



# DOSSIER PROFESSIONNEL (DP)

*Nom de naissance* ➤ **OUARI**  
*Nom d'usage* ➤ **-**  
*Prénom* ➤ **Salim**  
*Adresse* ➤ **5 rue des frégates , Le Monticole  
13015 Marseille**

## Titre professionnel visé

**Concepteur(trice) Développeur(se) Informatique**

### MODALITÉ D'ACCÈS :

- ☒ Parcours de formation
- ☐ Validation des Acquis de l'Expérience (VAE)

## Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.  
**Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

### Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel (DP)** dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

*[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]*

### Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

*Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.*



## Sommaire

### Exemples de pratique professionnelle

#### Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

p.

5

- TissApp- Maquetter une application

p.

p.

- TissApp

p.

p.

- Le jeu du Sokoban - Développer une interface utilisateur de type desktop

p

p.

#### Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

p.

16

- TissApp

p.

p.

- Portfolio

p.

p.

- Intitulé de l'exemple n° 3

p

p.

#### Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

p.

- TissApp

p.

p.

- Intitulé de l'exemple n° 2

p.

p.

- Intitulé de l'exemple n° 3

p

p.

Titres, diplômes, CQP, attestations de formation *(facultatif)*

p.

Déclaration sur l'honneur

p.

Documents illustrant la pratique professionnelle *(facultatif)*

p.

Annexes *(Si le RC le prévoit)*

p.

# **EXEMPLES DE PRATIQUE PROFESSIONNELLE**

## Activité-type 1 Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n°1 - TissApp - Maquetter une application

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Lors de ma formation à La Plateforme, j'ai développé une application mobile de chat en React Native en équipe .

L'objectif était de créer une plateforme simple et conviviale permettant aux gens de se connecter et communiquer entre eux. On n'avais pas de public cible spécifique en tête et on voulait que l'application soit ouverte et accessible à tous, offrant un espace de détente et de conversation.

Nous avons créé une application mobile accompagnée d'un panel admin.

À la suite de la rédaction du cahier des charges, j'ai réalisé dans un premier temps des maquettes basse fidélité puis haute fidélité sur **Figma**.

Avant de me lancer dans la réalisation des maquettes nous avons tout d'abord décidé d'une charte graphique.

L'outil **Figma m'a permis de travailler de manière collaborative** avec mes coéquipiers, ce qui a facilité l'échange de nos idées.

j'ai également utilisé **trrello** pour collaborer à la gestion et à l'organisation du projet, ce qui m'a permis de séparer les tâches à faire selon leurs priorités et leurs difficultés ce qui permet une meilleure organisation et une visibilité du client sur l'avancée du projet.

Grâce à cela j'ai pu apprendre à maquetter une application et à collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement.

Ce projet valide les compétences :

- **Maquetter une application**
- **Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement**

### 2. Précisez les moyens utilisés :

J'ai utilisé :

- **Figma** : maquettes
- **Trello** : gestion de projet

# DOSSIER PROFESSIONNEL (DP)

## 3. Avec qui avez-vous travaillé ?

Pour ce projet j'ai travaillé avec Tchessi PRE et Samir MOKADDEM

## 4. Contexte

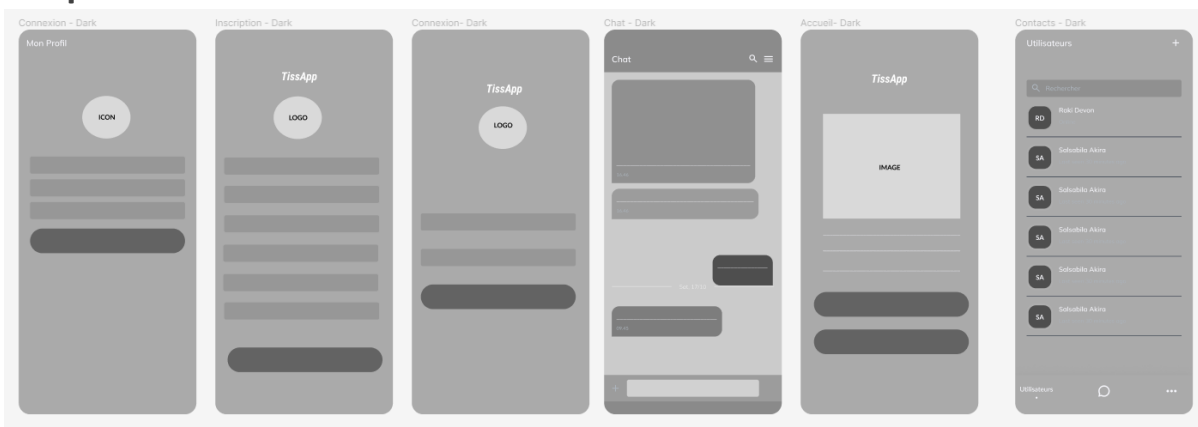
Nom de l'entreprise, organisme ou association ► La Plateforme

Chantier, atelier, service ► TissApp

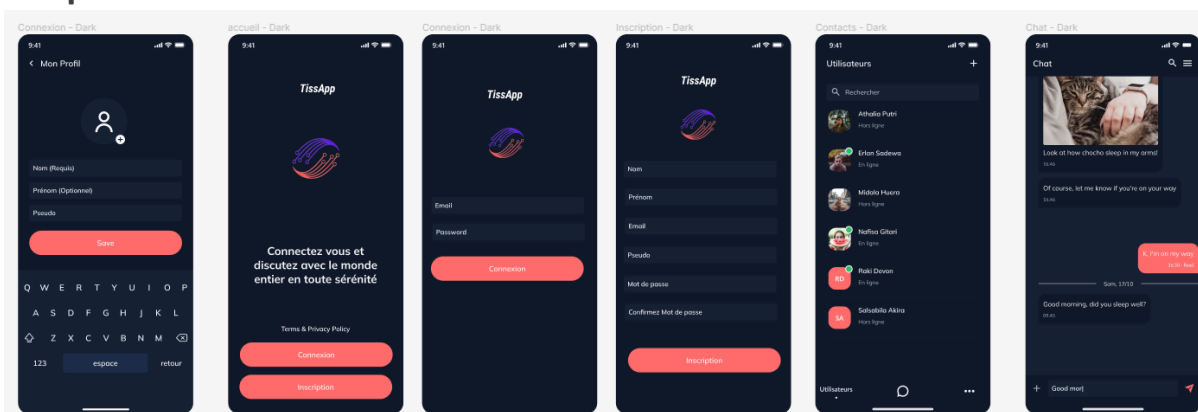
Période d'exercice ► Du : 02/01/2023 au : 31/01/2023

## 5. Informations complémentaires (facultatif)

### Maquette basse fidélité :



### Maquette haute fidélité :



## Activité-type 1

### Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n° 2 - Le jeu du Sokoban - Développer une interface utilisateur de type desktop

#### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Ce projet valide les compétences :

- **Développer une interface utilisateur de type desktop**

Dans le cadre de ma formation, j'ai développé le jeu du Sokoban en Python.

