

DOSSIER PROFESSIONNEL (DP)

<i>Nom de naissance</i>	▶ MOKADDEM
<i>Nom d'usage</i>	▶ -
<i>Prénom</i>	▶ Samir
<i>Adresse</i>	▶ 33 avenue Elleon 13011 Marseille

Titre professionnel visé

Concepteur(trice) Développeur(se) Informatique

MODALITE D'ACCES :

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

DOSSIER PROFESSIONNEL ^(DP)

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.

DOSSIER PROFESSIONNEL ^(DP)



<http://travail-emploi.gouv.fr/titres-professionnels>

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

DOSSIER PROFESSIONNEL ^(DP)

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 :

Au cours de ma formation à La Plateforme, j'ai eu l'opportunité de travailler en équipe sur le développement d'une application mobile de chat en utilisant React Native. Notre objectif principal était de créer une plateforme conviviale et accessible, permettant à tous les utilisateurs de se connecter et de communiquer facilement entre eux. Nous n'avons pas de public cible spécifique en tête, car nous souhaitons que l'application soit ouverte à tous, offrant un espace de détente et de conversation.

- Pour commencer, nous avons rédigé un cahier des charges détaillé. Ensuite, j'ai utilisé l'outil Figma pour créer des maquettes très professionnelles. Avant de me lancer dans la réalisation des maquettes, nous avons défini une charte graphique qui correspondait à notre vision du projet. Figma s'est révélé être un outil collaboratif extrêmement utile, nous permettant d'échanger facilement nos idées et de recueillir les commentaires de toute l'équipe.
- Pour gérer et organiser le projet, nous avons opté pour l'utilisation de Trello. Cette plateforme nous a permis de répartir les tâches en fonction de leur priorité et de leur complexité, assurant ainsi une meilleure organisation et une visibilité claire de l'avancement du projet pour le client.
- Grâce à cette expérience enrichissante, j'ai pu acquérir des compétences précieuses en matière de maquettage d'applications, de gestion de projets informatiques et d'organisation de l'environnement de développement. Ce projet a été une opportunité pour moi de valider les compétences suivantes :
 - **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

3. Avec qui avez-vous travaillé ?

Pour ce projet j'ai travaillé avec [Tchèssi PRE](#) et [Salim OUARI](#)

DOSSIER PROFESSIONNEL (DP)

4. Contexte

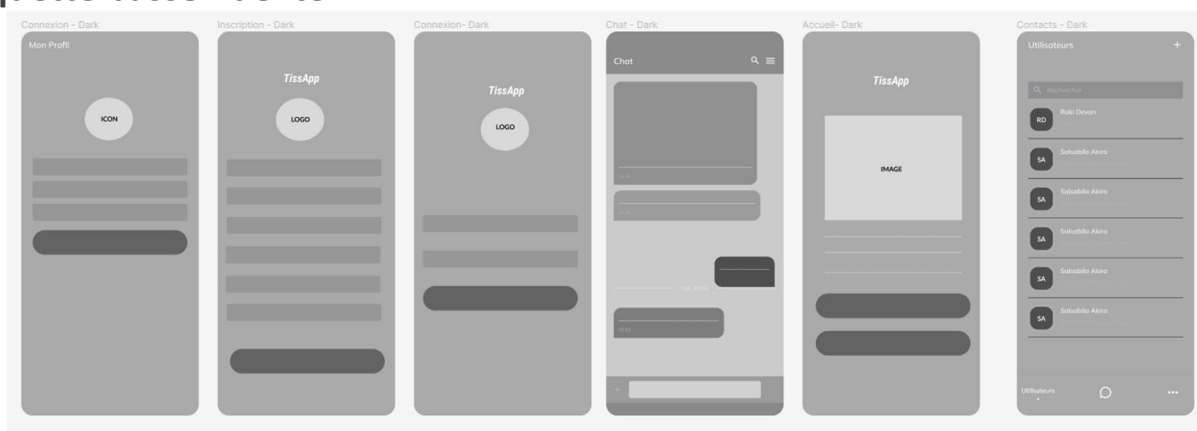
Nom de l'entreprise, organisme ou association ► *La Plateforme*

