Documentation technique

Table des matières

1. Présentation générale

1.1 Introduction du projet

Mercifood est une application web qui permet de partager de la nourriture autour de soi. Un utilisateur souhaitant cuisiner aura la possibilité de mettre à la disposition d'autres utilisateurs, proches de chez lui, les informations relatives à son plat.
Ces informations contiennent les aliments qui composent son plat à partager, le nombre de personnes maximum, il aura aussi la possibilité d'ajouter des photos.
Les utilisateurs intéressés pourront ensuite réserver le plat qui leur convient, et auront la possibilité de laisser une évaluation sur leur expérience.

1.2 Présentation des fonctionnalités

1.2.1 Authentification

Afin de proposer le partage ou d'avoir la possibilité de réserver un plat, il est nécessaire d'avoir un compte sur Mercifood et de se connecter.
Une adresse mail ou un numéro de téléphone est donc obligatoire afin de pouvoir obtenir des informations sur la réservation en cours au cas où l'utilisateur se déconnecte avant de récupérer son plat.
Chaque compte sera accompagné d'un profil, dans lequel les exigences de chaque utilisateur sera indiqué, notamment pour pouvoir réaliser des filtres sur leurs recherches de plats.

1.2.2 Partager ou récupérer un plat

En tant qu'utilisateur connecté, je souhaite avoir accès à ces deux fonctionnalités. Sur l'écran d'accueil j'aurais la possibilité de choisir quelle action je souhaite réaliser :

- Si je choisis de partager un plat, j'aurai accès à un formulaire à remplir avec les informations de celui-ci. Une fois que je suis prêt à partager, j'aurais accès à une carte laissant apparaître la localisation des utilisateurs autour de moi. Il est possible que le nombre de personnes intéressées soit supérieur au nombre de personnes maximum imposé. Les utilisateurs qui bénéficieront de ce partage seront les premiers à avoir réservé.
- Si je choisis de rechercher un plat, j'aurais accès à une carte laissant apparaître les utilisateurs ayant mis en ligne leur plat. Les données affichées seront cliquables afin d'avoir accès aux informations données par l'utilisateur qui partage et de vérifier la compatibilité entre le plat et notre profil (allergies, régime alimentaire, etc).

1.2.3 Communication à l'aide d'un chat

En tant qu'utilisateur connecté, je souhaite pouvoir échanger avec d'autres utilisateurs par un chat afin de faciliter le processus de réservation. Lorsqu'un utilisateur a partagé un plat et que je le réserve, je pourrai si besoin lui poser des questions ou répondre aux siennes.

Ce chat permet surtout à l'utilisateur qui partage d'avoir toutes les conversations entre tous les utilisateurs ayant réservé réunies dans une seule application, ce qui peut aussi être utile pour envoyer des messages génériques qui les concernent tous.

1.2.4 Laisser une évaluation

En tant qu'utilisateur connecté et après avoir effectué une réservation, j'aurai la possibilité de laisser une évaluation une fois mon plat récupéré. Cette évaluation est composée d'une note sur 5 ainsi que d'un commentaire (temps d'attente, communication avec l'utilisateur qui partage, qualité/conformité du plat, etc). Cette évaluation sera visible sur le profil de l'utilisateur qui a fourni le plat.

Gestion de projet

2.1 Présentation de l'outil mis en place

Afin de gérer l'équipe de projet, un outil réalisé en Java a été mis en place en parallèle de l'application. Cet outil nous permettra de gérer l'ensemble des tâches définies pour ce projet, soit en ligne de commande, soit via une interface.

