

Rapport de fin de projet

DEPUIS JANVIER 2022

Équipe de conception

- Mathys Gelez en charge de la conception de la base de données et de la réalisation de la documentation du compte rendu.
- Rayane El Asraoui en charge de la conception de la base de données et de la gestion du groupe.
- Robin Naulleau en charge de la programmation Java et de l'affichage sur SceneBuilder.
- Arnaud Rein en charge de la programmation Java, de l'affichage sur SceneBuilder et responsable de la gestion des commit sur github et de la partie javafx du compte rendu.

Le contexte

Des visites sont effectuées dans des centres médicaux afin de mettre à jour les connaissances sur le personnel médical et les produits utilisés.

Ces visites sont remboursées par l'entreprise Galaxy, qui souhaite avoir une application qui permettra d'uniformiser le suivi des visites.

Nous avons développé cette application qui permet d'informatiser et d'uniformiser le suivi des visites.

Fonctionnalités de l'application

L'application doit permettre à un utilisateur de s'identifier avec un login et un mot de passe.

Si l'utilisateur est un **visiteur**, il pourra remplir des fiches de remboursement (remplir le nombre de nuitée, de repas,...)

Si il est un **Comptable**, il pourra analyser les fiches de remboursement remplie par les visiteurs afin de les valider pour que celle-ci puisse être archivé et que l'entreprise puisse prendre en charge le remboursement.

Outil et langage utilisés pour la conception de base de données

En ce qui concerne la schématisation la base de donnée, nous avons utilisé LibreOffice calc (pour le dico de données) et Mocodo (pour le MCD).

Pour concevoir la base de données nous avons utilisés MySQL Workbench (utilisant MySQL comme langage de BDD).

Outil et langage utilisés pour la programmation de l'application

Nous avons eu recours à l'IDE Visual Studio Code pour écrire le code utiliser les commandes nécessaires à la création de "commit" sur Github.

Pour créer l'interface de l'application, nous avons utilisés Scene Builder, ce qui nous à permis de simplifier l'implémentation de bouton et de zone d'écriture lors des modifications.

**Documentation
Utilisateur**

Fiche utilisateur (Documentation utilisateur) :

- Pour débuter il faut rentrer son nom et son mot de passe qui sont dans la base de données workbench

	id_vi	matricule	nom	prenom	fk_ve	password	type_visiteur
▶	1	1A	rein	arnaud	1	rein	visiteur
	2	2A	gahide	patrice	2	java	comptable
	3	3A	bangalter	thomas	3	daft	visiteur
	4	4A	musk	elon	4	mars	comptable
	5	5A	white	walter	1	heisenberg	visiteur
	6	1B	depp	jhonny	2	pirate	comptable
	7	2B	cartman	eric	3	kfc	visiteur
	8	3B	terre	jean	4	pelle	comptable
	9	4B	macron	emmanuel	2	lrem	visiteur
	10	5B	momota	kento	3	yonex	comptable

The screenshot shows a login interface on an orange background. At the top, there are two input fields: the first contains the text 'rein', and the second contains four dots '....'. To the right of these fields is a button labeled 'se connecter'. Below the password field, the text 'rein' is displayed, and a small button labeled 'voir password' is positioned below it. At the bottom of the form, there are three buttons: 'Fiche à remplir', 'Fiche à valider', and 'validation ou non'.

Ensuite on appuie sur « se connecter » et selon que l'on soit « **comptable** » ou bien « **visiteur** » dans notre jeu de données, on arrivera sur une page bini spécifique pour « **comptable** » ou « **visiteur** ».

Si nous sommes visiteur nous atterrirons sur cette page « fiche de remboursement » où le visiteur va rentrer toutes les informations pour peut-être se faire rembourser.

Retour menu

Remboursement des frais

Visiteur

1978

De Niro

Robert

12/05/2022

2022-05-12

frais forfaitaires

Quantité

Montant unitaire

Total

nuitée

5

500

2500

Repas Midi

6

70

420

Kilométrage

50

10

500

clique pour convertir

3420

Autres frais

date

19/05/2022

libellé

taxi

Montant

500

signature

insérer dans la bdd et envoyer fiche au comptable

La partie en rouge correspond aux informations de celui qui va envoyer les valeurs

Il faut obligatoirement appuyer sur le bouton « cliquer pour convertir » si l'on veut avoir les totaux et que ces valeurs rentrent dans la base de données

Lorsque tout est rempli sans avoir laissé de blanc ou ayant laissé « null » si il n'y avait rien à mettre, c'est le comptable qui va se charger de la suite :

6

Le comptable après s'être login comme pour le visiteur, il n'a plus qu'à rentrer le matricule et le nom et appuyer sur « afficher résultat » pour obtenir toutes les informations sur la fiche de frais

Remboursement frais engagés

visiteur

matricule	date création	nom	prenom
<input type="text" value="1A"/>	<input type="text" value="2000-11-11 00:00:00"/>	<input type="text" value="rein"/>	<input type="text" value="arnaud"/>

frais forfaitaires

	Quantité	Montant unitaire	Total
nuitée	<input type="text"/>	<input type="text"/>	<input type="text"/>
Repas Midi	<input type="text"/>	<input type="text"/>	<input type="text"/>
Kilométrage	<input type="text"/>	<input type="text"/>	<input type="text"/>

