Dossier de Projet Professionnel

Création de l’application Notes de frais

Présenté par Mohamed HADJADJI

Sommaire

1. Résumé de Projet

Ce dossier présente mon projet professionnel en tant qu'un étudiant au centre de formation *La Plateforme,* pour le titre *RNCP Concepteur Développeur d'application*. Le projet est de créer l'application *Notes de frais,* pour les entreprises qui souhaitent simplifier le traitement de leurs notes de frais.

Le salarié peut saisir une demande de remboursement pour une note de frais. Il devra uploader un justificatif (par exemple, une photo d'une facture d'un restaurant, ou un fichier PDF) et saisir le montant de la note. Il lui est possible d'ajouter un commentaire. Il doit également écrire son nom. Dans une autre partie de l'application réservée au service RH, il est ensuite possible de valider ou de refuser les notes de frais.

Le projet devra obligatoirement posséder une API REST, les quatres opérations de base pour la persistance des données (CRUD) devront être implémentées dans l'API. Je devrai utiliser le framework *CodeIgniter* pour développer mon projet. Enfin, le projet devra être développé en *React Native*.

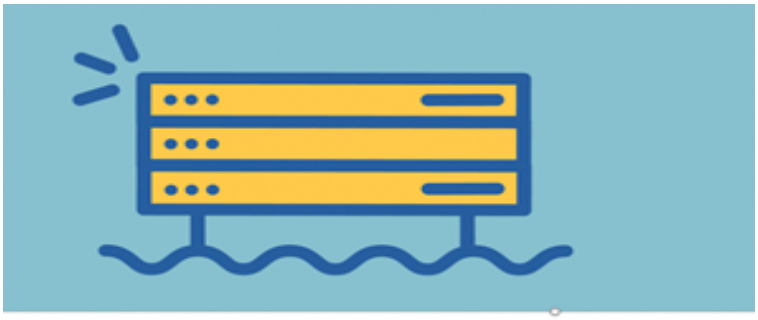
Mon rôle dans ce projet est de développer le back-end (API) en utilisant CodeIgniter 3, et l'interface de l'application en *React Native*, de la partie *Salarié*.

2. Présentation

2.1 Présentation personnelle

Je m'appelle Mohamed HADJADJI, j'habite à Marseille, j'ai un diplôme *d'ingénieur en économie agroalimentaire*, et un *titre professionnel RNCP Développeur Web et Web Mobile.* J'ai une expérience professionnelle dans l'informatique de gestion. Et maintenant je suis en contrat de professionnalisation et en formation pour le titre *RNCP Concepteur développeur d'application* à La Plateforme.

2.2 Présentation de La Plateforme



La Plateforme_
Marseille

La Plateforme est une école du numérique et des nouvelles technologies cofondée avec le Club Top 20 réunissant les grandes entreprises de la Métropole Aix Marseille. Elle comprend une offre de formations diversifiées destinées à former des codeurs et développeurs web, des experts en sécurité, des ingénieurs spécialisés en Intelligence Artificielle, et des cadres d'entreprises au travers de cycles de formations continues. La Plateforme est membre du programme Grande École du Numérique . Elle est soutenue par de grandes entreprises du territoire comme le Crédit Agricole Alpes Provence , par la Région Sud , le Département des Bouches du Rhône et la Métropole Aix Marseille Provence .

2.2 Présentation de l'équipe et répartition des tâches

Mon équipe avec laquelle j'ai travaillé sur ce projet regroupe 3 développeurs web, de formation à La Plateforme, et qui sont:

➢ Gregory FAUVEL

Il s'occupait des tâches suivantes:

- Maquetter le MCD de la BDD.
- Créer la base de données.
- Maquetter l'arborescence de l'application.
- Développer le Back-end (API) de l'application côté RH.
- Développer l'interface de l'application avec React Native côté RH.

➢ Moi même

Je m'occupais des tâches suivantes:

- Maquetter l'application.
- Le diagramme de Gantt.
- Hébergement de l'API dans Plesk.
- Développer le Back-end (API) de l'application côté Salariés.
- Développer l'interface de l'application avec React Native côté Salariés.

➢ Olivier CROZET

Il s'occupait des tâches suivantes:

- Création de GitHub.
- L'organisation de projet avec Trello.
- Développer le Back-end (API) de l'application côté Salariés et RH.
- Développer l'interface de l'application avec React Native côté Salariés et RH.

3. Compétence référentiel couvert par le projet

3.1 Connaissance en framework

- **CodeIgniter**

Pour développer L'API REST côté *salariés* j'ai utilisé *CodeIgniter 3*, un framework qui utilise la structure *MVC* **(Model - View - Controller)**. Et j'ai implémenté les quatres opérations de base pour la persistance des données (CRUD). Donc j'ai créé un model (Note Model), qui contient toutes les fonctions avec les requêtes d'accès aux données des notes de frais, après j'ai utilisé le controller pour créer le Contrôleur (Notes), qui utilise les fonctions de modèle pour récupérer ou envoyer les informations en JSON. Et chaque fonction de contrôleur représente une route de l'API.

Dans ce projet, je n'ai pas utilisé le View de CodeIgniter, car l'affichage se fait au niveau de l'application avec React Native.