Le code est divisé en plusieurs packages : database, models et view:

- Le package models contient les différentes classes du projet
- Le package view contient les fichiers fxml utilisés pour l'interface graphique ainsi que leurs contrôleurs

2.2 Explications des différentes classes

Classe Home:

Cette classe représente l'accueil, elle permet de regrouper l'ensemble des projets créés ainsi que tous les membres des projets.
L'accueil est composé de :

- un nom
- un liste de projets
- une liste de membres

Classe Project:

Cette classe permet de réunir les informations relatives aux membres ainsi qu'aux tâches d'un projet.
Un Projet est donc composé de :

- un nom
- une liste de membres
- une liste de tâches
- un prochain point de rendez-vous
- une deadline

- un accueil

Classe Task:

Cette classe regroupe tout ce qui concerne une tâche d'un projet.
Une tâche est composée de:

- un nom
- un propriétaire
- une deadline
- une date de création
- un état (cf. Énumération Sate)
- un commentaire
- un projet

Classe Member:

Cette classe regroupe tout ce qui concerne un membre d'un projet
Un membre est composé de:

- un nom
- une liste des tâches qui lui ont été assignées

Enumeration State:

Cette énumération permet de lister les différents états qu'une tâche peut avoir :

- To do : l'état par défaut attribué à une tâche lors de sa création
- In progress
- Done

Classe Database:

Cette classe sert de lien entre le projet et la base de données MySQL. Au lancement du programme, elle permet de faire la connexion à la base de données et de récupérer les valeurs existantes. Elle contient également les fonctions nécessaires à l'écriture dans la base.

2.3 Exemple d'utilisation

Afin de commencer, il est nécessaire de réaliser les étapes suivantes :

1) Se placer dans le répertoire backend/java/planning et ouvrir un terminal

2) Il faut mettre en place la base de données. Deux options sont possibles :

   Si votre docker est installé sur votre machine, la base de données peut être lancée à l'aide du docker-compose disponible dans le projet.

Il suffira de lancer la commande `docker-compose up`, et si besoin, d'ouvrir phpMyAdmin dans le navigateur avec l'adresse suivante : localhost:8081. (mêmes identifiants)

Dans le cas échéant, créez une base de données mySQL avec:

- nom de la base de données : home_projects
- utilisateur : root
- mot de passe root
- port : 3306

3) Pour lancer le projet, ouvrir un autre terminal et exécuter la commande suivante:
`java -jar planning.jar` afin de lancer le programme en ligne de commande, ou alors
`java -jar planning_interface.jar` afin de le lancer avec l'interface graphique

Dans la version en lignes de commandes, la première question posée à l'utilisateur est de choisir si il veut créer un nouveau projet où gérer un projet existant :

```
What do you want to do ?
1. Create a new project
2. Update an existing project
3. Nothing
```

Menu principal

Si l'utilisateur choisit 1, un nouveau projet sera créé et aura accès au menu d'édition d'un projet.
S'il choisit 2 il choisira un projet à modifier et aura ensuite accès au menu d'édition.
S'il choisit 3, il aura un récapitulatif de l'ensemble des projets existants.

```
What do you want to do in this project ?
1. Change the name
2. Add a member
3. Remove a member
4. Add a task
5. Update an existing task
6. Update next appointment
7. Update the deadline
8. Remove from home
9. Nothing
```

Menu d'édition d'un projet

Ce fonctionnement sera le même pour les autres actions. Si l'utilisateur souhaite ajouter une tâche à ce projet, après l'avoir créée il aura accès au menu d'édition pour cette tâche, de même pour modifier une tâche existante.

```
What do you want to do with this task ?
1. Update the state
2. Update the owner
3. Update the deadline
4. Write a comment
5. Remove from project
6. Nothing
```