Date validation

Lieu validation

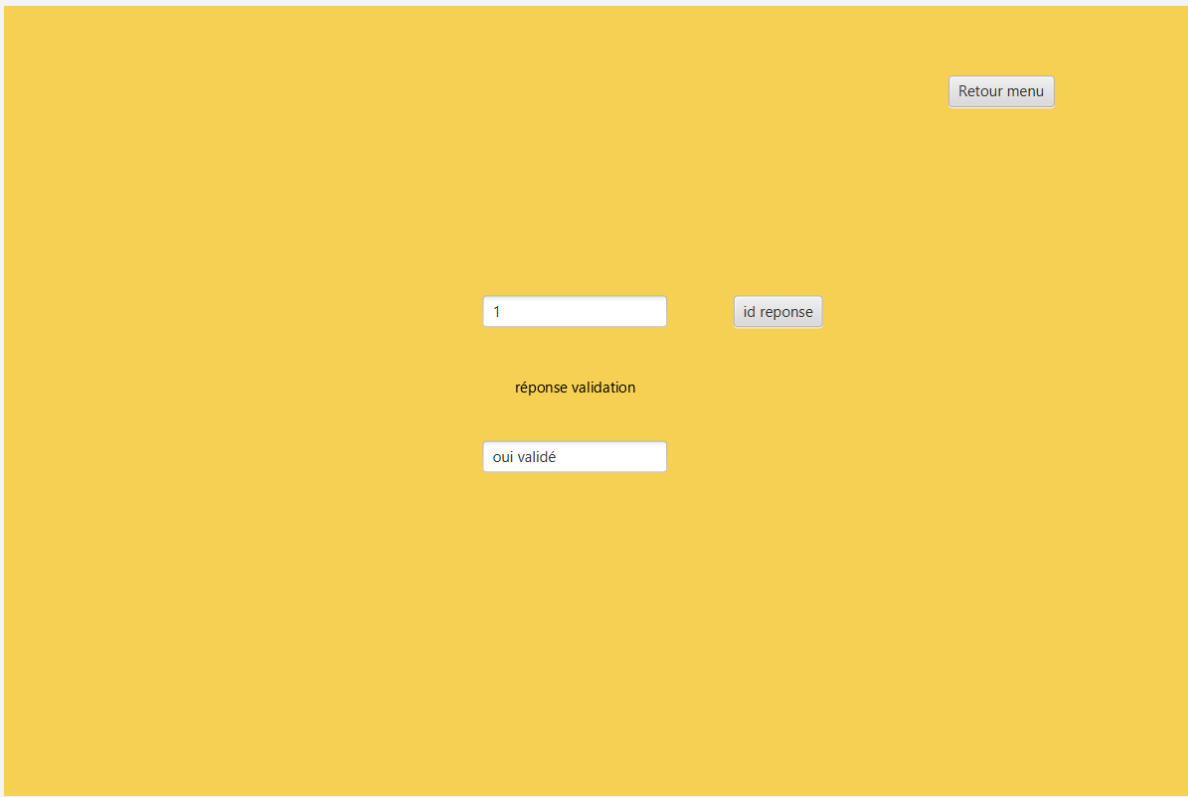
validation

autres frais

Date	Libellé	Montant
<input type="text" value="2000-11-"/>	<input type="text" value="externe"/>	<input type="text" value="50"/>

Ensuite, le comptable va déterminer si la fiche de frais est remboursable ou non, pour cela il indiquera la date de validation + le lieu + le commentaire qui valide si c'est remboursable dans la section « validation ». Si la fiche n'est pas remboursable, le comptable met en commentaire de la section validation : « non remboursable + la raison »

Pour conclure sur la présentation, nous allons voir la page de validation qui est encore à l'état de construction, là on voit que lorsque je rentre l'id d'un visiteur, on voit tout de suite si sa fiche est remboursable ou non, par la suite il y aura une liste déroulante avec les différentes fiches + les dates.



The screenshot shows a web form with a yellow background. In the top right corner, there is a button labeled "Retour menu". In the center, there is a text input field containing the number "1". To the right of this field is a label "id reponse". Below the input field, the text "réponse validation" is displayed. At the bottom, there is another text input field containing the text "oui validé".

CONCEPTION
BASE DE DONNÉES



Pour commencer nous avons établi le dictionnaire de données en s'appuyant sur l'exemple de fiche de remboursement donné dans le cahier des charges du projet.

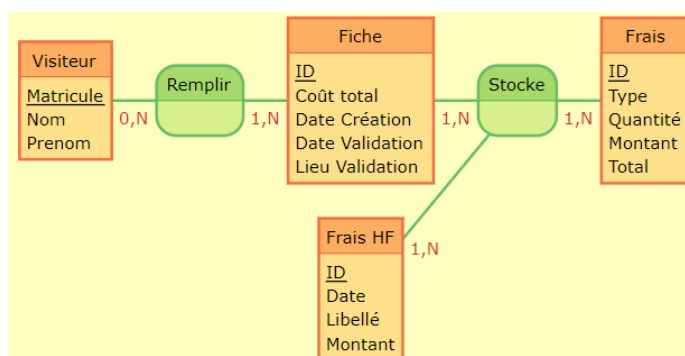
	A	B	C	D	E	F
1	données	type	taille	genre	id potentiel	obligatoire
2	fiche	alphanumérique	45	élémentaire	non	oui
3	Visiteur nom	alphanumérique	45	élémentaire	non	oui
4	Visiteur prenom	alphanumérique	45	élémentaire	non	oui
5	Visiteur matricule	alphanumérique	45	élémentaire	oui	oui
6	Date fiche	temporel		élémentaire	non	oui
7	Type frais forfaitaires	numérique	5	élémentaire	non	oui
8	quantités frais	numérique	5	élémentaire	non	oui
9	montant unitaire frais	numérique	5	élémentaire	non	oui
10	total frais	numérique	5	calculé	non	oui
11	date frais extra	temporel		élémentaire	non	non
12	libellé frais extra	alphanumérique	120	élémentaire	non	non
13	montant frais extra	numérique	5	élémentaire	non	non
14	Coût total	numérique	10	calculé	non	oui
15	validation fiche	logique	2	élémentaire	non	oui
16	date validation fiche	temporel		élémentaire	non	oui
17	lieu validation fiche	alphanumérique	45	élémentaire	non	oui

Il est indispensable pour nous d'établir ce document car il permet d'établir une liste complète de toutes les données et définir leurs caractéristiques.