Pour cela, j'ai utilisé la librairie Pygames, qui me permet de créer des éléments graphiques ainsi que d'écouter sur des événements bien précis comme sur celui des touches du clavier.

Ce projet a été codé en Orienté Objet pour pouvoir au mieux faire interagir mes éléments et une plus grande lisibilité du code.

J'ai commencé par initier le jeu avec la fonction init de Pygame et définir la surface sur laquelle on va jouer. J'ai rajouté une fonction drawMap et drawPlayer pour dessiner le fond de mon interface afin d'être sûr de la position de mes murs et de mon joueur.

```
# initialisation de Pygame
pygame.init()

# création de la fenêtre
screen = pygame.display.set_mode((LARGEUR, HAUTEUR))
pygame.display.set_caption(TITRE)

# chargement de l'image de fond
background = pygame.image.load("images/back.jpg")

# création de la grille et du joueur
_grille = Grille("levels/lv1.txt")
_grille.drawMap(screen)
_player = Player(_grille)
_player.drawPlayer(screen)

# affichage de la fenêtre
pygame.display.flip()
```

Ensuite, j'ai créé une classe Player pour gérer les différents événements de mon player.

```
class Player:
    def __init__(self, grille):
        pygame.mixer.music.load("sounds/mvrasseli_play_the_game.mp3")
        pygame.mixer.music.set_volume(0.5) # volume réglé à 50%
        pygame.mixer.music.play()

        self.gauche = pygame.image.load("images/mario_gauche.gif")
        self.droite = pygame.image.load("images/mario_droite.gif")
        self.bas = pygame.image.load("images/mario_bas.gif")
        self.haut = pygame.image.load("images/mario_haut.gif")

        self.position = self.droite

        self.grille = grille
        self.pos = self.grille.getPlayerPosition(self.grille)

        self.x = self.pos[0] // SIZE
        self.y = self.pos[1] // SIZE
        self.id_joueur = None
        self.score = 0 # initialisation du score à 0
```

Il ne me reste plus qu'à lancer une boucle qui mettra le jeu à jour avec les nouvelles données à chaque action possible dans le jeu selon la touche pressée que j'aurai au préalable configuré.



```
# boucle principale du jeu
continuer = True
while not _grille.is_fini():
    for event in pygame.event.get():
        if event.type == QUIT:
            # sauvegarde du score dans la base de données avant de quitter
            _grille.set_niveau(1) # définir le niveau atteint
            _player.set_id_joueur("joueur1") # définir l'identifiant du joueur
            query = "INSERT INTO scores (id_joueur, niveau, score) VALUES (%s, %s, %s)"
            params = (_player.id_joueur, _grille.niveau, _player.get_score())
            c.execute(query, params)
            conn.commit()
            sys.exit()
        if event.type == KEYDOWN:
            _player.move(event.key)
            # incrémente le score à chaque mouvement du joueur
            _player.set_score(_player.get_score() + 1)
            if event.key == K_r:
                # régénération de la grille et du joueur
                _grille = Grille("levels/lv1")
                _grille.drawMap(screen)
                _player = Player(_grille)
                _player.drawPlayer(screen)
```



# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 2. Précisez les moyens utilisés :

J'ai utilisé :

- **VScode**
- **Python** comme langage
- **Pip** comme gestionnaire de package
- les librairies : **pygame** , **random** et **sys**

## 3. Avec qui avez-vous travaillé ?

J'ai fait ce projet seul

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ *LaPlateforme*

Chantier, atelier, service ▶ *MySokoban*

Période d'exercice ▶ Du : *07/03/2023* au : *20/03/2023*

## 5. Informations complémentaires *(facultatif)*

## Activité-type 1 Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n° 1 - TissApp

---

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre du projet pour l'association le Sel de La Vie, nous avons développé un panel Admin web en **Nuxt.js** .

Ce projet valide les compétences :

- **Développer des composants d'accès aux données**
- **Développer la partie front-end d'une interface utilisateur**
- **Développer la partie back-end d'une interface utilisateur**

Cette interface contient un accès à un **CRUD** pour gérer les utilisateurs et le chat avec les messages.

J'ai choisi d'utiliser Nuxt.js pour plusieurs raisons. Tout d'abord, Nuxt.js est basé sur Vue.js, un framework JavaScript populaire et très performant. Vue.js offre une approche basée sur les composants, ce qui facilite la création d'interfaces utilisateur réactives et modulaires. En utilisant Nuxt.js, j'ai pu bénéficier de la puissance de Vue.js tout en ajoutant des fonctionnalités spécifiques au développement d'applications web.

Ensuite, Nuxt.js fournit une architecture solide pour le développement d'applications universelles. Il prend en charge le rendu côté serveur (SSR) et le pré-rendu statique, ce qui permet d'améliorer considérablement les performances de l'application. Avec Nuxt.js, j'ai pu créer des applications qui se chargent rapidement, offrant une expérience utilisateur fluide, notamment en optimisant le temps de chargement initial et en améliorant le référencement des pages.


J'ai créé une page d'administration des utilisateurs qui permet aux administrateurs de visualiser et de gérer les profils des utilisateurs de TissApp. En utilisant les composants de Nuxt.js, j'ai créé une interface utilisateur efficace qui récupère la liste de tous les utilisateurs enregistrés dans la base de données. Les administrateurs peuvent facilement visualiser les informations des utilisateurs, notamment leur email, leur prénom et leur mot de passe.

Rechercher un utilisateur

Entrer un nom d'utilisateur...

Nombre d'utilisateurs total : 2

User Panel

ID	Avatar	Nom d'utilisateur	Email	Rôle	Date de création	Status	Actions
1		Salim Ouari	salim.ouari@laplateforme.io	Administrateur	mardi 14 février, 16:26:10	En-ligne ●	
2		John Doe	raulgonz@hotmail.fr	Utilisateur	vendredi 12 mai, 18:46:21	Hors-ligne ●	<button>Modifier</button> <button>Supprimer</button>

## Explication de la logique DELETE Admin = true

J'ai créé une fonction appelée `deleteUser(userId)` qui me permet de supprimer un utilisateur en envoyant une requête DELETE.