Chantier, atelier, service ► *TissApp*

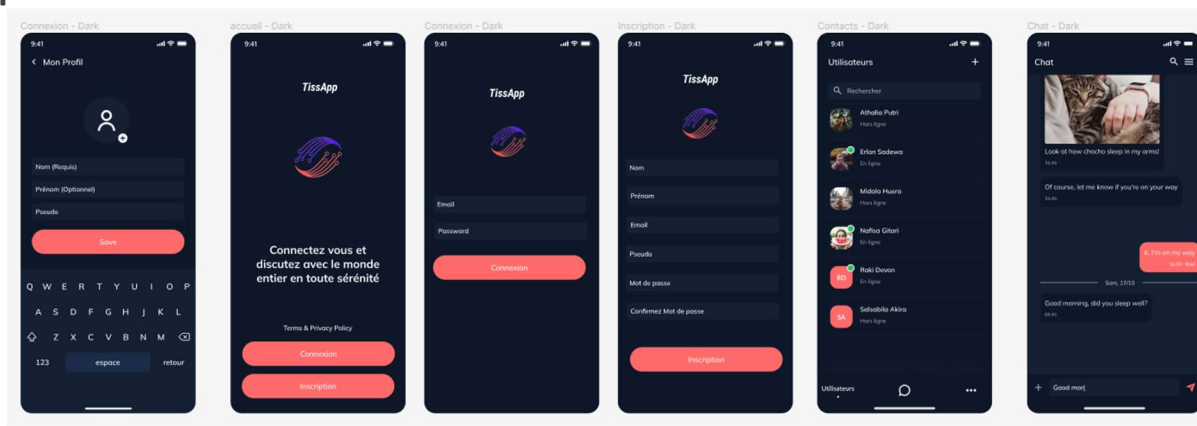
Période d'exercice ► Du : *02/01/2023* au : *31/01/2023*

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 utilisant Python et j'ai utilisé les bibliothèques Pygame et pymysql.

Pour commencer, j'ai utilisé **Pygame** pour créer les éléments graphiques du jeu, ainsi que pour écouter les événements tels que les touches du clavier. J'ai également utilisé la bibliothèque **pymysql** pour interagir avec une base de données MySQL.

Le projet a été mis en place en suivant une approche de programmation orientée objet, ce qui a permis une meilleure interaction entre les différents éléments du jeu et une plus grande lisibilité du code.

Pour démarrer le jeu, j'ai créé un fichier appelé "**game.py**" qui agit comme le point d'entrée de mon jeu. À l'intérieur de ce fichier, j'ai conçu ma **class Menu** qui affiche un menu de sélection avec les options "Jouer" et "Quitter".

Lorsque je choisis l'option "Jouer" dans le menu, je programme la création d'une base de données et d'une table spécifiquement adaptée à l'utilisation de mon jeu. Cela me permet de stocker et de gérer les informations relatives aux scores et aux progrès des joueurs.

Après avoir configuré la base de données, je lance la boucle principale du jeu. Cela me permet de créer l'interface graphique et de permettre au joueur d'interagir avec le jeu en déplaçant les éléments sur le plateau du **Sokoban**. La boucle principale s'occupe également de la détection des événements tels que les mouvements du joueur et les interactions avec les objets du jeu, Le déplacement du personnage, la grille de jeu et la logique sont créés grâce à la **class Player** et la **class Grille**

DOSSIER PROFESSIONNEL (DP)

Au lancement de `game.py` je lance mon menu afin de permettre aux joueurs d'avoir une interface graphique pratique et utile.