Ensuite, en se basant sur le dictionnaire de données, nous avons réparti les données dans des entités. Par exemple nous avons regroupé dans une entité «visiteur», le nom de l'utilisateur, le matricule,...

Ensuite nous avons construit une première version du MCD basé sur nos entités.

Première version du MCD:

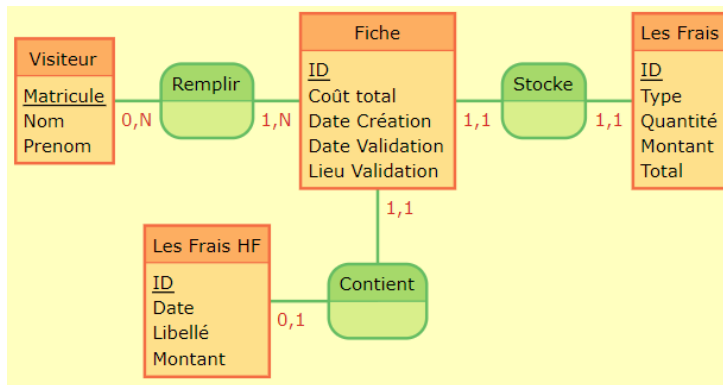


Nous nous sommes rendu compte que cette première version ne correspondait pas et que la mise en place de la base de données ne pouvaient pas se faire sur cette

première version du MCD.

Alors nous avons mis en place la semaine suivante une nouvelle version du MCD.

Deuxième version du MCD:

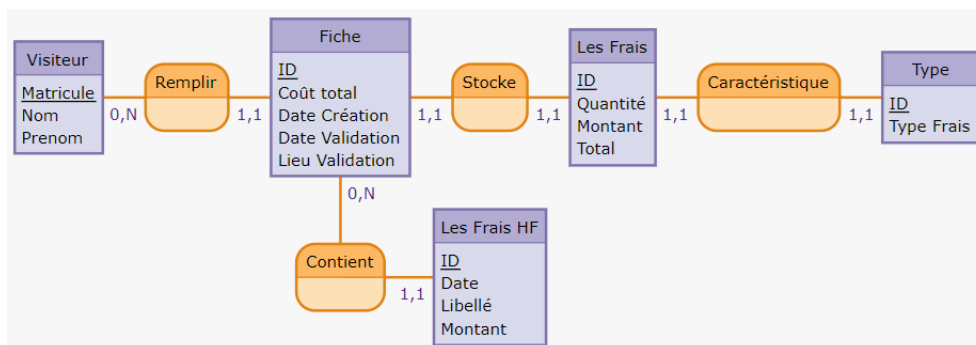


Dans cette version nous avons supprimé l'association bizarre qui n'était pas possible entre fiche, frais et frais HF pour la remplacer par des nouvelles entités avec des nouvelles associations.

Malgré les modifications, cette version ne convient pas et possède d'important problèmes en ce qui concerne les associations et les cardinalités.

Nous avons tout retravailler pour mettre en place une troisième version du MCD.

Troisième version du MCD:



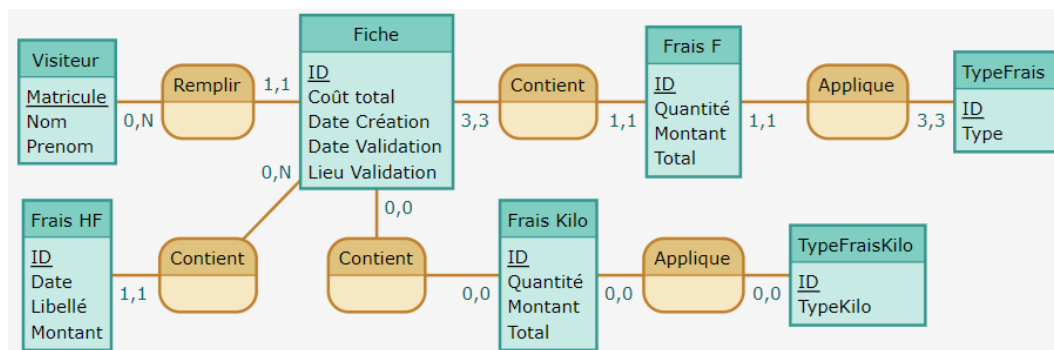
Dans cette version nous avons ajouter une l'entité «Type» pour définir le type de frais

dans la fiche de remboursement et nous avons révisé les cardinalités visiteur/fiche et fiche/frais HF.

Mais cette version est incomplète, les cardinalités ayant été laissées de base sur 1.1 sans avoir eu de modification. De plus nous nous sommes rendu compte que le MCD n'avait aucune données ou entités qui pouvait prendre en compte la partie sur le kilométrage et le type de véhicules (ce qui est défini dans le cahier des charges).

Pour palier à ces problèmes, nous avons eu recours à une remise à jour du MCD pour atteindre une quatrième version.

