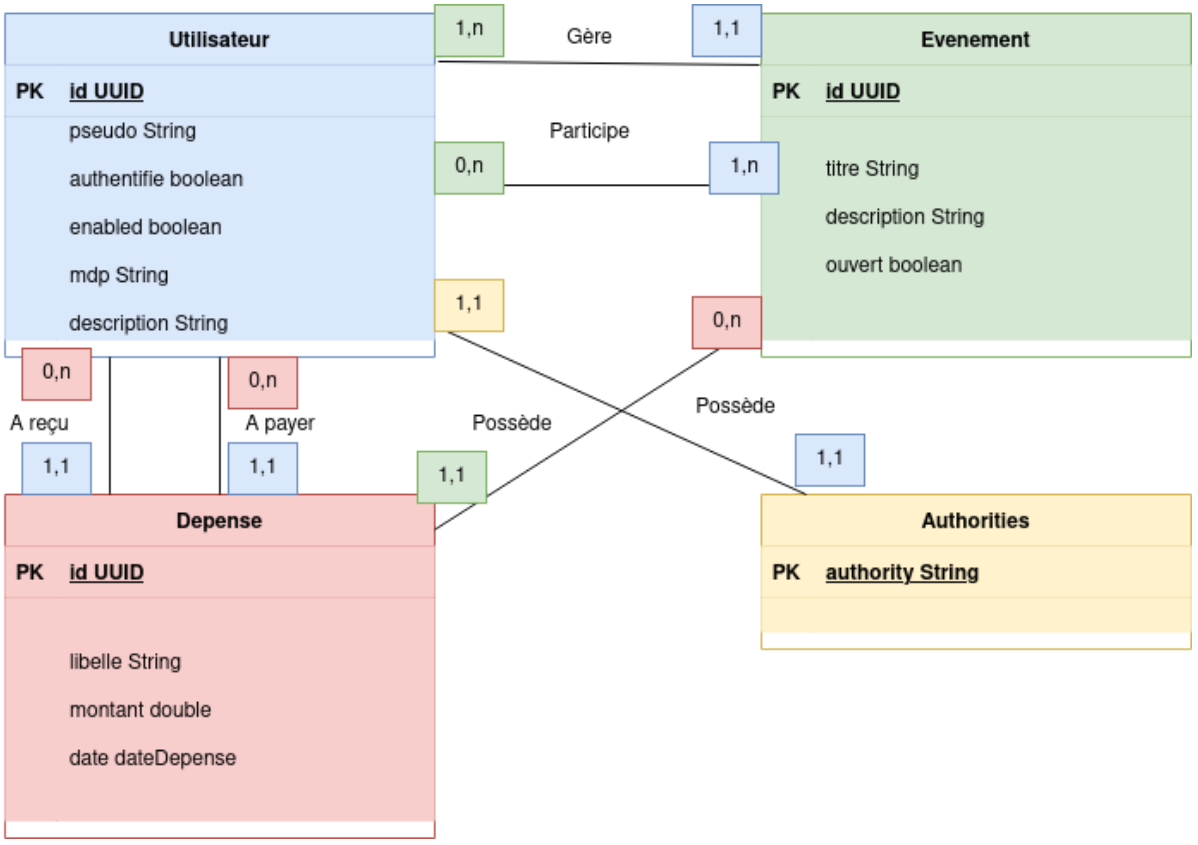
a)Structure du projet

Le projet à été structuré en suivant le principe de Spring MVC

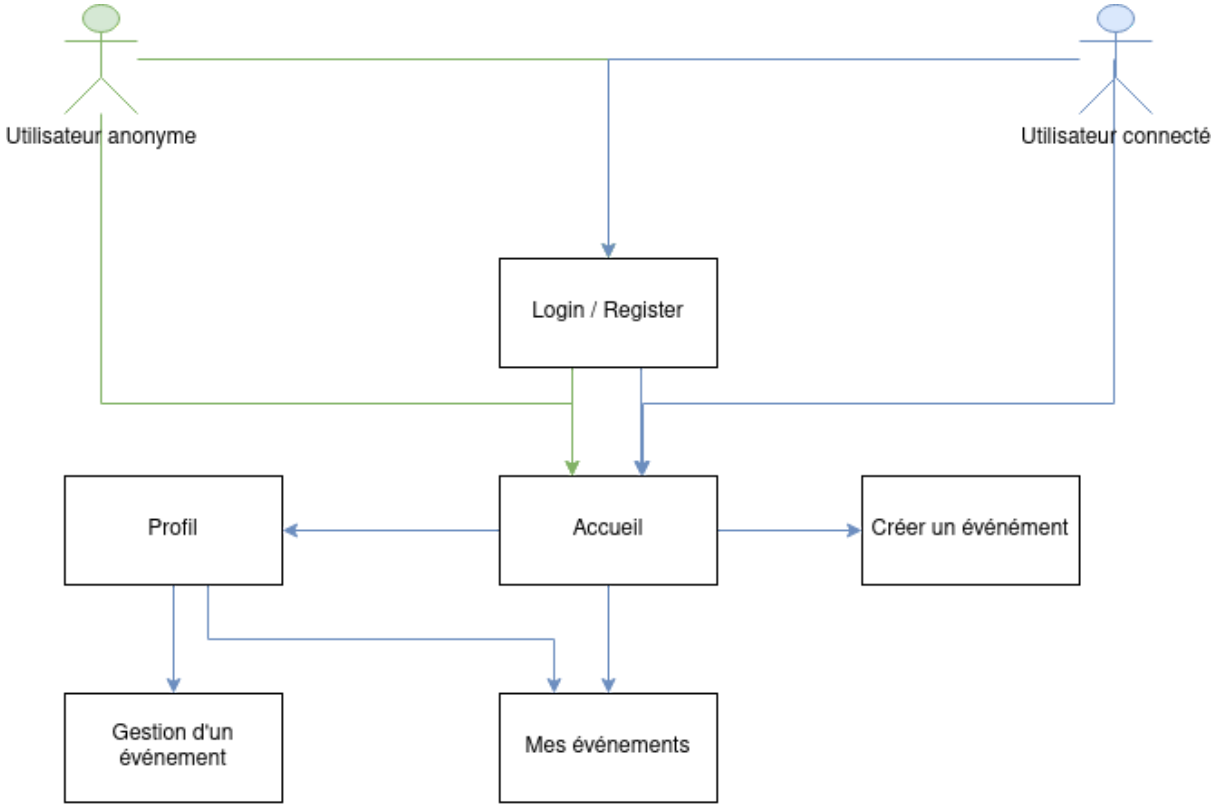
```
.
└── src
    ├── main
    │   ├── java
    │   │   └── fr
    │   │       └── da2i
    │   │           ├── config
    │   │           │   ├── ErrorConfig.java
    │   │           │   ├── JspConfiguration.java
    │   │           │   ├── LangConfig.java
    │   │           │   ├── Properties.java
    │   │           │   ├── UserSecurityService.java
    │   │           │   └── WebSecurityConfig.java
    │   │           ├── controllers
    │   │           │   ├── DepenseController.java
    │   │           │   ├── EvenementController.java
    │   │           │   ├── ExceptionController.java
    │   │           │   ├── IndexController.java
    │   │           │   ├── ProfilController.java
    │   │           │   └── UtilisateurController.java
    │   │           ├── entities
    │   │           │   ├── AuthoritiesID.java
    │   │           │   ├── Authorities.java
    │   │           │   ├── Depense.java
    │   │           │   ├── Evenement.java
    │   │           │   └── Utilisateur.java
    │   │           ├── RembourseMoiApplication.java
    │   │           ├── repositories
    │   │           │   ├── AuthoritiesRepository.java
    │   │           │   ├── DepenseRepository.java
    │   │           │   ├── EvenementRepository.java