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

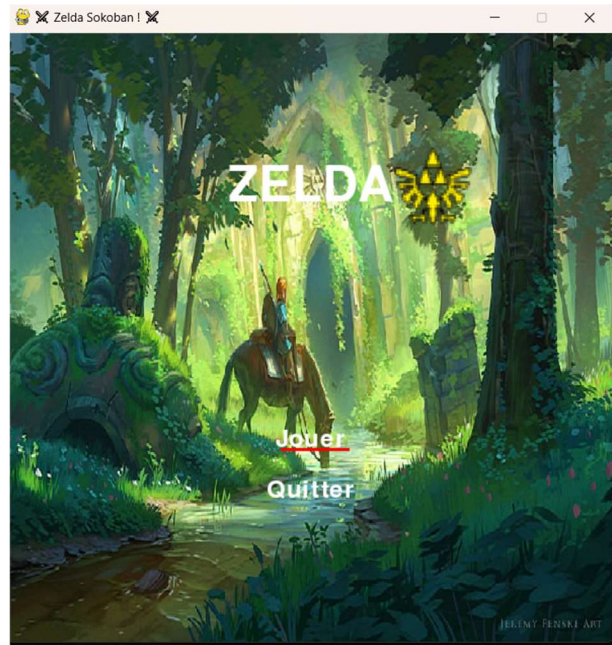
# Création de la fenêtre
screen = pygame.display.set_mode((LARGEUR, HAUTEUR))
pygame.display.set_caption(TITRE)

# Chargement de l'image de fond
background = pygame.image.load("img/back.png")
background = pygame.transform.scale(
    background, (screen.get_width(), screen.get_height()))

# Création du menu
menu = Menu()

# Boucle du menu
while not menu.quit_menu:
    menu.run()

# Vérification de l'état du menu après son exécution
if menu.quit_menu:
    break
```



Grâce à ma `class Player` dans le code du jeu `Sokoban` en Python et `Pygame` sert à gérer le personnage joueur. Elle charge les images du joueur dans différentes orientations et initialise la musique de fond du jeu. Elle possède des méthodes pour dessiner le joueur à l'écran, gérer ses déplacements en réaction aux touches du clavier, et vérifier les collisions avec les murs et les caisses. Elle permet également de définir et d'obtenir le score du joueur.

```
class Player:
    def __init__(self, grille):
        pygame.mixer.music.load(
            "son/Lost Woods - The Legend of Zelda: Ocarina Of Time.wav")
        pygame.mixer.music.set_volume(0.5) # volume réglé à 50%
        pygame.mixer.music.play(-1) # Lecture en boucle infinie
        # Définit un événement lorsque la musique se termine
        pygame.mixer.music.set_endevent(USEREVENT + 1)

        self.gauche = pygame.image.load("img/link_gauche.gif")
        self.droite = pygame.image.load("img/link_droite.gif")
        self.bas = pygame.image.load("img/link_bas.gif")
        self.haut = pygame.image.load("img/link_haut.gif")

        self.position = self.droite

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

DOSSIER PROFESSIONNEL (DP)

Dans ce code, la **class Grille** du jeu gère le terrain de jeu. Elle charge les images utilisées pour représenter les différents éléments de la grille, comme les murs, les caisses et les objectifs. Les informations de la grille sont stockées dans une liste où chaque élément représente une ligne de la grille. La classe offre des méthodes pour dessiner la carte à l'écran, obtenir la position du joueur, déplacer les caisses et vérifier si toutes les caisses ont atteint leurs objectifs. Dans la partie principale du code, **Pygame** est initialisé, la musique de fond est chargée et jouée, et une fenêtre graphique est créée avec la grille affichée à l'intérieur.

```
class Grille:
    def __init__(self, fichier):
        self.ref_img = {
            MUR: pygame.image.load("img/tree.jpg"),
            CAISSE: pygame.image.load("img/hyrule_sign.png"),
            OBJECTIF: pygame.image.load("img/objectif.png"),
            CAISSE_OK: pygame.image.load("img/hyrule_sign.png"),
        }
        with open(fichier, 'r') as fich:
            self.lvtest = [[int(l) for l in line.strip().split(" ")]
                           for line in fich]

        self.coord_objec = []
        for y in range(len(self.lvtest)):
            for x in range(len(self.lvtest[y])):
                if self.lvtest[y][x] == OBJECTIF:
                    self.coord_objec.append((x, y))
        # initialisation du niveau à 1
        self.niveau = 1
```