Quatrième version du MCD:

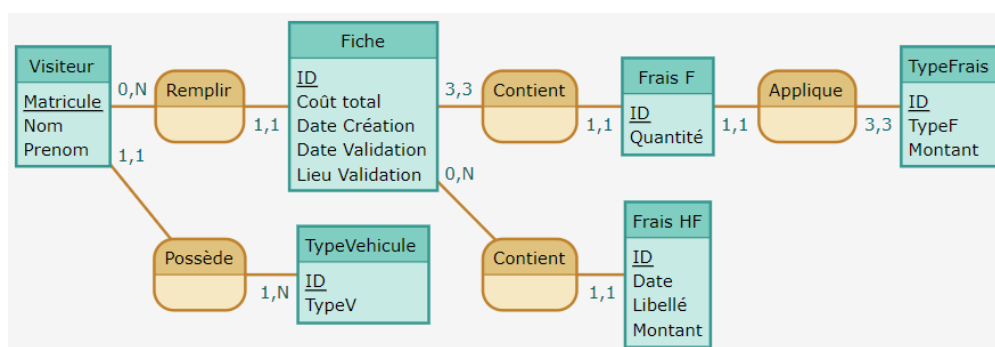


Dans cette version nous avons ajouté deux entités: «Frais Kilo» et «TypeFraisKilo» pour ajouter la partie qui concerne le type de véhicule des utilisateurs. Nous avons également remplacé les cardinalités qui ne correspondaient pas.

Avant même de la finir nous nous sommes rendu compte que cela ne pouvait pas fonctionner correctement ou du moins que cette architecture était inutilement complexe à cause de l'ajout des nouvelles entités en plus d'ajouter quelques problèmes. Ce plan n'était donc pas envisageable pour la conception de la base de données

Donc nous nous sommes basés sur ce MCD pour établir la version finale du MCD:

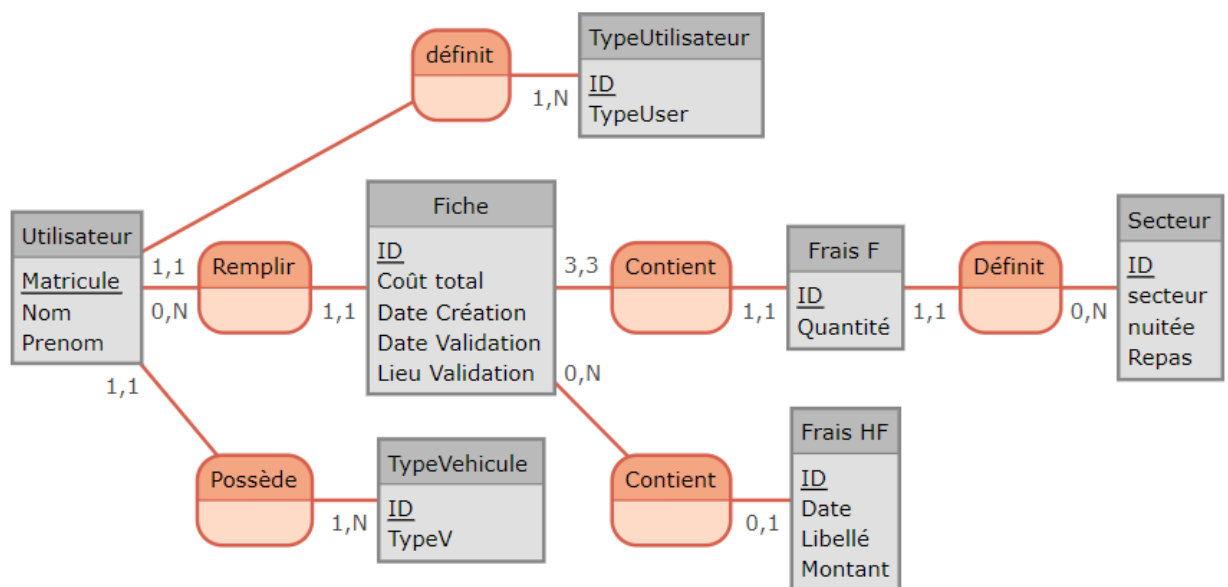
Cinquième et dernière version du MCD:



Sur cette version du MCD nous avons gardé les quatres entités «Visiteurs, Fiche, Frais F et TypeFrais». Nous avons remplacé toute la partie qui devait permettre de prendre en charge le kilométrage et le type de véhicule par une entité «TypeVéhicule» en association avec l'entité «Visiteur». Ainsi nous prenons en compte qu'un visiteur possède un type de véhicule pour réaliser les transports.

Cette version du MCD est très complète. Il ne manque que quelques détails pour qu'il convienne à 100% aux besoins du projet.

Paufinage du MCD:



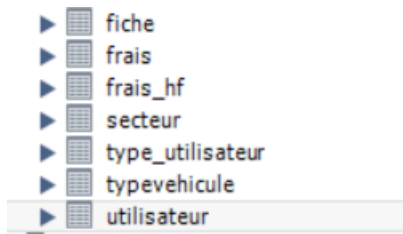
Nous avons ajouter quelques modifications:

- Ajout de l'entité «TypeUtilisateur» associé à l'entité «Utilisateur» qui permet de distinguer les visiteurs des comptables.
- Changement de nom: L'entité «Visiteur» est renommé «Utilisateur» pour éviter toute confusion.
- Transformation de l'entité «TypeFrais» pour qu'elle devienne «Secteur». Ainsi cette entité permet de donner un prix pour la nuitée et le repas en fonction du secteur de travail (région par exemple).

Grâce à cette dernière et ultime version du MCD, nous avons pu concevoir la base de donnée via MySQL Workbench.



Nous avons créer les différentes table et leurs contenu:



Dans chaque tables nous avons créer les champs dont ceux de clés primaire afin que cela correspondent au MCD.