- **React Native**

j'ai développé l'application note de frais avec **React Native,** c'est un framework qui utilise le langage de programmation JavaScript et les librairies React. Donc j'ai installé **Expo** qui est un ensemble d'outils construits autour de React Native, et qui m'a permis d'avoir un simulateur directement sur mon téléphone, pour pouvoir créer et modifier les différentes pages de l'application.

3.2 Conception de l'application

- **Maquetter l'application**

J'ai créé la maquette qui illustre toutes les pages avec *Adobe XD,* grâce à sa bibliothèque d'éléments très fournie qui m'a permis de répertorier les zones du site pour une structuration optimale de l'information, et d'avoir des détails des zonings avec

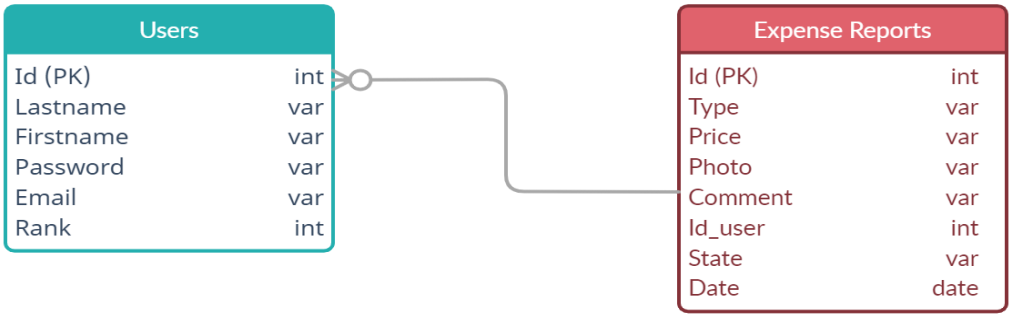
l'intégration du contenu dans les blocs. C'est un moyen de partager ma vision du projet avec les membres d'équipe et d'examiner les différentes possibilités.

Adobe XD

Dans Adobe XD, nous pouvons créer et manipuler nos designs et prototypes à l'aide de divers éléments, tels que les barres d'outils, les panneaux et l'inspecteur Propriétés. Ces éléments constituent l'espace de travail d'Adobe XD.

- **Conception et création de la BDD**

Pour la conception de la BDD j'ai réaliser un MPD avec l'outil https://app.creately.com/



J'ai utilisé phpMyAdmin, pour la création de la BDD "EMS", qui contient les deux tables **(Users - Expense Reports)**

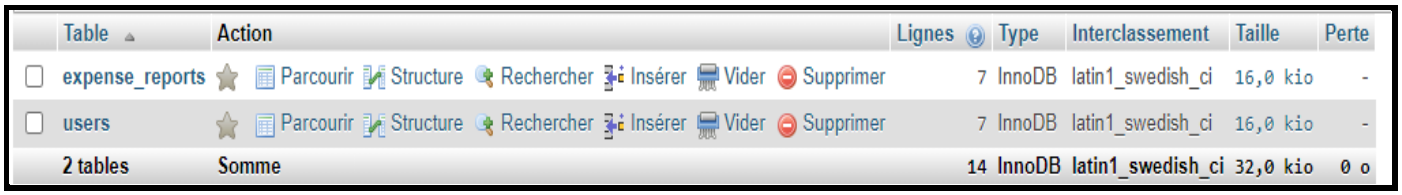


	Table	Action	Lignes	Type	Interclassement	Taille	Perte
☐	expense_reports	Parcourir Structure Rechercher Insérer Vider Supprimer	7	InnoDB	latin1_swedish_ci	16,0 kio	-
☐	users	Parcourir Structure Rechercher Insérer Vider Supprimer	7	InnoDB	latin1_swedish_ci	16,0 kio	-
	2 tables	Somme	14	InnoDB	latin1_swedish_ci	32,0 kio	0 o

3.3 Gestion de projet et organisation de l'environnement de développement

- **Trello**

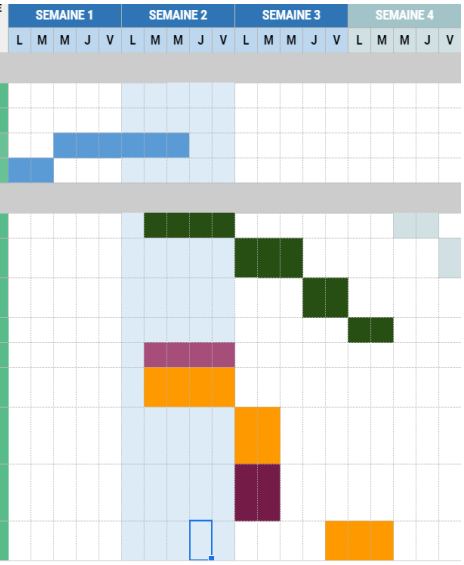
C'est l'outil de gestion visuelle du travail qui permet aux équipes de proposer des idées, de planifier, de gérer et de célébrer leur travail ensemble de manière collaborative, productive et organisée.