Lorsque j'appelle cette fonction, j'envoie une demande à l'URL **`http://localhost:3100/api/auth/users-delete/${userId}`** en utilisant la méthode DELETE. Pour cela, j'utilise l'objet `fetch` qui me permet d'effectuer des requêtes HTTP. Cette fonction prend en paramètre l'ID de l'utilisateur que je souhaite supprimer. Ensuite, j'attends la réponse de la requête en utilisant `await data.json()`, ce qui me permet d'obtenir les données renvoyées par le serveur au format JSON.

```
// REQUEST DELETE USER samir-mokaddem, 3 months ago • add sweet alert ...
async deleteUser(userId) {
  try {
    const data = await fetch(`http://localhost:3100/api/users/${userId}`, {
      method: 'DELETE',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${localStorage.getItem('token')}`
      }
    })
    const response = await data.json();
    console.log("success delete user");
    console.log(response);
    this.getUsers();
  } catch (error) {
    console.log("catch delete user");
    console.log(error);
    this.errorMessage = error.message;
  }
},
```

Une fois que j'ai les données de la réponse, je les affiche dans la console en utilisant `console.log(response)`. Si la suppression de l'utilisateur réussit, j'affiche un message de succès dans la console avec `console.log("success delete user")`. Ensuite, j'appelle la fonction **getUsers()** pour mettre à jour la liste des utilisateurs après la suppression. En cas d'erreur lors de la requête, j'affiche un message d'erreur dans la console avec `console.log(error)`.

De plus, j'assigne le message d'erreur à la variable **"errorMessage"** pour pouvoir l'afficher à l'utilisateur.

```
// ROUTE ADMIN

exports.deleteUserAccount = async (req, res, next) => {
  try {
    const user = req.user.admin
      ? await User.findOne({ where: { id: req.params.id } })
      : req.user;
    await user.softDestroy();
    res.status(200).json({ message: 'Compte supprimé' });
  } catch (error) {
    res.status(400).json({ error });
  }
};
```

Dans cette route j'utilise le controller `deleteUserAccount` qui gère la suppression d'un compte utilisateur. Lorsqu'une requête DELETE est effectuée, cette fonction est appelée. Si l'utilisateur est un administrateur, elle recherche l'utilisateur cible en utilisant l'ID fourni dans les paramètres de la requête. Ensuite, elle utilise la méthode **softDestroy()** pour marquer le compte comme supprimé dans la base de données. Si la suppression réussit, une réponse avec le **statut 200** et un message indiquant que le compte a été supprimé est renvoyée. En cas d'erreur, une réponse avec le **statut 400** et les détails de l'erreur est renvoyée.

En résumé, ce contrôleur permet de supprimer des comptes utilisateurs en fonction des permissions de l'utilisateur qui effectue la demande.

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 2. Précisez les moyens utilisés :

J'ai utilisé :

- **VScode**
- **Nuxt js**
- **Node js**
- **sequelize**
- les librairies **react** utilisées : **Jwt-decode**, **fetch...**

## 3. Avec qui avez-vous travaillé ?

J'ai collaboré avec **Samir MOKADDEM** et **Tchèssi PRE** durant ce projet.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*

Chantier, atelier, service ▶ *TissApp*

Période d'exercice ▶ Du : *02/01/2023* au : *31/01/2023*

## 5. Informations complémentaires (facultatif)

## Activité-type 2 Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 - TissApp

---

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

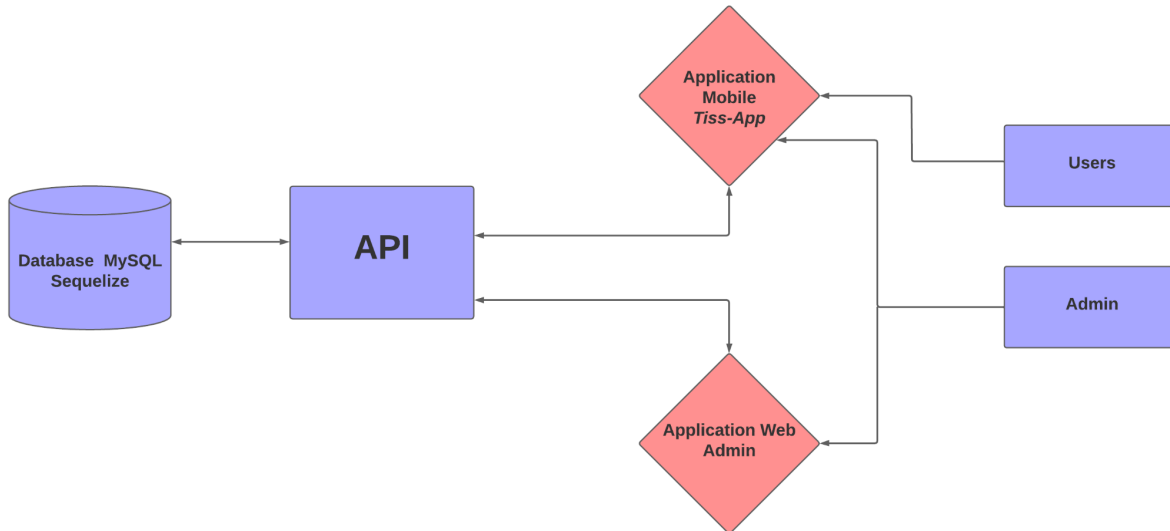
Après avoir pris en note dans le cahier des charges des besoins de l'application (voir exemple 1), il a été décidé de réaliser une application en **React Native** qui serait composée :

- d'une page d'accueil
- d'une page d'inscription
- d'une page de connexion
- d'une page profil
- d'une page d'accueil
- d'une page chat
- d'une page contact
- 

Ce projet me permet de valider les compétences :

- **Développer une application mobile**
- **Développer des composants métier**
- **Concevoir une application**
- **Construire une application organisée en couches**

Nous avons architecturé l'application mobile comme représenté dans le schéma suivant :