Exemple avec les champs de la table secteur:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_ty	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
secteur	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
nuitée	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
repas	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

«**id_ty**» correspond à la clé primaire

Nous avons également créer les clés étrangères en se basant sur les cardinalités présente sur le MCD.

Exemple avec les champs de la table Utilisateur.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
matricule	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
nom	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
prenom	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
fk_ve	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
fk_tu	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Ici vous pouvez voir que les champs «**fk_ve**» et «**fk_tu**» sont des champs de clé étrangère (on peut le remarquer aux marqueurs rouge)

Les clés étrangères sont présentes dans cette tables car de manière générale, les champs de clés étrangères sont placé dans la tables qui contient la cardinalités maximales la plus petite.

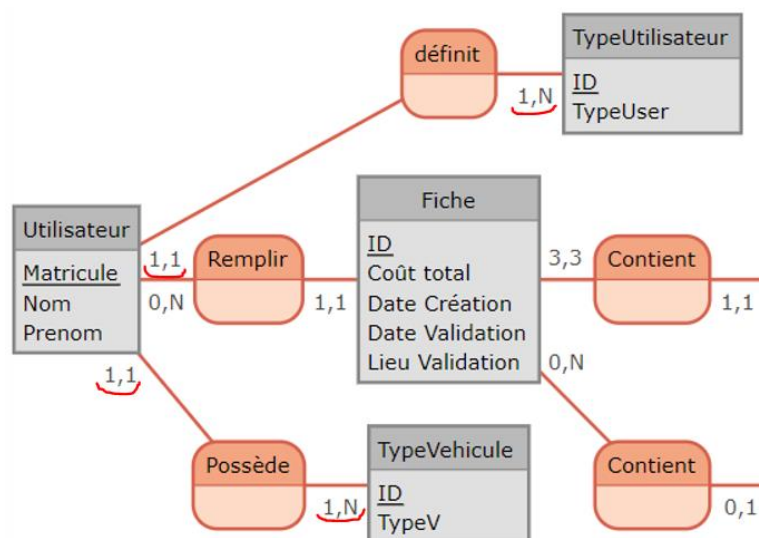
Cardinalités maximales

Utilisateur / TypeUtilisateur

1 < N

Utilisateur / TypeVehicule

1 < N



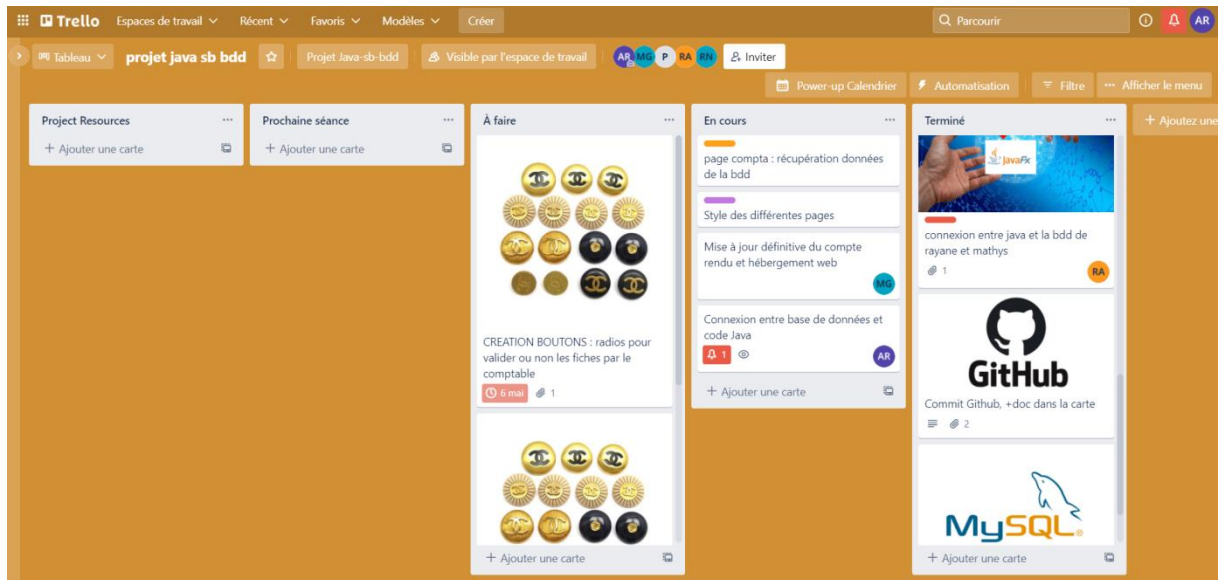
Nous avons rajouté un champ correspondant aux mot de passe dans la table «utilisateur» pour sécuriser les accès lors de l'identification dans l'application.

Ainsi, en suivant les étapes, la base de données est enfin correctement construite et prête à être utilisé.



OUTILS DE GESTION DE PROJET

Nous avons utilisé TRELLO, dont vous êtes membre VIP :





EXPÉRIENCE --- DE TRAVAIL

Bilan de cette première expérience :

ROBIN :