Que mon équipe et moi-même commençons quelque chose de nouveau ou que nous essayions d'organiser notre travail existant, Trello s'adapte à n'importe quel projet. Il nous aide à simplifier et standardiser le processus de travail de notre équipe de manière intuitive. Trello est convivial, mais permet de gérer les projets les plus complexes de notre équipe.

- **Diagramme de Gantt**

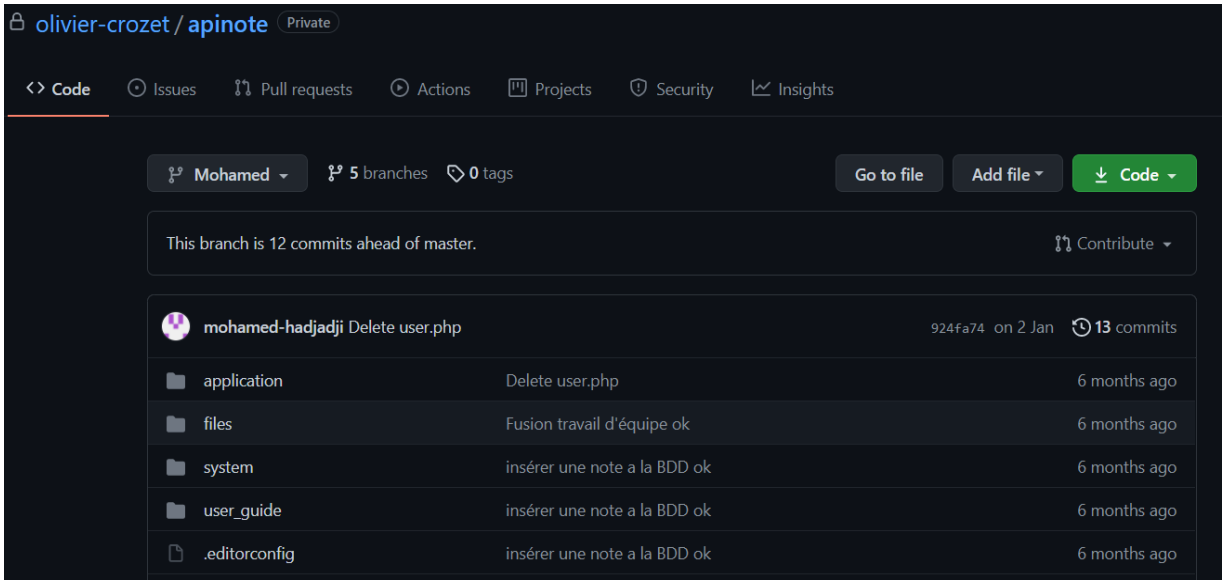
J'ai mis en place notre diagramme de gantt pour répertorier toutes les tâches à accomplir, et mener le projet à bien, et indiquer la date à laquelle ces tâches doivent être effectuées (le planning).

NUMÉRO	TITRE DE LA TÂCHE	PROPRIETAIRE DE LA TÂCHE	DATE DE DÉBUT	DATE LIMITE	DURÉE	TACHE TERMINEE (EN %)
1	**Conception et mise en route du projet**					
1.1	Diagramme bdd	Gregory	26/11/20	27/11/20	1	100 %
1.1.1	Creation bdd	Gregory	27/11/20	27/11/20	0	100 %
1.2	Documentation api	Gregory	15/01/21	21/01/21	6	90 %
1.3	Maquette	Mohamed	10/01/21	12/01/21	2	90 %
2	**Développement api**					
2.1	Insertion d'une personne	Gregory	22/12/20	26/12/20	4	100 %
2.2	Récupertion de la liste des personnes existantes	Gregory	27/12/20	30/12/20	3	100 %
2.3	Récupérer les informations détaillées d'une personne	Gregory	03/01/21	05/01/21	2	100 %
2.5	Supprimer une personne	Gregory	06/01/21	08/01/21	2	100 %
2.6	Créer une note de frais	Mohamed	22/12/20	26/12/20	4	100 %
2.7	Récupérer la liste des notes de frais	Mohamed	22/12/20	26/12/20	4	100 %
2.8	Récupérer les informations relatives à une note de frais	Mohamed	27/12/20	29/12/20	2	100 %
2.9	Mettre à jour les informations relatives à une note de frais	Olivier	27/12/20	29/12/20	2	100 %
2.10	Supprimer une note de frais	Mohamed	03/01/21	06/01/21	3	100 %



- **GitHub**

Nous avons ajouté deux repositories pour le git les fichier de code, ainsi que le merge (**apinote - appapinote).** pour les données de l'API et l'application. Mes fichiers que j'ai créés, je les ai git directement sur ma branche (Mohamed).



4. Cahier des charges

Le projet doit respecter certains critères, il devra obligatoirement posséder une API REST, développée par vos soins. Les quatres opérations de base pour la persistance des données (CRUD) devront être implémentées dans l'API. Je devrai utiliser le framework CodeIgniter pour développer le projet. Enfin, il devra être développé en React Native.