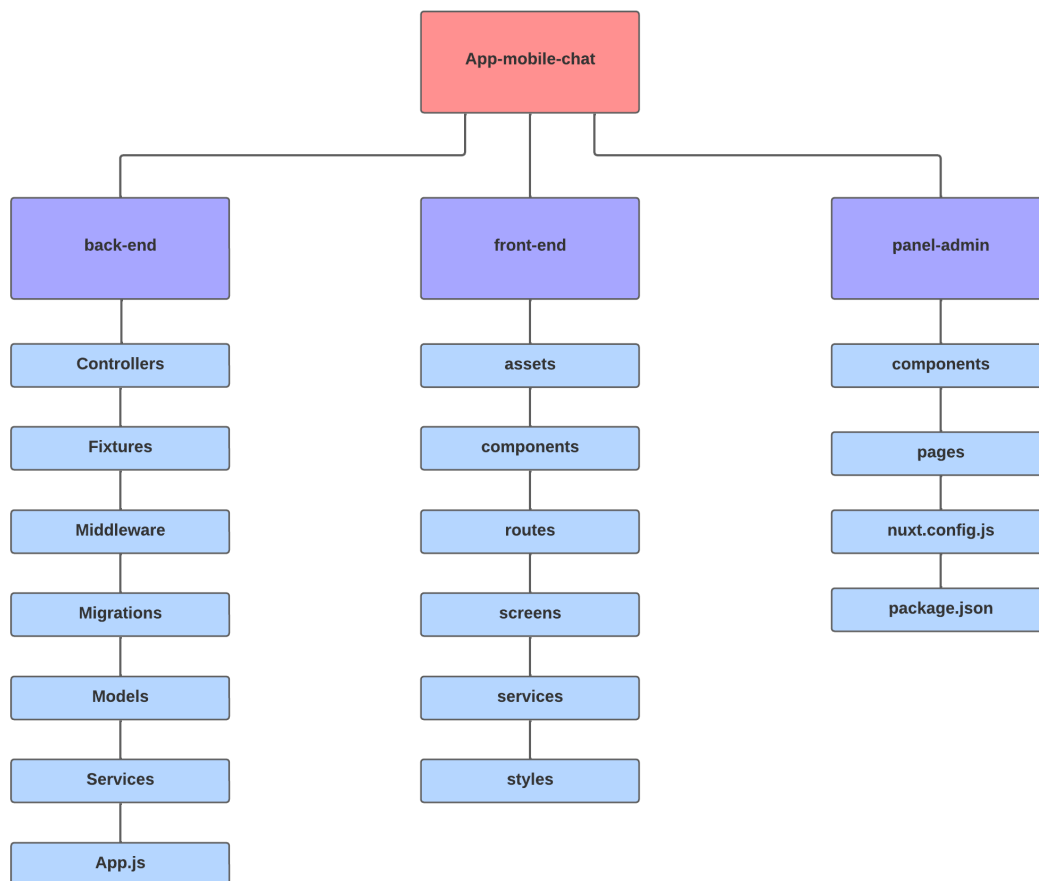


On ne peut pas accéder au reste de l'application si l'on n'est pas connecté.

Nous avons utilisé Expo, qui est à la fois un framework et une plateforme qui simplifient la création et le déploiement d'applications mobiles avec React Native. Expo embarque de nombreux outils utiles et des librairies natives pour React Native. Il gère aussi la mise à jour de ces librairies.



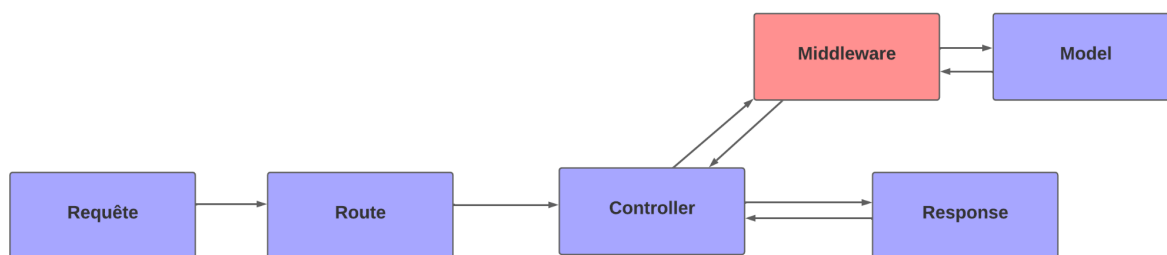
## Arborescence du projet



## Fonctionnement de l'API

Lorsque le client envoie une requête sur mon API, le routeur analyse l'URL, et, en fonction de la route et de la méthode, un Controller est appelé. Ce contrôleur/Controller va faire appel à un service qui va communiquer avec le modèle afin de récupérer des données. Ensuite, ses données sont analysées par le service puis une réponse est envoyée au format JSON avec un code statut.

### Architecture de l'API



## Fonctionnement des routes

Lorsqu'une application Express reçoit une requête HTTP, elle crée deux objets : "req" contenant les informations de la requête, et "res" contenant les méthodes pour renvoyer une réponse. "req" est utilisé pour accéder aux données de la requête, telles que les paramètres d'URL ou les données du corps. "res" est utilisé pour envoyer une réponse, telle qu'une page HTML ou un objet JSON. Si un middleware est utilisé, il peut être appelé en utilisant la fonction "next", pour passer la requête au middleware suivant ou à la prochaine route correspondante.

```
var express = require('express');
var app = express();
```

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

```
app.get('/', function(req, res, next) {
  next();
});
```

Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function, called "res" by convention.

HTTP request argument to the middleware function, called "req" by convention.

```
app.listen(3000);
```

j'ai créé une fonction middleware qui est chargée de sécuriser certaines routes de l'application en vérifiant la validité du jeton d'authentification.

Tout d'abord, j'ai importé les modules nécessaires pour mettre en œuvre cette fonctionnalité. J'ai utilisé la bibliothèque **"jsonwebtoken"** pour générer et vérifier les jetons d'authentification, et j'ai importé le modèle "User" de la base de données sequelize.

Ensuite, j'ai créé la fonction middleware elle-même. Cette fonction middleware est appelée chaque fois qu'une requête est envoyée à l'application, et elle vérifie si le jeton d'authentification est valide. Si le jeton est valide, la fonction middleware appelle la fonction **"next"** pour passer la main au middleware suivant, sinon elle renvoie une erreur d'authentification.

Pour vérifier la validité du jeton d'authentification, j'ai commencé par extraire le jeton de la requête HTTP à partir de l'en-tête **"Authorization"**. J'ai ensuite utilisé la méthode "verify" de la bibliothèque "jsonwebtoken" pour décoder le jeton et obtenir l'ID de l'utilisateur.

Ensuite, j'ai vérifié si l'ID de l'utilisateur obtenu à partir du jeton correspond à l'ID de l'utilisateur qui a envoyé la requête. Si les ID ne correspondent pas, j'ai renvoyé une erreur d'authentification. Sinon, j'ai utilisé le modèle "User" pour récupérer l'utilisateur correspondant à l'ID, et j'ai ajouté cet utilisateur à l'objet de requête ("req.user") pour qu'il soit accessible aux middlewares suivants.

Enfin, si une erreur se produit à tout moment lors de la vérification du jeton ou de la récupération de l'utilisateur, j'ai renvoyé une erreur d'authentification avec un message personnalisé.