DOSSIER PROFESSIONNEL (DP)

La boucle principale du jeu **Sokoban** est une boucle while qui s'exécute tant que la variable "continuer" est vraie. Cette boucle gère les événements du jeu et les actions du joueur. Dans cette boucle, il y a des boutons associés à des actions spécifiques. Par exemple, le bouton "X" permet de fermer le jeu, le bouton "R" permet de redémarrer le niveau actuel, le bouton "N" permet de recommencer le jeu depuis le premier niveau.

Enfin, la boucle vérifie si la grille est terminée en appelant la méthode "**is_fini()**" de la classe "**Grille**". Si la grille est terminée, le niveau, le score et l'identifiant du joueur sont enregistrés dans une base de données. La variable "continuer" est également définie sur **False** pour sortir de la boucle principale et passer au donjon suivant.

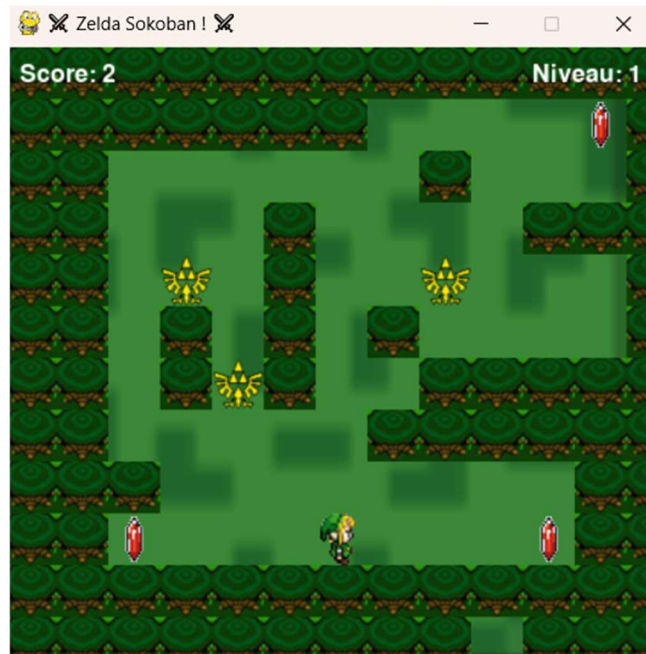
```
# Boucle principale du jeu
continuer = True
victoire = False
display_victoire = False
niveau = 1
niveau_max = 5
id_joueur = uuid_str # Generer un id_joueur unique pour chaque joueur

while continuer:
    # print("Niveau " + str(niveau) + " - Score: " + str(_player.get_score()))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            continuer = False

        # Fermeture du jeu quand je clique X
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_x:
                print("Au revoir Link: " + id_joueur)
                sys.exit()

        # restart le lvl lorsque je clique R
```

DOSSIER PROFESSIONNEL (DP)



2. Précisez les moyens utilisés :

J'ai utilisé :

- **Vscode**
- **Python** comme langage
- **Pip** comme gestionnaire de package
- Les librairies : **pygame** , **pymysql**

3. Avec qui avez-vous travaillé ?

Moi

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 de la formation avec mon équipe, 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**