    │   │           │   └── UtilisateurRepository.java
    │   │           └── service
    │   │               └── LoggerService.java
    │   ├── resources
    │   │   ├── application.properties
    │   │   ├── import.sql
```

```
│   │   ├── mescertifs.p12
│   │   ├── messages_de.properties
│   │   ├── messages_en.properties
│   │   ├── messages_fr.properties
│   │   ├── messages_fu.properties
│   │   ├── messages_it.properties
│   │   ├── messages.properties
│   │   └── messsages_es.properties
│   └── webapp
│       ├── resources
│       │   ├── css
│       │   │   └── styles.css
│       │   └── images
│       │       ├── error.gif
│       │       ├── lille.png
│       │       ├── logo-da2i.png
│       │       └── logo.png
│       └── WEB-INF
│       └── jsp
│           ├── creerEvenement.jsp
│           ├── depense.jsp
│           ├── error.jsp
│           ├── evenement.jsp
│           ├── formAddUser
│           │   ├── formAnonyme.jsp
│           │   └── formAuth.jsp
│           ├── global
│           │   ├── footer.jsp
│           │   └── navbar.jsp
│           ├── index.jsp
│           ├── login.jsp
│           ├── mesEvenements.jsp
│           ├── profil.jsp
│           ├── register.jsp
│           ├── vueDepense.jsp
│           ├── vueDettes.jsp
│           └── vueEvenement.jsp
└── test
└── java
└── fr
    └── da2i
    └── RembourseMoiApplicationTests.java
```