En somme, cette fonction middleware est un moyen simple mais efficace de sécuriser les routes de notre application en vérifiant l'authentification de l'utilisateur à l'aide d'un jeton d'authentification.

```
const db = require('../models');
const jwt = require('jsonwebtoken');
const { User } = db.sequelize.models;

module.exports = (req, res, next) => {
  try {
    const token = req.headers.authorization.split(' ')[1]; //r cup ration du token depuis le header
    Authorization
    const decodedToken = jwt.verify(token, 'RANDOM_TOKEN_SECRET');
    const userId = decodedToken.userId;
    if (req.body.userId && req.body.userId !== userId) {
      throw 'User ID non valable !';
    } else {
      User.findOne({ where: { id: userId }
    }).then((user) => {
      req.user = user;
      next();
    });
  }
  catch (error) {
    res.status(401).json({
      error: new Error('Requ te non authentifi e !'),
    });
  }
};
```

## Les Controllers

Les controllers sont créés dans le dossier controllers de mon application, qui contenait toutes les logiques pour ce contrôleur spécifique. J'ai également exporté la fonction du contrôleur en utilisant **module.exports**.

Pour placer le contrôleur dans mon application Node.js, j'ai créé une route correspondante dans mon fichier de routes, qui faisait appel à la fonction du contrôleur.

**Exemple :** Si une erreur se produit lors de l'appel aux méthodes Sequelize, la fonction "catch" l'erreur et renvoie une réponse JSON avec un code d'erreur 500 et le message d'erreur dans l'objet JSON.

En gérant les erreurs de cette manière, j'ai pu fournir des réponses claires et informatives aux utilisateurs de mon application, tout en assurant que les erreurs étaient correctement gérées et que mon application restait stable.

```
exports.login = async (req, res, next) => {
  try {
    const response = await
    User.authenticate(req.body.email, req.body.password);

    if (response.valid) {
      // Logique de isOnline lors d'une connexion celui
      // passe en true, (lors d'une deconnexion penser à rajouter
      // la methode PUT isOnline False)
      await User.update({ isOnline: true }, { where: {
        id: response.user.id } });
      const updatedUser = await
      User.findByPk(response.user.id);
      res.status(201).json(newToken(response.user));
    } else {
      res.status(401).json({ error: response.message });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: error.message });
  }
};
```

### Mise en place d'un component React Native

L'utilité d'un composant React Native est de permettre la création de fonctionnalités et d'interfaces réutilisables dans une application mobile, favorisant ainsi la modularité, la simplicité du code et l'accélération du développement.

Le component que j'ai créé permet de gérer l'insertion de messages et leur envoi. Ce composant encapsule la création du message et de l'image ainsi que le traitement de l'envoi du message.

Je pourrai maintenant réutiliser ce composant à plusieurs endroits de mon application, ce qui me permettra de gagner du temps et de maintenir une cohérence dans l'expérience utilisateur.

Dans le component ci-dessus, j'importe plusieurs packages que j'ai installé à l'aide de la commande npm install.

```
import React, { useState, useEffect } from 'react';
import { Image, View, TouchableOpacity, Text, StyleSheet, Modal } from 'react-native';
import { AntDesign } from '@expo/vector-icons';
import * as ImagePicker from 'expo-image-picker';
import * as Permissions from 'expo-permissions';
import jwt_decode from 'jwt-decode';
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';
import BaseUrl from '../services/BaseUrl';
const API_URL = BaseUrl
```

J'exporte ensuite ma logique et j'utilise les hooks useState pour gérer les états locaux des composants. Ils me permettent de déclarer des variables d'état et de les mettre à jour de manière réactive.

L'utilisation de useState est bénéfique car cela me permet de suivre et de gérer facilement les changements d'état dans mes composants. Je peux initialiser une variable d'état avec une valeur par défaut et utiliser la fonction de mise à jour associée pour modifier cette valeur ultérieurement.

Cela est particulièrement utile dans le contexte de ce projet car il me permet de maintenir et de synchroniser l'état des différentes parties de l'application.

Par exemple, je peux utiliser `useState` pour stocker l'image sélectionnée par l'utilisateur, le nouveau message saisi, ou encore pour contrôler la visibilité d'un élément comme une modal. Grâce à leur utilisation, je peux rendre mon application réactive en mettant à jour dynamiquement les valeurs des variables d'état. Cela facilite également la communication entre les différents composants, car je peux passer ces variables d'application.

Une fois mes variables d'état récupérées, je crée ma fonction avec ma requête Axios

```
const response = await axios.post(`${API_URL}/api/posts`, postMessage, {
  headers: {
    'Content-Type': 'multipart/form-data',
    'Authorization': `Bearer ${token}`,
  },
});

if (response.status === 201) {
  setNewMessage('');
  setPostMessageSuccess("Message envoyé avec succès");
  removePicture();
} else {
  console.log("error posting message");
  setPostMessageError("Erreur lors de l'envoi du message");
}
} catch (error) {
  console.log(error);
  setPostMessageError("Erreur network lors de l'envoi du message");
} finally {
  setIsSending(false);
}
```

Cette requête permet donc d'envoyer mon message et d'enregistrer une image s'il y a une image enregistrée dans ma variable d'état image.

```
useEffect(() => {
  if (postImageError !== '' || postMessageSuccess !== '' || postMessageError !== '') {
    setTimeout(() => {
      setPostImageError('');
      setPostMessageSuccess('');
      setPostMessageError('');
    }, 2000);
  }
}, [postImageError, postMessageSuccess, postMessageError]);
```

Une fois ma requête envoyée, j'utilise aussi un `useEffect`. Dans mon composant il me permet d'exécuter du code dans mes composants fonctionnels en réponse à des changements ou événements spécifiques. Cela rend mes composants plus réactifs et flexibles. Lorsque j'ai utilisé `useEffect` dans mon code, c'était pour gérer l'affichage des messages d'erreurs et de succès pendant une courte période. Je l'ai configuré pour surveiller les variables d'état `postImageError`, `postMessageSuccess` et `postMessageError`.

```
return (
  <View style={PostStyle.postContainer}>
    <View>
      { /* Input & Button views */ }
      {postMessageError !== '' && <Text style={PostStyle.errorText}>{postMessageError}</Text>}
      {postMessageSuccess !== '' && <Text style={PostStyle.SucessText}>{postMessageSuccess}</Text>}
    </View>
    { /* BTN UPLOAD IMAGE */ }
    <View style={PostStyle.inputContainer}>
      {!image ? (
        <TouchableOpacity onPress={() => setModalVisible(true)} style={PostStyle.selectImageButton}>
          <Ionicons name="add-outline" size={24} color="white" />
        </TouchableOpacity>
      ) : null}
    </View>
  </View>
);
```

Une fois que j'ai terminé la logique de mon composant, je peux procéder à la création de l'affichage.

Dans la partie `return` de mon composant, je peux commencer à créer mes éléments en utilisant les balises correspondantes à ceux que je souhaite afficher.