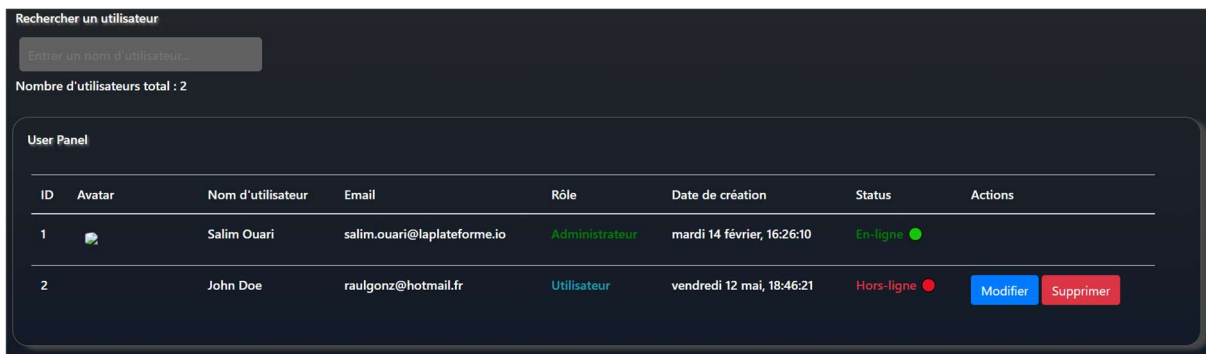
Cette interface contient un accès à un **CRUD** pour gérer les utilisateurs et le chat avec les messages.

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.

DOSSIER PROFESSIONNEL (DP)



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
async deleteUser(userId) {
  try {
    const data = await fetch(`http://localhost:3100/api/auth/users-delete/${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)`.

```
// CTRL ROUTE ADMIN DELETE USER
exports.deleteUserAccount = async (req, res, next) => {
  try {
    const user = req.user.admin
      ? await User.findOne({ where: { id: req.params.id } })
      : req.user;
    // console.log(user.user.admin);
    await user.softDestroy();
    res.status(200).json({ message: "Compte supprimé" });
  } catch (error) {
    res.status(400).json({ error });
  }
};
```

Dans ce contrôleur j'utilise le contrôleur `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 ^(DP)

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 :

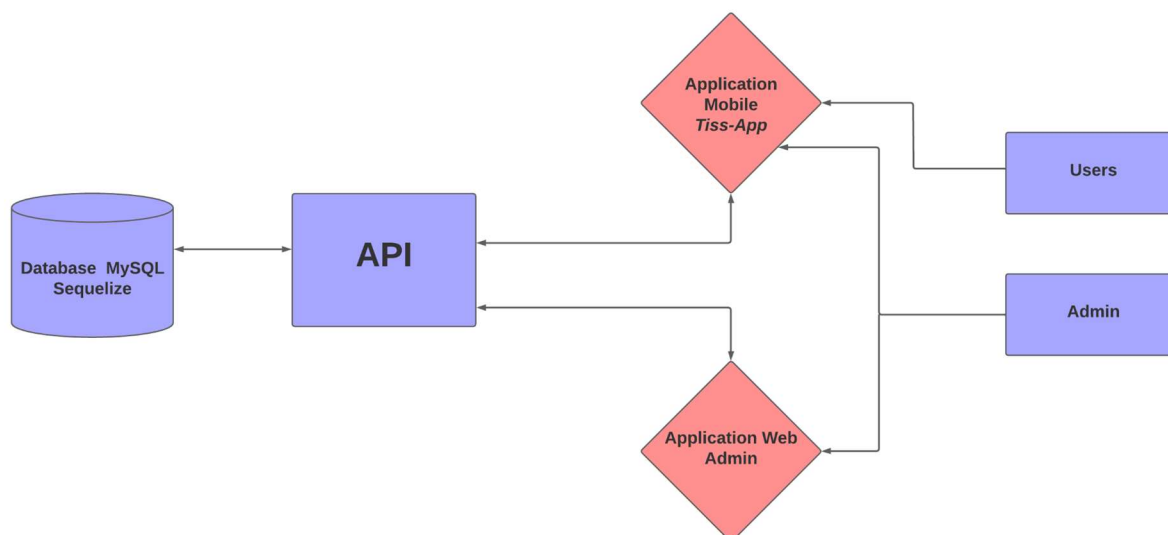
Après avoir pris en note les besoins de l'application dans le cahier des charges, nous avons consacré du temps à l'élaboration d'un plan détaillé. Ce plan comprenait les différentes fonctionnalités à implémenter, les tâches à effectuer, ainsi que les délais et les ressources nécessaires. Nous avons également défini une structure de navigation pour l'application, en identifiant les différentes pages et les flux d'interaction entre elles. Ce processus de planification nous a permis de clarifier nos objectifs, de répartir les responsabilités au sein de l'équipe de développement et d'avoir une vision claire de la façon dont l'application serait développée et livrée.