J'ai appris à faire des git avec plusieurs vidéos

Donc installer git un git config, git init, gitignore, git status et git add

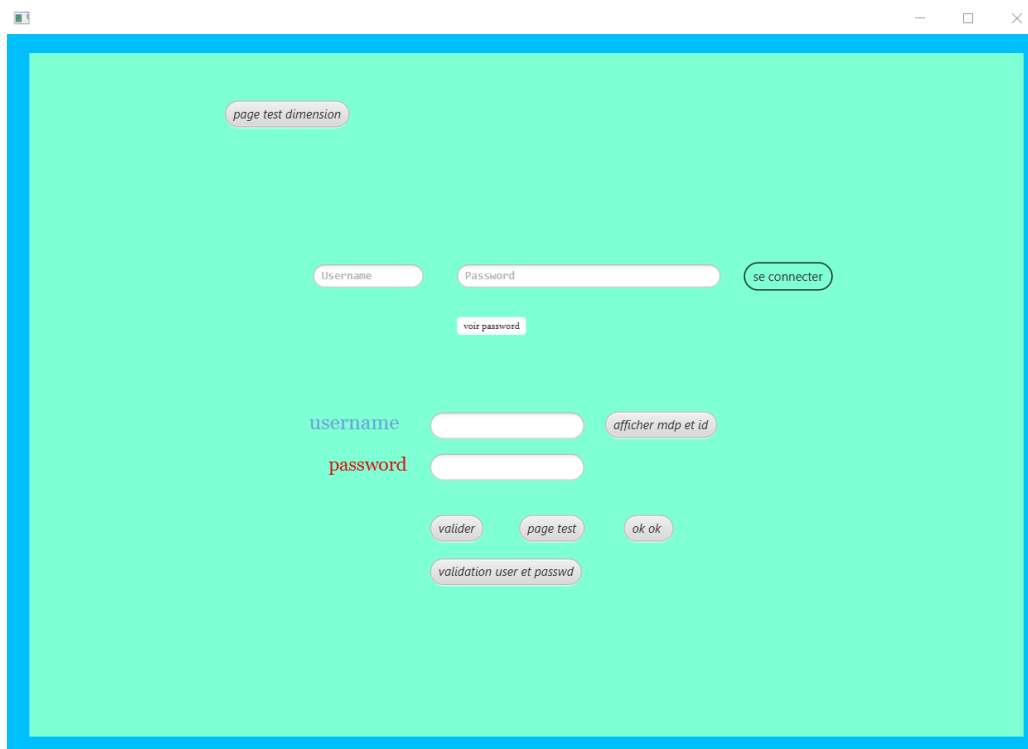
Ensuite regarder comment la bdd avait été faite par Rayane et Mathis pour mieux la comprendre avec le MCD et les tables avec les différents champs et la création de jeux de données

Correction de plusieurs bugs du code Java FX et scenebuilder

Créations et modifications du style sur les différentes pages Java FX et scenebuilder

Modifications du code visiteurs et comptables

Fait plusieurs test avec différentes bdd de données de test et la connexion avec le code Java pour comprendre le fonctionnement puis ajout de la base de données de Rayane et Mathis



matricule

nom

ID

Button

Date validation

affiche resultat

retour au menu

Remboursement frais engagés

visiteur

matricule

date création

nom

prenom

frais forfaitaires

	Quantité	Montant unitaire	Total
nuitée			
Repas Midi			
Kilométrage			

autres frais

	Date	Libellé	Montant
Total		lieu de validation	Montant total

ARNAUD REIN :

Cela a été une très bonne expérience de travail, j'ai appris énormément de choses :

- Base de données :
J'ai appris au fur et à mesure à maîtriser les bdd tel que créer des champs et des clés étrangères que je ne maîtrisais pas totalement au début du projet, le projet m'a permis d'appliquer ce que l'on apprenait avec Monsieur Lempereur tel que « les jointures, les updates, insert into,... »
- SceneBuilder :
J'ai eu du mal à bien comprendre comment ça marchait au début du projet, mais comme pour les Bases de données au fur et à mesure d'écrire des lignes de codes et d'y associer des ID sur des textfields, des actions à des méthodes,... En fait ce logiciel devient un indispensable car je ne pourrai plus créer une page sans lui, quand on voit toutes les possibilités qu'il offre, le temps qu'il nous fait gagner, les lignes de codes qu'il nous fait épargner, la clarté et surtout la facilité dont il fait preuve font que j'ai apprécié travailler avec.
- JavaFx :
Certes il a été ma bête noire durant les premières semaines du projet, mais par la suite j'ai su l'apprécier tout comme SceneBuilder, mais là je ne pourrai pas donner un avis objectif car je n'ai pas produit d'autres logiciels avec java et encore moins avec un autre langage de programmation à la différence de SceneBuilder où je pouvais un peu comparer avec le css. Donc pour conclure je pense que c'est un très bon langage pour ce genre de choses, à voir par la suite.

Pour conclure sur les logiciels que j'ai appris :

J'ai pris beaucoup de plaisir durant le dernier mois à coder, associer les langages entre eux, connecter la base de données, créer des méthodes pour des logins, pour récupérer des données dans la bdd et les réutiliser. J'en garderai un bon souvenir car lorsque l'on réussit quelque chose comme connecter la bdd à javafx, on est super content et très fier, on a envie de continuer et de voir progresser le logiciel.