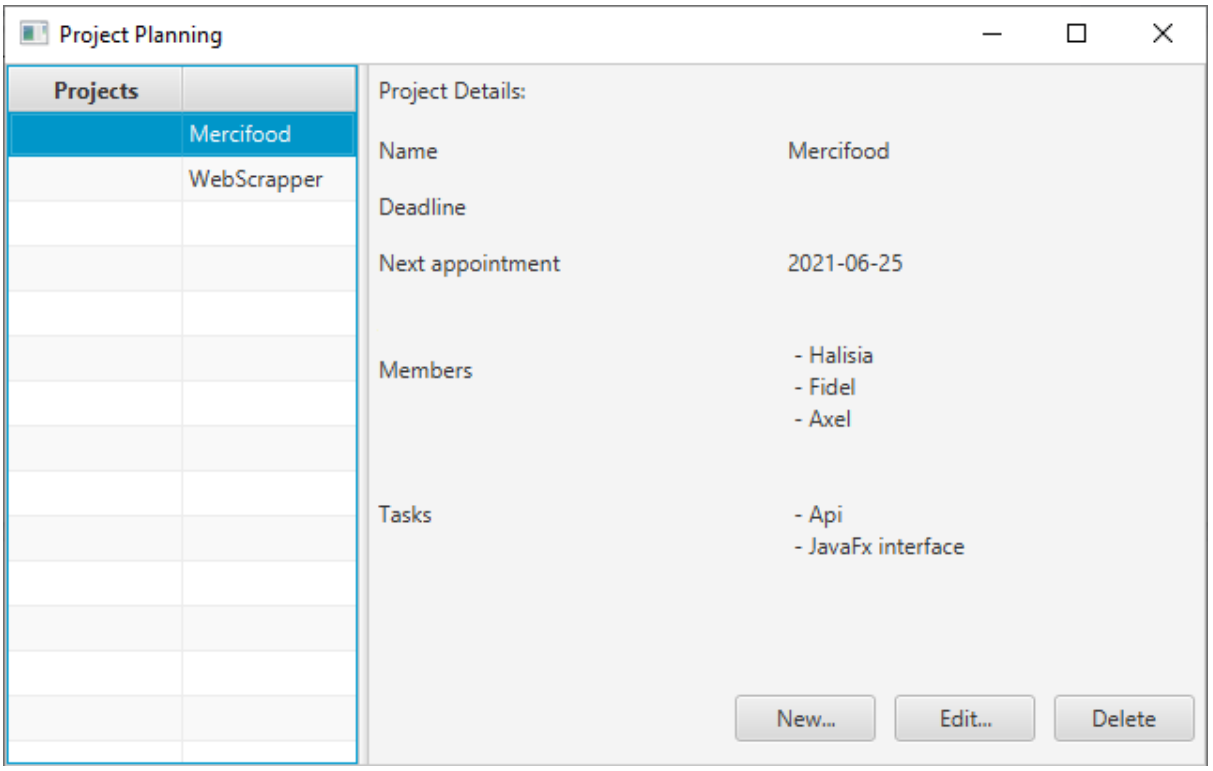
Menu d'édition d'une tâche

Lorsque l'utilisateur termine une action, il appuie sur l'option "Nothing" qui sert de retour. Après avoir édité une tâche, il sera redirigé vers le menu d'édition du projet en cours, et s'il appuie encore sur "Nothing", il obtiendra le récapitulatif des projets existants.

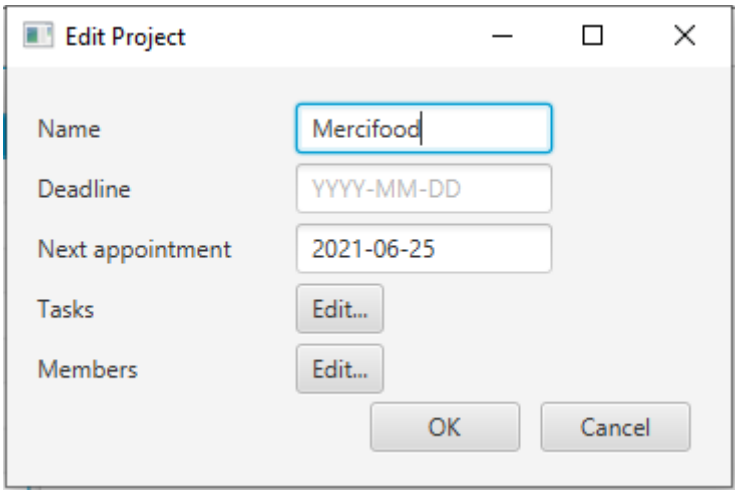
Exemple de récapitulatif obtenu:

```
[PROJECT Mercifood
--------------------------
Members :  - Halisia - Fidel - Axel
Next Appointement : 2021-06-25
Deadline : null
Tasks : [
    Api
    ---
    Owner : null
    State : DONE
    Date : 2021-06-25
    Deadline : 2021-04-23
    Comment :
,
    JavaFx interface
    ---
    Owner : null
    State : DOING
    Date : 2021-06-25
    Deadline : 2021-06-25
    Comment :
]

, PROJECT WebScrapper
--------------------------
Members :
Next Appointement : null
Deadline : null
Tasks : []

]
```