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**

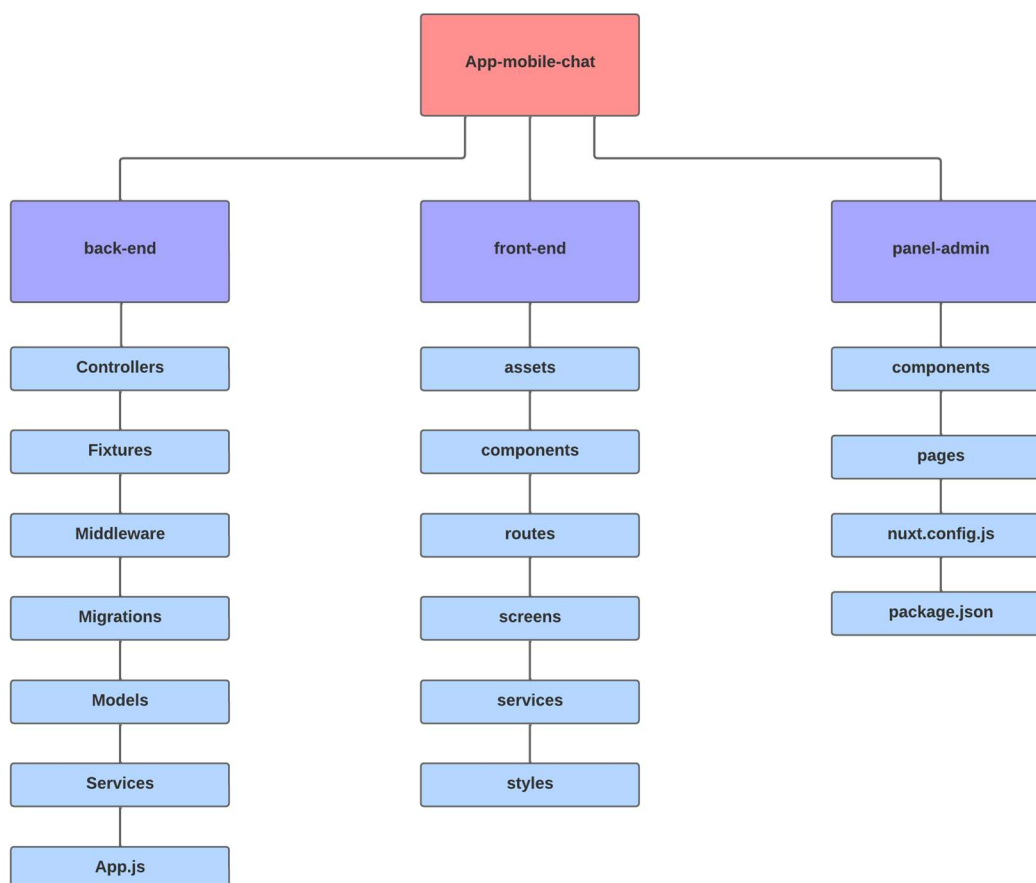
En fin de compte, grâce à une planification minutieuse, une utilisation judicieuse des outils de développement et une collaboration efficace au sein de l'équipe, nous avons pu réaliser les pages principales de l'application conformément aux besoins spécifiés dans le cahier des charges. Cela marque une étape importante dans le développement de l'application, en nous rapprochant de notre objectif final de livrer un produit fonctionnel et convivial.