4.1 Une interface utilisateur pour le RH:

Les ressources humaines de l'entreprise ont besoin de voir la liste des salariés avec leur informations personnelles, et ils ont le droit de créer, modifier ou supprimer des comptes utilisateurs. Et ils ont aussi l'accès aux notes de frais ajoutées par les salariés, avec la possibilité de valider ou refuser ces notes de frais en cours.

Après la connexion l'utilisateur avec le statut *RH*, a une barre de navigation qui contient deux pages:

- **Page Liste d'utilisateurs:**

  C'est la page d'atterrissage, avec toutes les informations des utilisateurs (noms, prénoms, emails, statuts), et aussi trois boutons qui sont:

  - **Modifier:** Pour afficher la page *Modifier l'utilisateur, elle contient* un formulaire avec les informations de l'utilisateur, et la possibilité de les modifier.
  - **Effacer:** Pour la suppression d'un utilisateur, en cliquant sur le bouton une fenêtre d'alerte s'affiche, pour confirmer ou annuler.
  - **Notes:** Pour afficher la page *Notes d'utilisateur,* elle contient la liste de notes de frais de la personne concerné, plus un bouton pour modifier le statut (Validé, Refusé).

- **Page Ajouter un utilisateur:**

  C’est la page de création des comptes utilisateurs à l’aide d’un formulaire de création qui contient (nom, prénom, mot de passe, email, statut). Et un bouton pour la validation.

4.2 Une interface utilisateur pour les salariés

Le salarié à l'accès pour sa liste des notes de frais, avec leur informations. Il a la possibilité de créer et supprimer ses propres notes, ou les modifier si elles sont en mode en cours.

Après la connexion de salarié avec son email et le mot de passe, il a une barre de navigation qui contient deux pages:

- **Page Liste des notes:**

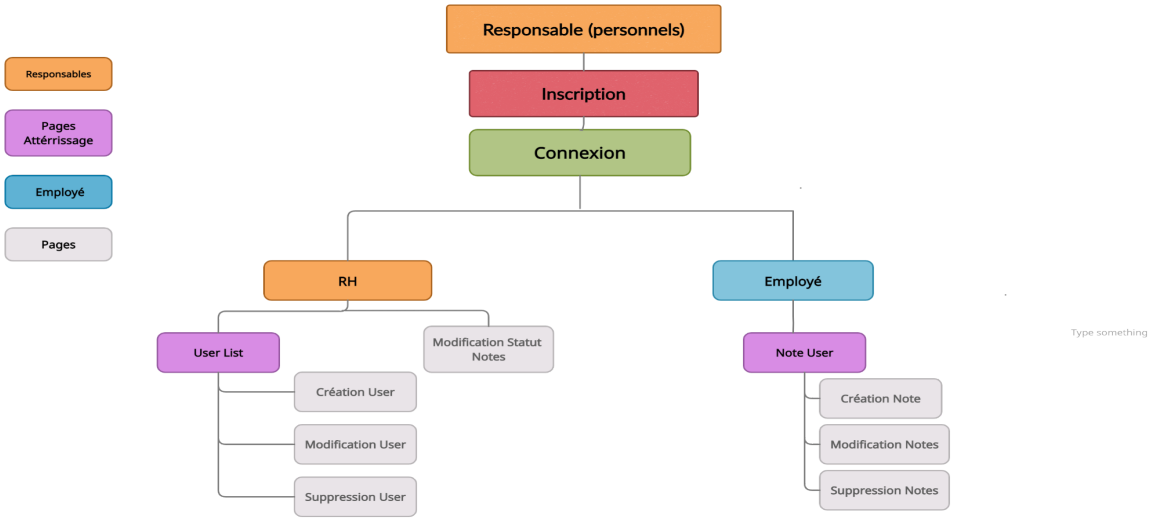
  C’est la page d'atterrissage, avec toutes les informations des notes (types, montants, date, statut), et aussi le justificatif de la facture en photo ou fichier PDF. ainsi que deux boutons pour:

  - ➢ **Modifier la note:** seulement si la note est en mode *En Cours.* On cliquant sur modifier, la page *modifier une note* s’affiche, elle contient un formulaire avec les informations de la note, il est donc possible de les modifier si besoin.
  - ➢ **Supprimer:** pour effacer une note de sa propre liste, en cliquant sur le bouton une fenêtre d'alerte s’affiche, pour confirmer ou annuler.

- **Page Ajouter une note:**

  Le salarié peut aussi accéder à cette page en cliquant sur le bouton plus, situé au début de la liste des notes. Cette page contient le formulaire de création des notes frais avec (Type, Montant, Ticket, Date, Commentaire). Et aussi la possibilité d'uploader une photo ou un fichier pdf de la facture justificative.

Après analyse des besoins dans le cahier des charges, nous avons maquetté l'arborescence des pages à afficher pour l'application, nous avons donc définit deux parties, qui s'affichent selon le statut de l'utilisateur:



5. Spécifications techniques du projet & sécurités

5.1 Spécifications techniques de l'API note

J'ai utilisé mon hébergeur **Plesk** pour héberger et gérer l'API note, il utilise Le mode de sécurité renforcé protège les données confidentielles stockées dans la base de données, et cela permet de:

- Tous les mots de passe stockés dans la base de données Plesk sont chiffrés à l'aide de la clé secrète de Plesk. Ainsi, même si un tiers obtient un dump de la base de données Plesk, vos clients ne sont pas compromis.
- Les données confidentielles (par. : les mots de passe utilisateur) peuvent être récupérées à l'aide de l'API de Plesk.
- Les mails de récupération de mots de passe ne contiennent plus le mot de passe

en texte brut. À la place, il contient un lien vers un formulaire de réinitialisation du mot de passe.