Sur l'interface graphique, le principe est le même. On crée ou on édite un projet et nous avons accès à l'ensemble de ses informations :



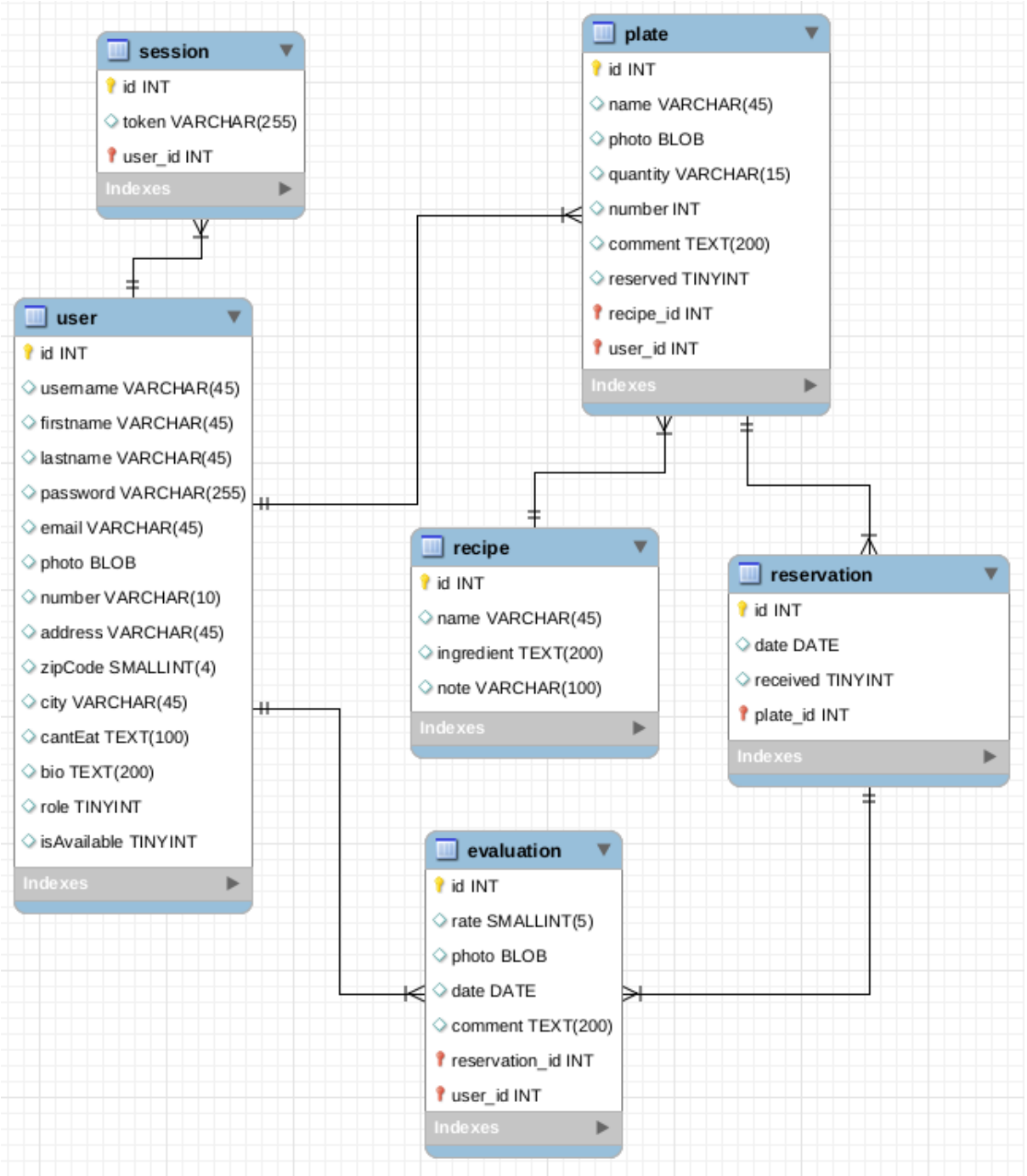
Page d'accueil de l'interface graphique



Page d'édition d'un projet

Base de données de l’application

Organisation des classes



-

Partie API

Fonctionnement

CRUD Mercifood API			
Method	Path	Role	Description
POST	/auth/subscribe	0	Create a new user
POST	/auth/login	0	Create a new session
POST	/recipes	0	Create a mew recipe
POST	/plates	0	Create a new plate
POST	/evaluations	0	Create a new evaluation
POST	/reservations	0	Create a new reservation
POST			
GET	/users	0	Show all users
GET	/users/{id}	0	Show a specific user by id
GET	/users/{name}	0	Show a specific user by name
GET	/users/availables		Show all available user
GET	/recipes	0	Show all recipes
GET	/recipes/{id}	0	Get a specific recipe
GET	/recipes/{name}	0	Get a specific recipe by name
GET	/plates	0	Show all plates
GET	/plates/{id}	0	Show a specific plate by id
GET	/plates/{name}	0	Show a specific plate by name
GET	/evaluations	0	Show all evaluations
GET	/evaluations/{username}	0	Create a new plate evalauation by username
GET	/reservations	0	Show all reservations
GET	/reservations/{id}	0	Get a specific reservation
GET			
PUT	/users/{id}	1	Update an existing user
PUT	/recipes/{id}	1	Update an existing recipe
PUT	/plates/{id}	1	Update an existing plate by id
PUT	/plates/{name}	1	Update an existing plate by name
PUT	/reservations/{id}	1	Update an existing reservation
PUT		1	
DELETE	/users/{id}	1	Delete an existing user

DELETE	/users/{id}	0	Unsuscribe an existing user
DELETE	/recipes/{id}	1	Delete an existing recipe
DELETE	/plates/{id}	1	Delete an existing plate id
DELETE	/plates/{name}	1	Delete an existing plate by name
DELETE	/reservations/{id}	1	Delete an existing reservation
DELETE			