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



Nous avons utilisé Expo pour avancer dans ce projet cela nous a simplifié le développement en fournissant une approche simplifiée pour créer des applications mobiles avec React Native. Grâce à Expo, j'ai pu bénéficier des fonctionnalités natives préconstruites, de la configuration simplifiée du projet et de l'accès à une variété d'outils et de bibliothèques utiles. L'un des avantages supplémentaires d'Expo est la possibilité de créer un fichier APK (Android Package Kit) de mon application. Cela m'a permis de distribuer facilement mon application aux utilisateurs Android sans avoir à passer par le processus de publication sur le Google Play Store. J'ai également inclus des instructions détaillées dans le fichier README pour guider les utilisateurs sur l'installation et l'exécution de l'application APK. Cela a facilité le déploiement et l'utilisation de mon application sur les appareils Android.

Arborescence du projet

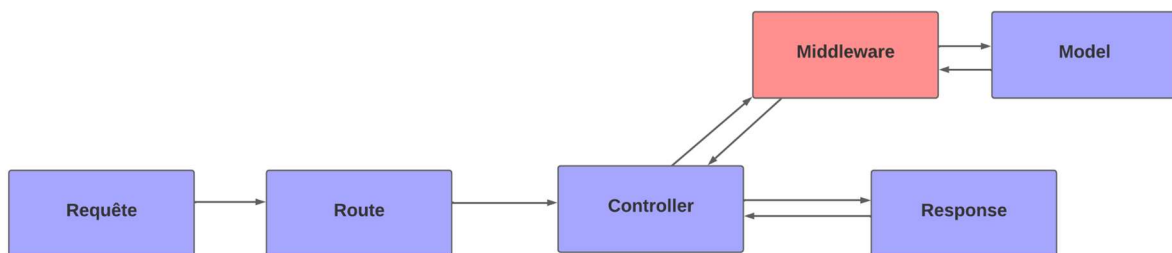


DOSSIER PROFESSIONNEL (DP)

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);
```

Fonctionnement d'un middleware

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.

DOSSIER PROFESSIONNEL (DP)

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

module.exports = (req, res, next) => {
  try {
    const token = req.headers.authorization.split(' ')[1]; //récupération du token depuis
    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('Requête non authentifiée !'),
    });
  }
};
```

Fonctionnement des 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 **400** 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.

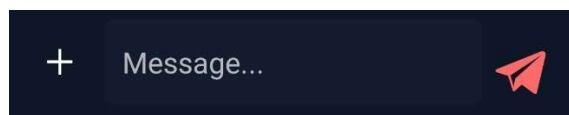
```
exports.signup = async (req, res, next) => {
  if (
    !req.body.firstName ||
    !req.body.lastName ||
    !req.body.email ||
    !req.body.password
  ) {
    res.status(400).json({ error: "Vous devez fournir tous les champs" });
  } else {
    try {
      // Créer un nouvel utilisateur
      const user = await User.create({
        firstName: req.body.firstName,
        lastName: req.body.lastName,
        email: req.body.email,
        password: req.body.password,
      });
      res.status(201).json(newToken(user));
    } catch (error) {
      // Vérification de l'email unique d'un utilisateur
      if (
        error.name === "SequelizeValidationError" &&
        error.errors[0].path === "email"
      ) {
        res.status(409).json({ error: "Email déjà utilisé" });
        console.log("Email déjà utilisé");
      } else {
        res.status(400).json({ error });
        console.log(error);
      }
    }
  }
}
```

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.