Pour sécuriser les routes de l'API, j'ai utilisé un **Token (un jeton en français).** C'est un nombre ou une chaîne de caractère aléatoire qui va être testée avant toute modification ou édition d'un article. Ce token doit être créé dans un fichier **PHP** qui sera appelé sur toutes les pages. Typiquement, il s'agit d'une functions.

À l'intérieur de mon **controller,** j'ai ajouté le fichier *"Jwt Token.php",* qui contient la fonction "Login Token()", qui va générer mon token, une fois appelé.

```php
public function LoginToken()
{
        $tokenData['uniqueId'] = '11';
        $tokenData['role'] = 'alamgir';
        $tokenData['timeStamp'] = Date('Y-m-d h:i:s');
        $jwtToken = $this->objOfJwt->GenerateToken($tokenData);
        echo json_encode(array('Token'=>$jwtToken));
}
```

Le token est généré puis appelé dans la fonction *"Connect()",* de controller, et il contient les informations de l'utilisateur (id - email - rank).

```php
    $tokenData['email'] = $user->email;
    $tokenData['rank'] = $user->rank;
    $tokenData['id'] = $user->id;

$jwtToken = $this->objOfJwt->GenerateToken($tokenData);
echo $jwtToken;
```

Après pour sécuriser chaque route de l'API, j'ai ajouté la condition qui vérifie la fiabilité de token et ses informations concernant l'utilisateur connecté.

```
$token =$this->input->get_request_header('Token');
if ($this->objOfJwt->verifyToken($token) === false) {
    return;
}
```

5.2 Spécifications techniques de l'application

J'ai développé l'application Notes de frais côté salariés avec React Native et Expo.

Le salarié une fois connecté, le token sera récupéré puis stocké, en utilisant le rank (statut) salarié, il se redirige vers le côté salarié et ses notes de frais. Pour cela j'ai utilisé les librairies suivantes:

- **Axios:** qui utilise la route de l'API de la fonction "connect()", en générant le token.
- **Secure Store** qui permet de stocker le token en toute sécurité localement sur l'appareil.
- **Navigation** pour la redirection vers la page des notes.

```
import axios from "axios";

import * as SecureStore from 'expo-secure-store';

import {withNavigation} from "react-navigation";
```

```
save = async(key, value) => {
    await SecureStore.setItemAsync(key, value);
}
```

```
axios
   .post("https://mohamed-hadjadji.students-laplateforme.io/apinote/index.php/User/connect",data)
   .then((response) => {
      axios.defaults.headers.common['Token'] = `Bearer${response.data}`; // une fois connecté, on configure axios
      this.save('token', `${response.data.substr(1)}`);
      // faire une redirection vers MainPage.js
    //    console.log('hello',this.props.navigation);
    this.props.navigation.navigate('MainPage');
   })
```

Enfin Pour utiliser les informations de l'utilisateur fournies par le token, j'ai utilisé la librairie **jwt-decode,** pour décoder le token.

```
import jwt_decode from "jwt-decode";
```

En décodant le token, j'ai récupéré l'identifiant de l'utilisateur, pour que le salarié connecté ait l'accès uniquement à ses notes de frais.

```
async componentDidMount() {
 await SecureStore.getItemAsync('token').then((val) => {
   var decoded = jwt_decode(val);
   var iduser = decoded.id;
   console.log(decoded);

 axios.get('https://mohamed-hadjadji.students-laplateforme.io/apinote/index.php/Note/index_note/'+iduser)
   .then(response => {
```