Role => 0: User 1: Admin

Webscraping

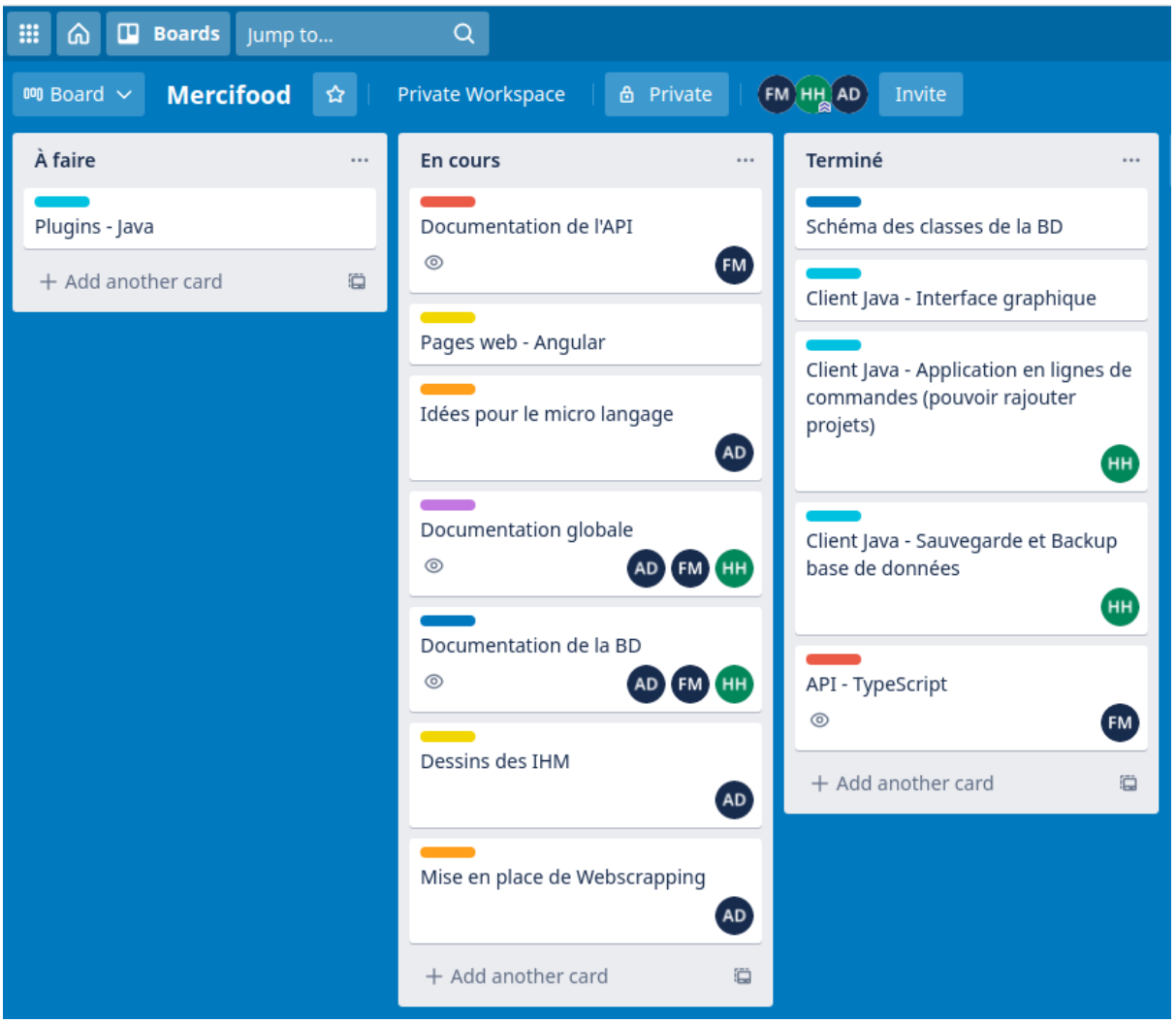
5.1 Utilisations

- Webscrapping Python pour récupérer des informations qui ne seront pas stockées dans la base de données ou qui seront utilisées pour des analyses.

5.2 Exemples

- Récupération des possibles allergies des personnes voulant recevoir des repas pour pouvoir trier automatiquement les repas que les personnes peuvent voir ou non.

Trello


Boards
Jump to...
Board
Mercifood
Private Workspace
Private
FM HH AD
Invite
À faire
Plugins - Java
Add another card
En cours
Documentation de l'API
FM
Pages web - Angular
Idées pour le micro langage
AD
Documentation globale
AD FM HH
Documentation de la BD
AD FM HH
Dessins des IHM
AD
Mise en place de Webscrapping
AD
Add another card
Terminé
Schéma des classes de la BD
Client Java - Interface graphique
Client Java - Application en lignes de commandes (pouvoir rajouter projets)
HH
Client Java - Sauvegarde et Backup base de données
HH
API - TypeScript
FM
Add another card