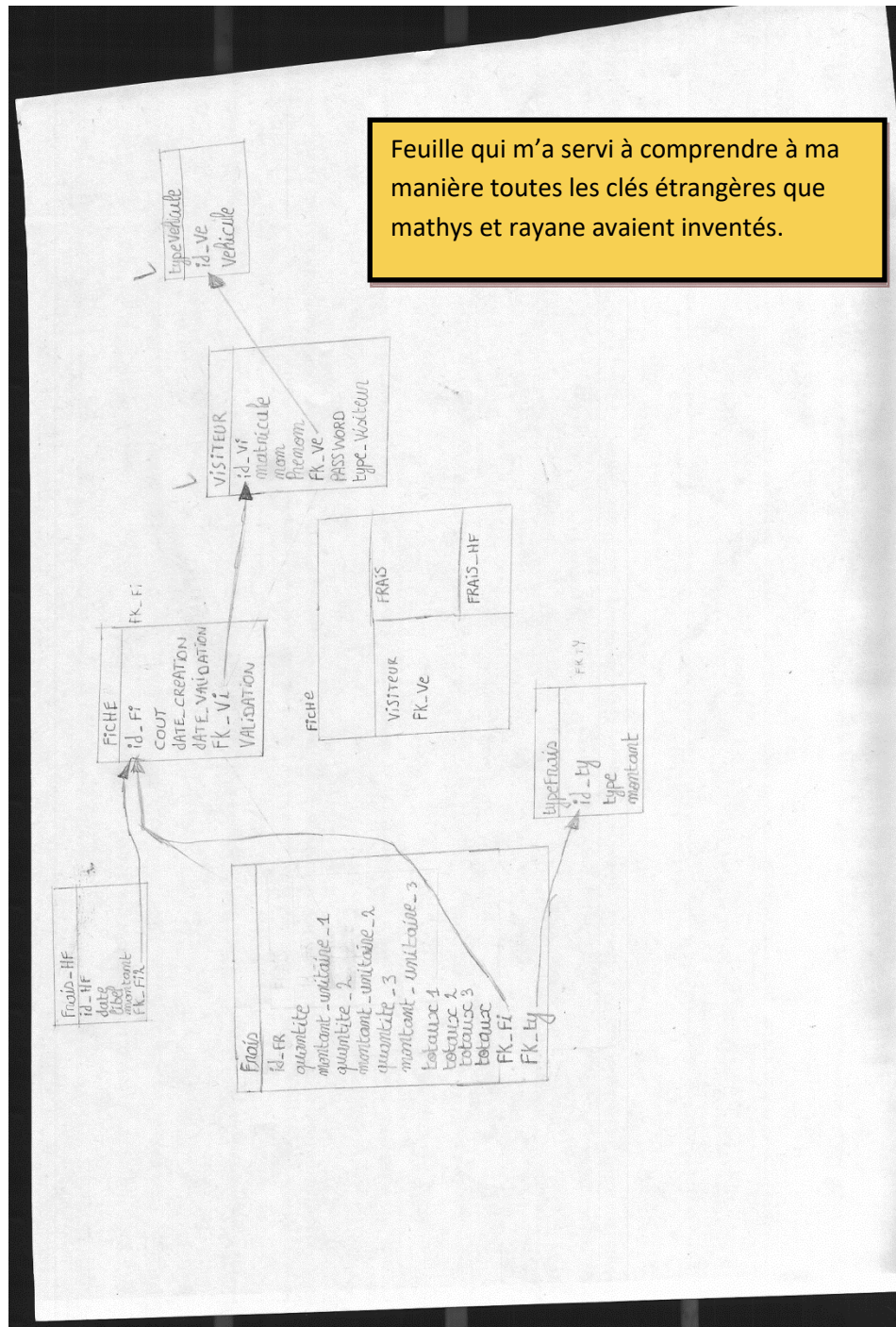
Le travail en équipe :

Comme on en avait parlé à la fin du test de notre logiciel le 09/05, il faut :

1. Beaucoup communiquer
2. Prendre 15 minutes sur chaque séance pour que chacun partage les avancés qu'il a fait et qu'il nous les enseigne
3. Savoir synchroniser les tâches que l'on fait (dans notre cas c'était la bdd de mathys et rayane avec la javafx)

Etre assidu sur un outil de gestion de projet pour bien répartir les tâches et voir l'avancement à tout moment et surtout pour comprendre ces dernières.

J'ai aussi appris à poser les choses sur des feuilles pour que mes idées soient bien claires comme dans les exemples suivants (j'en ai beaucoup plus mais je ne voulais pas tous mettre)



Là encore c'est une feuille qui me permet de comprendre à ma manière la fonction rs.next, mais moi je l'ai renommé la « 777 »

FONCTIONNEMENT RS.NEXT:

```

WHILE (rs.next()) {
    String Name = rs.getString(columnLabel: "Nom");
    String Password = rs.getString("password");

    if (rs.getString(columnLabel: "nom").equals("nom")) {
        switch To Primary();
    }
}

```

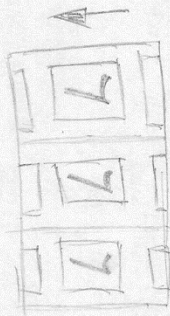
ID	NOM	password
1	ALANAD	KEIN
2	THOMAS	SAUGALTER
3	JULIE	KEIN

```

if (nom & password) {
    switch
}

```

3
switch
nom password



LES COMMITS

J'ai aussi appris à faire des commits (même si lors du compte rendu une erreur de « git push » est venu s'immiscer mais tout fonctionner à merveille par le passé) :

```
29 git config --global user.name "arnaudofficiel"
30 git config --global user.mail "arnaud.rein029@gmail.com"
31 git config --global --list
32 git init
33 git status
34 git status
35 git add .
36 git status
37 git commit -m "premier commit" -m"version qui fonctionne"
38 git status
39 git status
40 git commit -m"test commit pour voir le git log"
41 git status
42 git log
43 git add .
44 git status
45 git commit -m"test commit git log"
46 git status
47 git log
48 git status
49 git remote add origin https://github.com/arnaud-rein/APSBJAVAFX.git
50 git remote -v
51 git push -u origin master
52 git push
53 git push
54 git push
55 git status
56 git add .
57 git status
58 git push
59 git commit -m"troisième commit de test"
60 git log
61 git push
62 history
```

De la ligne 29 à 32 c'est la configuration de notre commit (dépôt) sur notre projet.

Ensuite on va faire un « git status », tout les fichiers en rouge sont des fichiers qui ont été modifié depuis le dernier commit et qui n'ont pas encore été resauvegarder.