6. Réalisation de l'application

6.1 Choix des frameworks

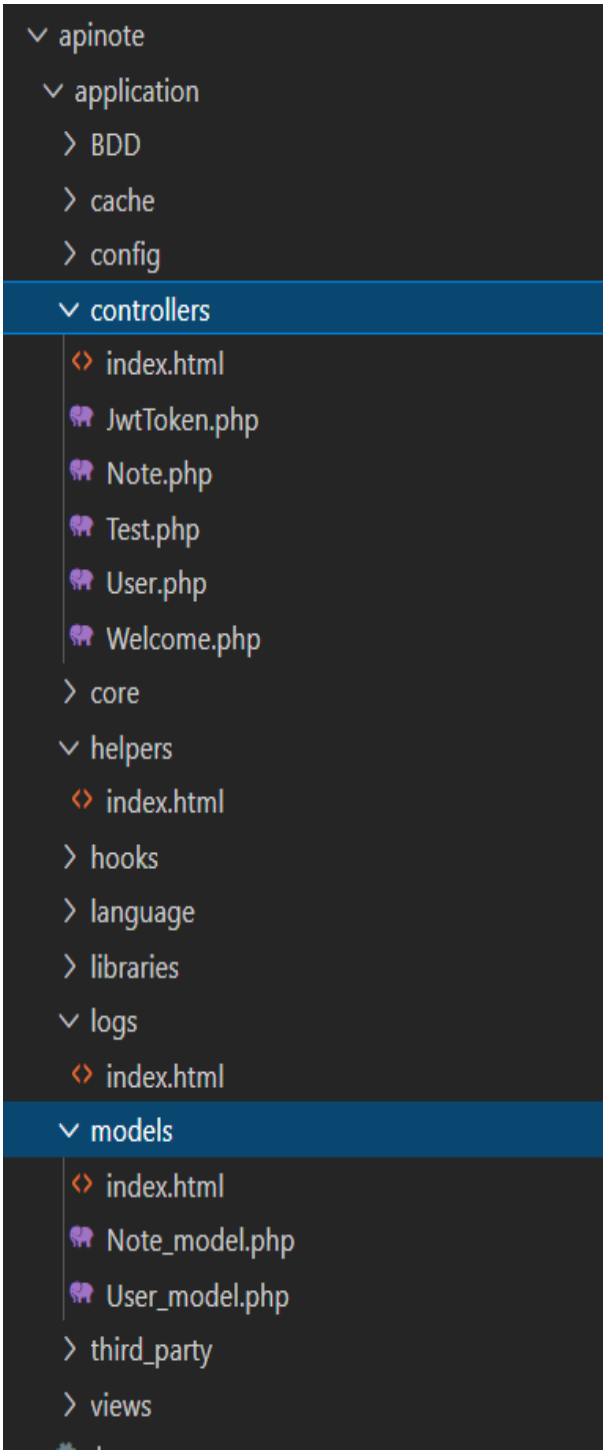
Comme j'avais précisé auparavant, dans le cahier des charges. Le projet devra être développé avec les deux frameworks suivant:

- **CodeIgniter:** pour le développement de l'API REST, avec les quatres opérations de base pour la persistance des données (CRUD). Cependant j'ai choisi la version 3 de CodeIgniter, pour la stabilité et la structure.

- **React Native:** l'avantage de React Native est que cet outil permet de mutualiser le code de manière partagée entre les deux systèmes d'exploitation (OS) iOS et Android. Il permet effectivement de développer et de déployer ensemble une application mobile sur les différentes plateformes tout en n'utilisant qu'une seule base de code.

6.2 Développement de l'API

l'API notes représente la partie back-end de projet, Pour la développer j'ai utilisé CodeIgniter 3. qui utilise la structure *MVC* **(Model - View - Controller)**. Et j'ai implémenté les quatres opérations de base pour la persistance des données (CRUD).



Dans le *Model*, j'ai créé un fichier model:

- **Note model:** qui contient les fonctions suivantes:

- *Create note()*: Pour insérer des notes de frais.
- *All note (user Id)*: Pour afficher la liste des notes de l'utilisateur connecté.
- *Get Note ($noteId)*: Pour sélectionner l'identifiant de la note.
- *Update Note ($noteId)*: Pour la modification de la note, sélectionné par son id.
- *Delete Note ($noteId):* Pour supprimer la note, sélectionné par son id.
- *Update statut Note ($note Id):* Pour modifier le statut de la note, sélectionné par son id.

Dans le Controller, j'ai ajouté un controllers:

- **Note:** qui contient les fonctions suivantes:

- *Index note ($userId)*: Qui appelle la fonction *All_note(userId)* de *Note model*, pour la récupération des données des notes de l'utilisateur connecté, en format *JSON*.
- *Get Id Note ($noteId)*: Qui appelle la fonction *Get Note ($note Id)* de *Note model*, et qui sélectionne l'id note.
- *Create note ()*: Qui appelle la fonction *Create note()* de *Note model*, pour l'insertion des données notes de frais.
- *Edit note ($noteId)*: Qui appelle la fonction *Update Note ($note Id)* de *Note model*, pour la modification de la note sélectionnée par son id.
- *Delete note ($noteId):* Qui appelle la fonction *Delete Note ($note Id)* de *Note model*, Pour supprimer la note, sélectionné par son id.
- *Edit RH note ($noteId):* Qui appelle la fonction *Update statut Note ($note Id)* de *Note model*, pour modifier le statut de la note, sélectionné par son id.

- Pour sécuriser les routes de l'API, j'ai ajouté la fonction **Login Token(),** dans le fichier **jwt token** de controller, qui contient la fonction *Login Token(),* et qui génère le *Token* une fois que l'utilisateur est connecté. Le Token est vérifié dans chaque fonction de controller. Par exemple:

```
function index_note($userId){
    $token =$this->input->get_request_header('Token');
    if ($this->objOfJwt->verifyToken($token) === false) {
        return;
    }
    $this->load->model('Note_model');

    $notes = $this->Note_model->all_note($userId);
    $data = array();
    $data['notes'] = $notes;
    echo json_encode($notes);
}
```

je n'utilise pas de View, car l'affichage se fait dans la partie front de l'application, qui utilise l'API note avec ses routes, appelant les différentes fonctions du controller, pour toutes les tâches réalisées dans l'application.