Par exemple, je peux utiliser la balise `<Text>` pour afficher du texte, la balise `<Image>` pour afficher une image, la balise `<View>` pour créer des conteneurs, et ainsi de suite.

En résumé, une fois que mon composant est fonctionnel et terminé, je peux le réutiliser à plusieurs reprises dans mon application, ce qui améliore la cohérence et l'efficacité de l'expérience utilisateur.



# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 2. Précisez les moyens utilisés :

- **React Native**
- **Node js**
- **sequelize**
- **Expo**
- les librairies react utilisées : Jwt-decode, axios

## 3. Avec qui avez-vous travaillé ?

J'ai collaboré avec Samir MOKADDEM et Tchèssi PRE durant ce projet.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme*

Chantier, atelier, service ▶ *TissApp*

Période d'exercice ▶ Du : *02/01/2023* au : *31/01/2023*

## 5. Informations complémentaires (facultatif)

## Activité-type 3 Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Exemple n°1 - TissApp

---

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour ce projet, nous avons créé une base de données pour répondre à la demande du client.

Ce projet valide les compétences :

- Concevoir une base de données
- Mettre en place une base de données
- Développer des composants dans le langage d'une base de données

Sur cette base de données, on va pouvoir enregistrer les différentes informations liées :

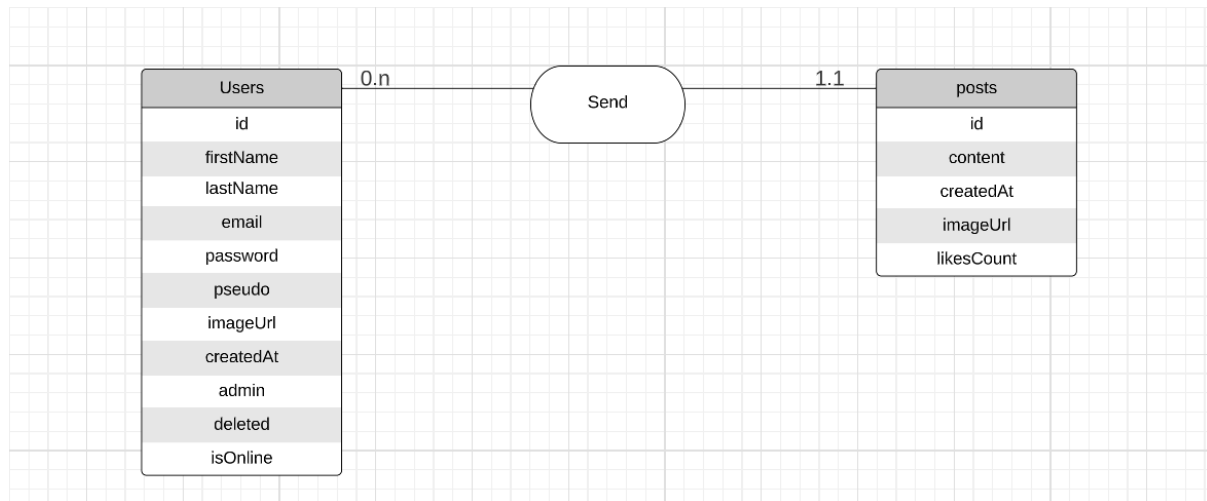
- à l'utilisateur
- aux messages du chat

On a utilisé node Js ainsi que la librairie sequelize qui prend en charge MySQL et offre une prise en charge solide des transactions et des relations. Sequelize est un ORM ("Object Relational Mapping") qui sert à mettre à disposition des classes objet permettant de manipuler les bases de données relationnelles.

Nous nous sommes appuyés sur la méthode "**Merise**" pour concevoir notre base de données SQL.

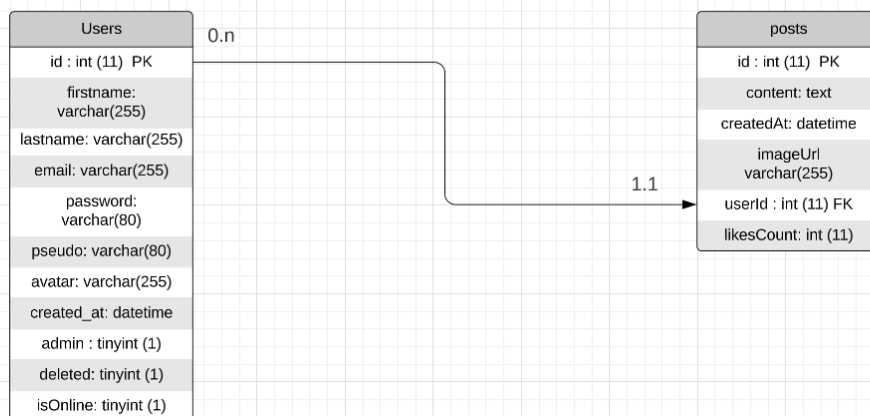
On a commencé par réaliser sur **LucidCharts**, le **MCD** (modèle conceptuel de données) et enfin le **MLD** (modèle logique de données) comme on peut le voir sur l'exemple ci-dessous.

## MCD



## MLD

Diagram de MLD Chat



La création de projet est réalisée avec Sequelize pour gérer une base de données MySQL. Après avoir configuré Sequelize dans mon projet et créé mes modèles, j'ai voulu créer ma base de données en utilisant la commande **`npx sequelize-cli db:create`**. Cette commande a créé une nouvelle base de données avec le nom que j'avais spécifié dans ma configuration Sequelize.

Pour générer une migration depuis un model **`npx sequelize-cli migration:generate --name create-table`**

Ensuite, j'ai voulu migrer mes modèles vers ma base de données en utilisant la commande **`npx sequelize-cli db:migrate`**. Cette commande a appliqué toutes les migrations en attente à ma base de données, ce qui a permis de créer toutes les tables et les colonnes définies dans mes modèles. Cela m'a permis de m'assurer que ma base de données était à jour avec les dernières modifications de mon code source.

Dans l'ensemble, l'utilisation de Sequelize a grandement simplifié la gestion de ma base de données dans mon projet Node.js.

La méthode "init" est une méthode statique fournie par la classe "Model" de sequelize. Cette méthode est utilisée pour initialiser la définition de notre modèle de données.

Dans notre cas, j'ai utilisé cette méthode pour définir les différentes propriétés de notre entité "User". J'ai défini le type de données, le statut d'obligation ou non de la propriété, et j'ai également ajouté des fonctions de validation pour certaines propriétés.

J'ai également passé quelques options supplémentaires à cette méthode. Par exemple, j'ai fourni l'instance de "sequelize" que j'utilise pour la gestion de la base de données et j'ai donné un nom à notre modèle pour faciliter sa référence ultérieurement.