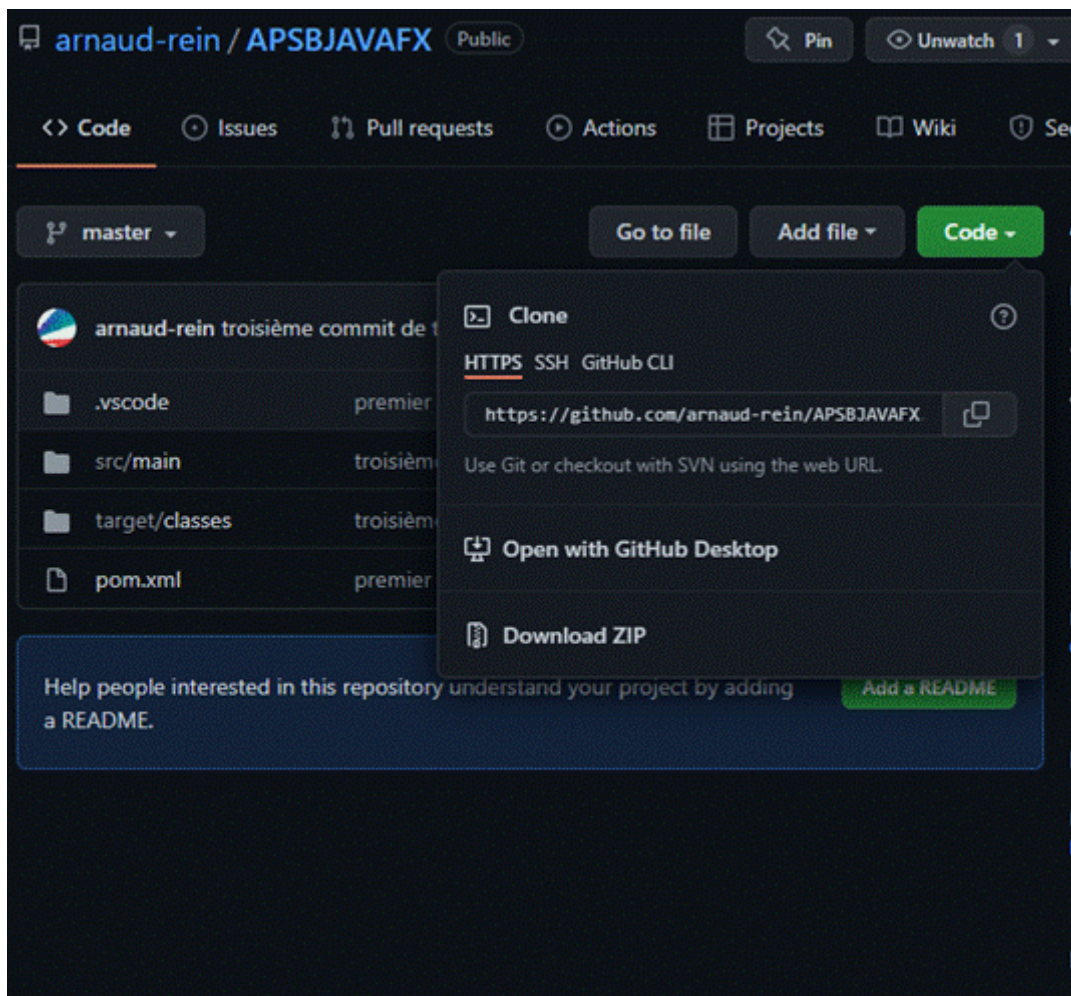
Pour les sauvegarder, on va les ajouter pour les sauvegarder en faisant la commande « git add . »

Et on va enfin faire le commit en faisant la commande « git commit -m « nom du commit » - m « autre nom » »

Ensuite on refait un « get status » et si il y a n'y a plus rien qui apparait ca veut dire que tout s'est bien envoyé.

Et pour voir tout les commit que l'on a fait, on fait un «git log »

Ensuite pour envoyer le commit sur github, on créer un repository



On récupère le lien

On fait la commande « git remote add origin nomlien » pour associer notre vscode à notre repository

Ensuite on enchaîne les commandes « git remote -v » et « git push -u origin master » et ensuite on peut commit en faisant :

« Git push »

A chaque fois que l'on voudra faire un commit on devra répéter ces commandes :

1. « git status »
2. « git add . » ou « git add nomfichier »
3. « git commit -m « nomcommit » »
4. « git push »

INNER JOIN

Et voici un exemple d'explications que j'essaie de plus en plus mettre en place afin de toujours comprendre mon code, dans ce cas là c'est le INNER JOIN (car je pense que lorsque je vais revenir regarder le code dans quelques mois, ça sera super important d'avoir ces explications pour que je comprenne) :

INNER JOIN JAVA FX

```
String sql = "SELECT id_vi, matricule, nom, prenom, fk_ve, id_ve, vehicule FROM visiteur INNER JOIN typevehicule ON visiteur.fk_ve = typevehicule.id_ve WHERE id_vi='"+idnombre+"'";
```

Dans la requête on voit apparaitre un inner join, il nous sert à récupérer le type de diesel ou bien d'essence du véhicule à partir de fk_ve qui lui renvoie à la table «type véhicule» et qui par : visiteur.fk_ve = typevehicule.id ne nous renverra pas l'ID de la table typevehicule mais renverra la valeur véhicule grâce à :

```
labmontant.setText(rs.getString("vehicule"));
```

<https://sql.sh/cours/jointures/inner-join>

Et il y aurait encore beaucoup à montrer sur ce que j'ai appris mais je m'arrête ici.