Exemple d'une fonction model

```
function all_note($userId)
{
    $this->db->where('id_user', $userId);
    return $notes = $this->db->get('expense_reports')->result_array(); //SELECT * from expense_reports
}
```

6.3 La base de données

L'application *notes de frais*, nécessité l'utilisation des données des utilisateurs et des notes pour son fonctionnement, donc elle est composée de deux tables **(Users** &

Expense Reports).

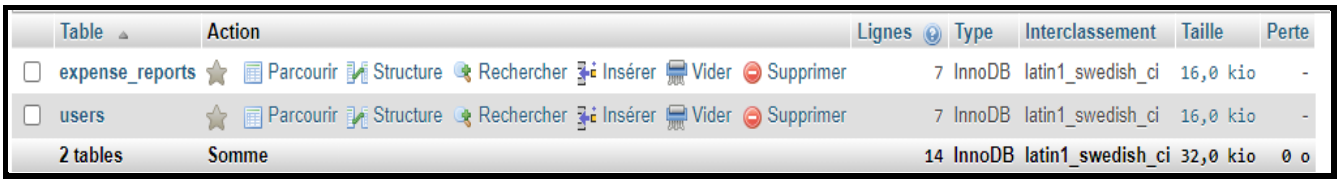


	Table	Action	Lignes	Type	Interclassement	Taille	Perte
☐	expense_reports	Parcourir Structure Rechercher Insérer Vider Supprimer	7	InnoDB	latin1_swedish_ci	16,0 kio	-
☐	users	Parcourir Structure Rechercher Insérer Vider Supprimer	7	InnoDB	latin1_swedish_ci	16,0 kio	-
	2 tables	Somme	14	InnoDB	latin1_swedish_ci	32,0 kio	0 o

Dans mon serveur Plesk, et en utilisant phpMyadmin j'ai créé la table **Expense Reports** pour stocker les informations de chaque note de frais insérée par salarié. Elle est composée de:

- Id note en int auto_increment.
- Type en varchar taille 255.
- Prix en varchar taille 65.
- Photo en varchar taille 255.
- Comment en varchar taille 255.
- Id_user en int.
- State en varchar 255.
- Date en date.

6.4 Les composants d'accès aux données

Dans le fichier *application/config/config.php* de l'API, j'ai défini l'URL de la base de données ('http://mohamed-hadjadji.students-laplateforme.io/apinote/'). Puis j'ai inséré les coordonnées de la BDD dans le fichier *application/config/database.php*.

```
'hostname' => 'localhost',
'username' => '                ',
'password' => '        ',
'database' => 'mohamed-hadjadji_ems',
'dbdriver' => 'mysqli',
```

Pour développer les composants d'accès à la base de données, j'ai utilisé la méthode **CRUD.** Le terme CRUD est étroitement lié avec la gestion des données numériques. Plus

précisément, CRUD est un acronyme des noms des quatre opérations de base de la gestion de la persistance des données et applications :

- Create (créer)
- Read ou Retrieve (lire)
- Update (mettre à jour)
- Delete ou Destroy (supprimer)

Nos utilisateurs doivent pouvoir supprimer, ajouter, mettre à jour des données sur l'application

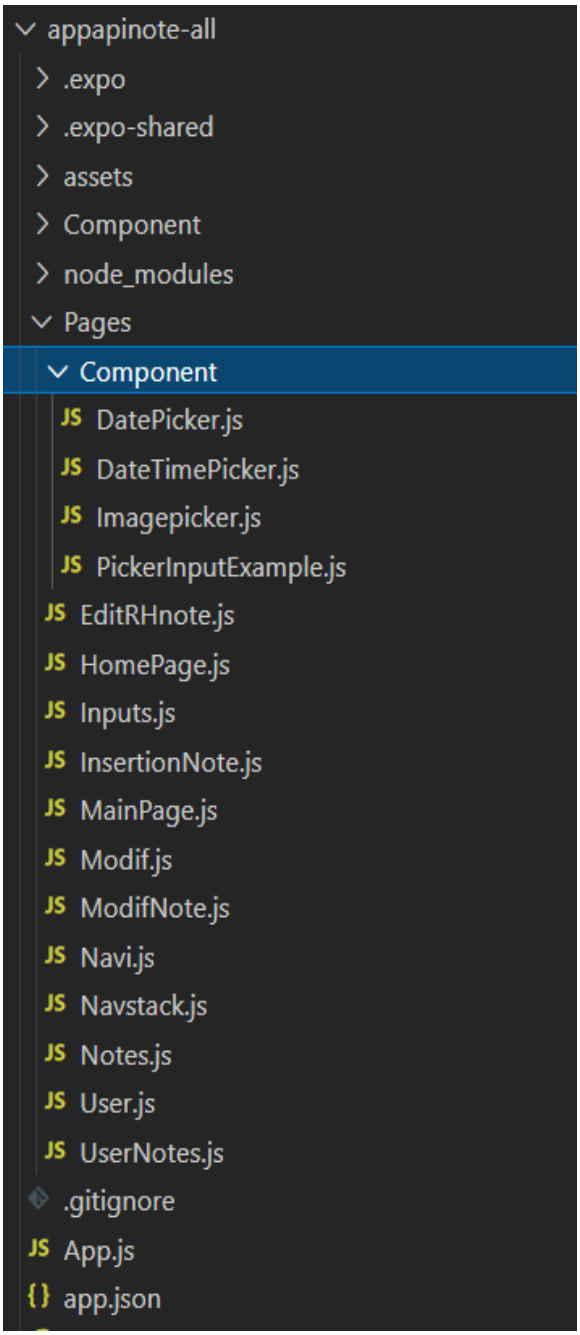
CRUD-Opération	SQL	RESTful HTTP	Description
Create	INSERT	POST	Création des notes.
Read	SELECT	GET	Affichage des notes.
Update	UPDATE	PUT	Modification des notes.
Delete	DELETE	DELETE	Supprimer des notes.