Après avoir créé mon "model", j'ai dû effectuer une migration pour créer la table correspondante dans ma base de données. Pour cela, j'ai utilisé la commande "`npx sequelize-cli migration:generate`" pour générer un fichier de migration vide, que j'ai ensuite rempli avec le code nécessaire pour créer ma table.

Dans ce fichier de migration, j'ai spécifié le nom de ma table ainsi que les différentes colonnes que je voulais y ajouter en utilisant la syntaxe fournie par Sequelize.

J'ai également spécifié les types de données pour chaque colonne, ainsi que les contraintes de validation nécessaires.

Une fois le fichier de migration rempli, j'ai utilisé la commande "npx sequelize-cli db:migrate" pour exécuter cette migration et créer ma table dans ma base de données.

Model Post

```
'use strict';    Tchessi, 5 months ago • Back-end api init ...
const { Model } = require('sequelize');

const moment = require('moment');

const { deleteFile } = require('../services/file-removal');

module.exports = (sequelize, DataTypes) => {
  Tchessi, 5 months ago | 1 author (Tchessi)
  class Post extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatic
     */
    static associate(models) {
      Post.belongsTo(models.User, { foreignKey: 'userId' });
      Post.hasMany(models.Comments);
      Post.hasMany(models.Likes);
    }

    readableCreatedAt() {
      return moment(this.createdAt).locale('fr').format('LL');
    }
  }
  Post.init(
    {
      userId: DataTypes.INTEGER,
      content: DataTypes.TEXT,
```

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 2. Précisez les moyens utilisés :

J'ai utilisé :

- VScode
- Node js
- sequelize
- MySQL
- LucidChart

## 3. Avec qui avez-vous travaillé ?

J'ai collaboré avec Samir MOKADDEM et Tchèssi PRE durant ce projet.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme

Chantier, atelier, service ▶ TissApp

Période d'exercice ▶ Du : 02/01/2023 au : 31/01/2023

## 5. Informations complémentaires (facultatif)

## Activité-type 3

### Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 - Portfolio

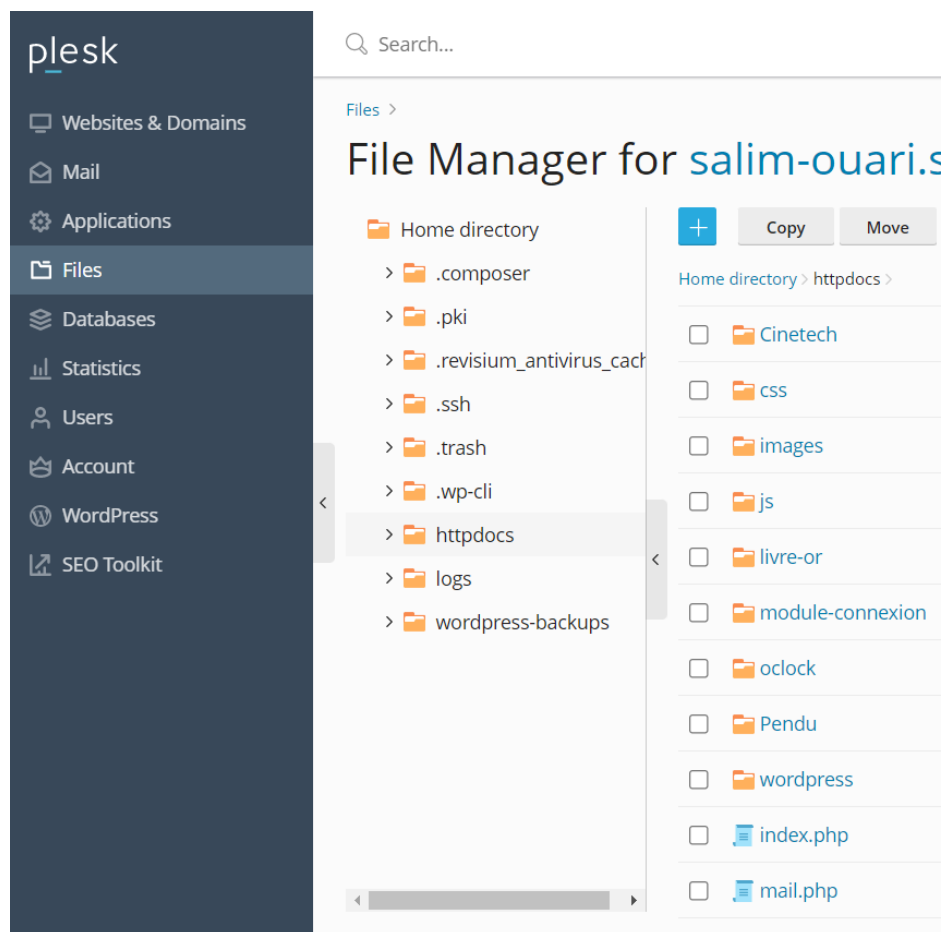
#### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre de ma recherche d' alternance, j'ai pu réaliser un portfolio que j'ai pu déployé sur un hébergeur (plesk)

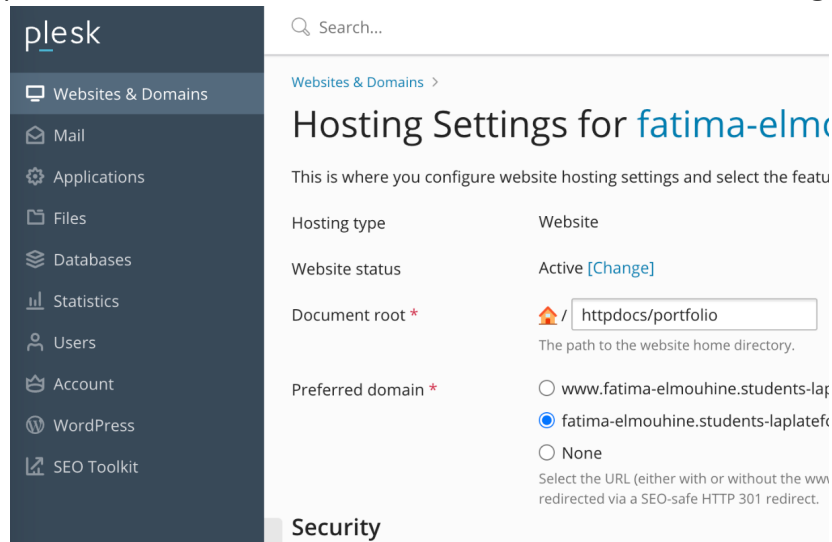
Cet exemple me permet de valider la compétence :