b)Structure de la base de donnée



c)Structure du site



ii) Objectif technique du travail

L'aspect technique du projet réside dans la conception de l'algorithme de remboursement. Cet algorithme permet de calculer les soldes et dettes entre les participants d'un événement.

Il prend en entrée un objet Événement, qui contient une liste de Dépense et une liste d' utilisateur. Il utilise ces informations pour calculer les soldes de chaque utilisateur, c'est-à-dire le montant qu'il doit ou qu'il est dû par rapport aux dépenses.

Tout d'abord, il initialise une map des soldes pour chaque participant à 0. Ensuite, il parcourt chaque dépense et met à jour les soldes en fonction des utilisateurs qui ont payé et qui ont été payés. Pour chaque utilisateur payé, il diminue son solde du montant total de la dépense divisé par le nombre de payeurs. Pour l'utilisateur payeur, il augmente son solde du montant total de la dépense.

Une fois que tous les soldes ont été calculés, l'algorithme cherche à déterminer les dettes et crédits entre les participants. Il parcourt les soldes et crée une liste de dettes pour chaque utilisateur qui a un solde positif. Pour chaque dette, il cherche un Utilisateur avec un solde négatif qui doit au moins autant que la dette (afin de diminuer les transactions), et il met à jour les soldes en conséquence. Enfin, il renvoie une liste de String qui contient les dettes et crédits de chaque participant sous la forme "Utilisateur A doit X à Utilisateur B".

iii) Principe de réalisation

a)Hébergement

Le projet est sur le gitlab de l'université de Lille afin d'avoir un système de versioning et aussi de pouvoir travailler dessus sur différents ordinateurs.

b)Les branches

De nombreuses branches ont été créées lors d'ajouts de features afin d'éviter les conflits et de permettre à chacun de travailler sur sa partie.

c)La répartition

Au début Hugo assurait une grande partie du back tandis que Basil s'occupait du front, sur la fin du projet, les différentes features restantes en back ont été distribuées entre les deux collaborateurs.

d)Technologies utilisées

Le projet tourne sous Spring Boot 3 à l'aide de Maven et utilise en librairie externe Boostrap 5.

iv) Difficultés techniques rencontrées et solutions apportés

a)JSTL

Le projet étant réalisé en Spring Boot 3, nous avons rencontrés un soucis avec l'utilisation des JSTL, en effet, elles ne possèdent plus le même URL d'appel, il faut utiliser :

```
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
```

b)Algorithme

L'algorithme est passé par plusieurs versions de test et nous a posé quelques problèmes, au début nous n'étions capables d'équilibrer que si tout le monde dépensait pour tout le monde. En nous renseignant sur internet et en faisant beaucoup d'essais, nous avons finalement abouti à la version finale de l'algorithme.

IV/ Documentation utilisateur

i) Objectif de l'application

L'objectif de l'application est de fournir à l'utilisateur un système de gestion d'événement et de dépenses lié à ces derniers afin de permettre une répartition équitable des charges lors de l'organisation et du déroulement d'un événement.

ii) Du point de vue de l'utilisateur

Vous avez la possibilité de vous créer un compte afin de créer des événements, en rejoindre, en gérer, effectuer des dépenses.. Cependant vous pouvez aussi être ajouté en tant que participant anonyme (sans compte) par le biais d'un autre, dans ce cas c'est cette personne qui déclare vos dépenses et ce que vous devez.