6.5 Développement de l'application

J'ai utilisé **Expo CLI**, qui est un ensemble d'outils construits autour de React Native et, bien qu'il possède de nombreuses fonctionnalités , la fonctionnalité la plus pertinente pour nous en ce moment est qu'il peut nous permettre d'écrire une application React Native en quelques minutes.

j'ai donc utilisé **npm** pour installer *Expo* avec la commande `npm install -g expo-cli`. puis `expo start` pour le démarrer.

j'ai commencé par modifier le fichier **App.js,** qui est le point de départ de l'application, il contient un componente standard prédéfinis du React native (**App)**. Puis j'ai ajouté les fichier suivant:



- **Navi.js:** Pour La navigation entre les pages de côté salarié, dans le fichier j'ai créé deux fonctions en utilisant les librairies :

```
'@react-navigation/native';
'@react-navigation/bottom-tabs';
"@react-navigation/stack";
```

```
export function Navi_salarie() {
```

pour naviguer entre les deux pages principales (Notes - Ajouter une note), avec le système **Tab.Navigator.**

```
export function AppBout()
```

Pour naviguer avec les boutons. en utilisant le système **Stack.Navigator.**

- **HomePage.js:** contient le componente

```
class HomePage extends Component
```

qui utilise les librairies:

```
TextInput - axios - SecureStore - Navigation
```

Pour afficher le formulaire de connexion. En utilisant Axios avec la route connect de l'API, je récupère le token et je le stock dans local storage. Puis Navigate s'est dirigé vers (MainPage).

- **MainPage.js:** contient le componente

```
const MainPage = (props) => {
```

En récupérant le token a partir de localstorage, et en utilisant `jwt_decode` j'ai décodé le token puis j'ai utilisé le **rank (salarié),** pour faire la redirection vers la page d'accueil qui est le componente **Note Liste** de **Notes.js.**

- **Notes.js:** c'est la page d'atterrissage pour le salarié, après la connexion. Elle contient la liste de ses notes de frais avec les différents statuts (En cours - Validé - Refusé).

J'ai utilisé ***Axios.get*** pour se connecter avec la fonction `index_note($userId)`, de l'API qui renvoie les informations des notes de l'utilisateur concerné, en récupérant son **Id** à partir de ***Token*** grâce à la librairie `jwt_decode`.

```
async componentDidMount() {
 await SecureStore.getItemAsync('token').then((val) => {
   var decoded = jwt_decode(val);
   var iduser = decoded.id;
   console.log(decoded);

 axios.get('https://mohamed-hadjadji.students-laplateforme.io/apinote/index.php/Note/index_note/'+iduser)
   .then(response => {

     this.setState(prevState => ({
         NoteList: prevState.NoteList = response.data
```

Ensuite j'ai utilisé `<FlatList>` avec `renderItem= {({ item })` pour afficher chaque information des notes envoyer par l'API en JSON. Par exemple `item.date` pour la date.

Pour la sécurité, le salarié n'a pas le droit de modifier sa note de frais une fois, le RH a validé ou refusé la note. Donc j'ai ajouté deux conditions par rapport au statut de la note:

```
- if (item.state == 'En cours')
```

Afficher la note avec le bouton de modification.

```
- else
```

Autrement (Validé ou Refusé), n'afficher que la note .

- **Insertion Notes.js:** qui affiche la page de création des notes de frais, avec un formulaire d'insertion.

Pour afficher ce formulaire, j'ai utilisé `TextInput` pour afficher les inputs, pour la date et la sélection de types, j'ai utilisé les librairies:

```
import DatePicker from 'react-native-datepicker'
import { Picker} from 'native-base';
```

Puis j'ai utilisé ***Axios.post*** qui fait appel à la fonction `create_note()` de l'API. Et `let bodyFormData = new FormData();` pour l'insertion des données de la note.

```
bodyFormData.append('Type',Type)
bodyFormData.append('price',price)
bodyFormData.append('comment',comment)
bodyFormData.append('id_user',iduser)
bodyFormData.append('date',date)
```

```
axios
  .post("https://mohamed-hadjadji.students-laplateforme.io/apinote/index.php/Note/create_note", bodyFormData, {
    headers: {
      'Content-Type': 'multipart/form-data'
    }
  })
  .then((response) => {
```

- **Modif notes.js:** c'est la page de modification de la note avec le formulaire de modification.