- **Préparer et exécuter le déploiement d'une application**

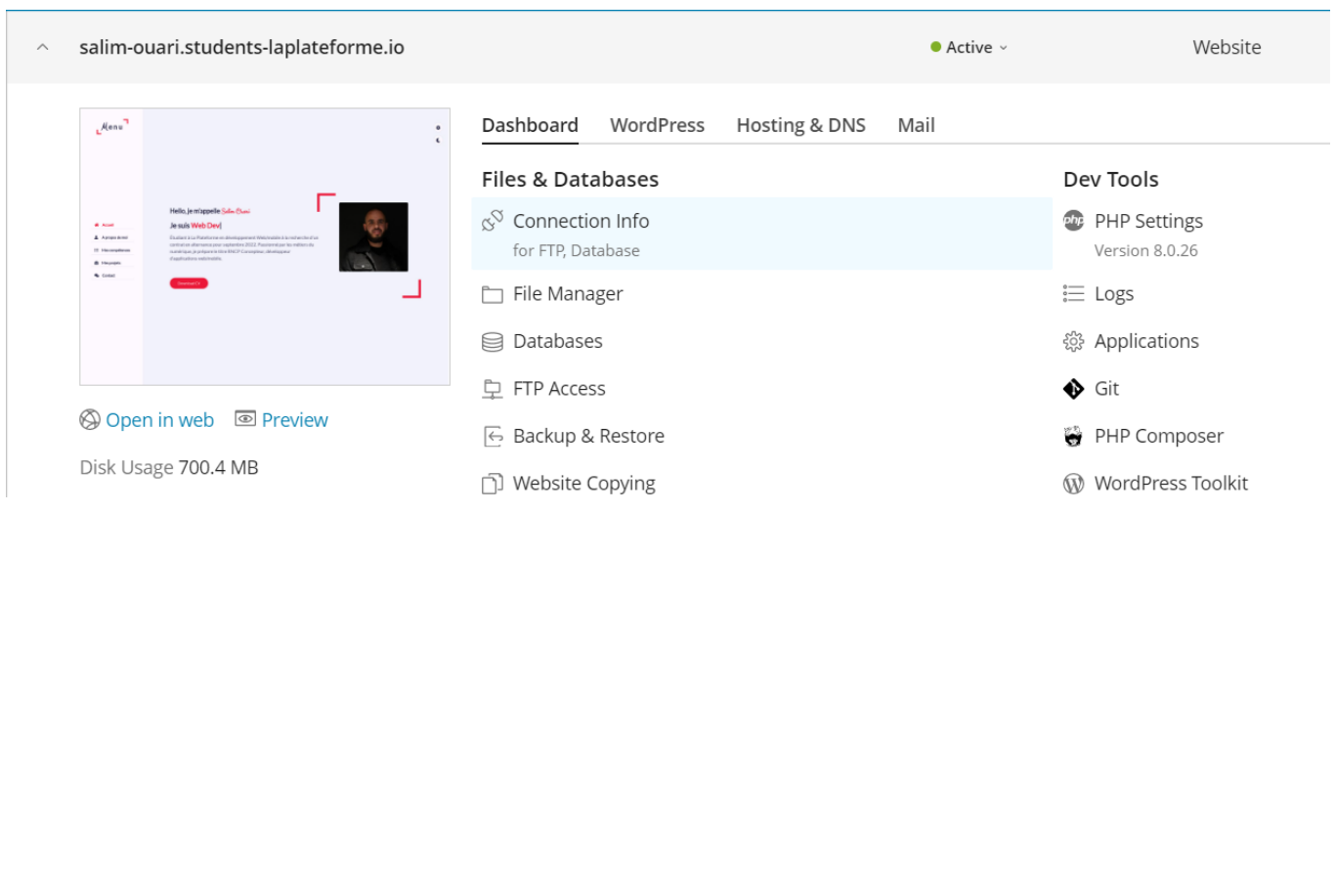
Je mets dans un premier temps mon dossier portfolio directement sur le serveur > dossier httpdocs.



Ensuite, j'indique le point d'entrée de mon domaine, mon site est maintenant en ligne



The screenshot shows the Plesk interface for configuring website hosting settings. On the left is a dark sidebar with the Plesk logo and a menu: Websites & Domains (selected), Mail, Applications, Files, Databases, Statistics, Users, Account, WordPress, and SEO Toolkit. The main area is titled 'Hosting Settings for fatima-elmouhine'. It includes a search bar, a 'Websites & Domains' breadcrumb, and a description: 'This is where you configure website hosting settings and select the features for this website.' The settings are as follows: Hosting type is 'Website'; Website status is 'Active' with a '[Change]' link; Document root is '/httpdocs/portfolio' with a note 'The path to the website home directory.'; Preferred domain is 'fatima-elmouhine.students-laplateforme.io' (selected) with options for 'www.fatima-elmouhine.students-laplateforme.io' and 'None'. A note states: 'Select the URL (either with or without the www) redirected via a SEO-safe HTTP 301 redirect.' At the bottom, there is a 'Security' tab.



The screenshot shows the Plesk Website Dashboard for 'salim-ouari.students-laplateforme.io'. The top bar shows the domain name, a green 'Active' status, and a 'Website' tab. Below the top bar is a navigation menu: Dashboard (selected), WordPress, Hosting & DNS, and Mail. The main content area is divided into three sections. On the left is a preview of the website, showing a 'Hello, j'ai installé WordPress' message and a profile picture. Below the preview are links for 'Open in web' and 'Preview', and a note 'Disk Usage 700.4 MB'. In the center is a 'Files & Databases' section with links: Connection Info for FTP, Database; File Manager; Databases; FTP Access; Backup & Restore; and Website Copying. On the right is a 'Dev Tools' section with links: PHP Settings (Version 8.0.26); Logs; Applications; Git; PHP Composer; and WordPress Toolkit.



# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 2. Précisez les moyens utilisés :

J'ai utilisé :  
- plesk

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La plateforme*

Chantier, atelier, service ▶ *Formation*

Période d'exercice ▶ Du : *20/09/2022* au : *30/10/2022*

## 5. Informations complémentaires *(facultatif)*

### Titres, diplômes, CQP, attestations de formation

*(facultatif)*

Intitulé	Autorité ou organisme	Date
Concepteur développeur d'applications	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

### Déclaration sur l'honneur

---

Je soussigné(e) Ouari Salim ,  
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis  
l'auteur(e) des réalisations jointes.

Fait à Marseille

le 02/06/2023

pour faire valoir ce que de droit.

Signature : ***Ouari Salim***

## Documents illustrant la pratique professionnelle

*(facultatif)*

Intitulé
Cliquez ici pour taper du texte.

---

## DOSSIER PROFESSIONNEL <sup>(DP)</sup>

---

### ANNEXES

---

*(Si le RC le prévoit)*