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

export default function ImageUploadMessage() {
```

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.

DOSSIER PROFESSIONNEL (DP)

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");
}
```

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}>
          <Icons name="add-outline" size={24} color="white" />
        </TouchableOpacity>
      ) : (
        <Image alt="Image preview" style={PostStyle.imagePreview} />
      )}
    </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 ^(DP)

2. Précisez les moyens utilisés :

- **React Native**
- **Node js**
- **sequelize**
- **Expo**
- Librairies disponible package.json

3. Avec qui avez-vous travaillé ?

J'ai collaboré avec [Salim Ouari](#) 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 aux utilisateurs, aux messages...

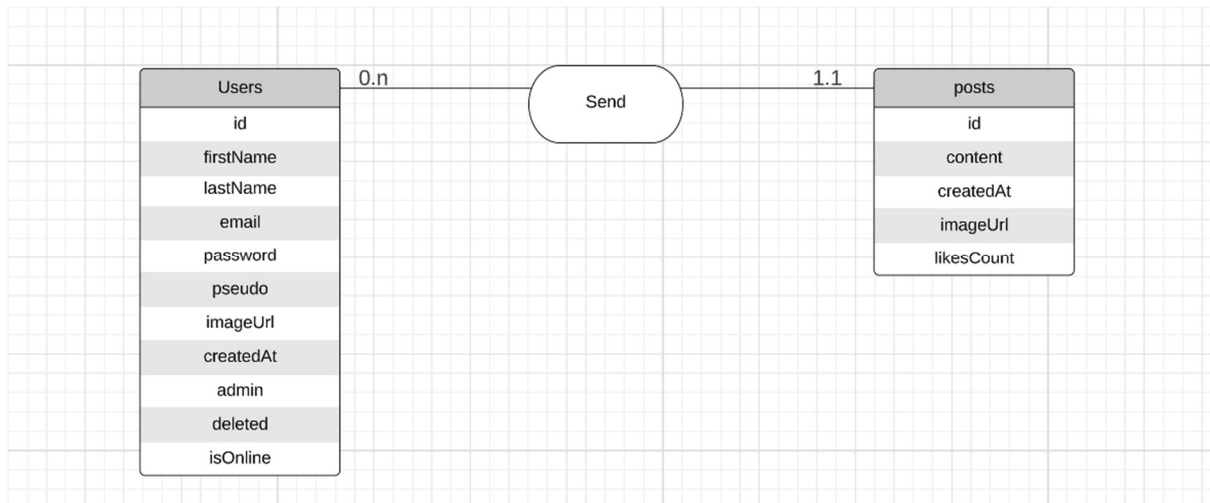
On a utilisé Node Js ainsi que la librairie sequelize qui prend en charge MySQL et offre une prise en charge solide des transactions et des relations. Sequelize est un ORM ("Object Relational Mapping") qui sert à mettre à disposition des classes objet permettant de manipuler les bases de données relationnelles.

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.

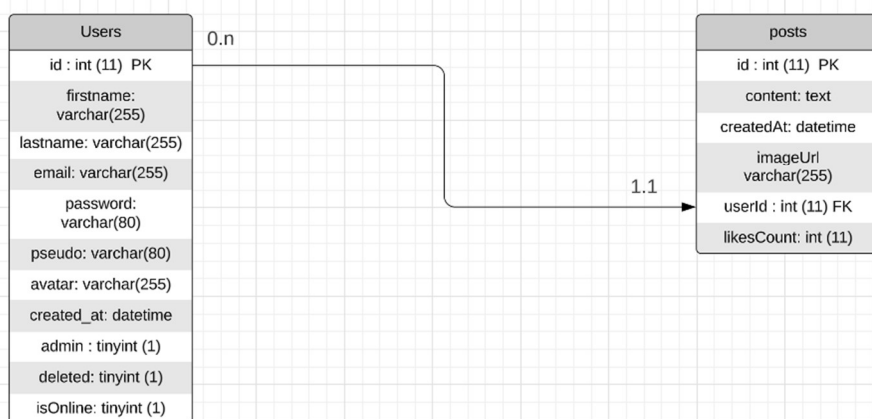
DOSSIER PROFESSIONNEL (DP)

MCD



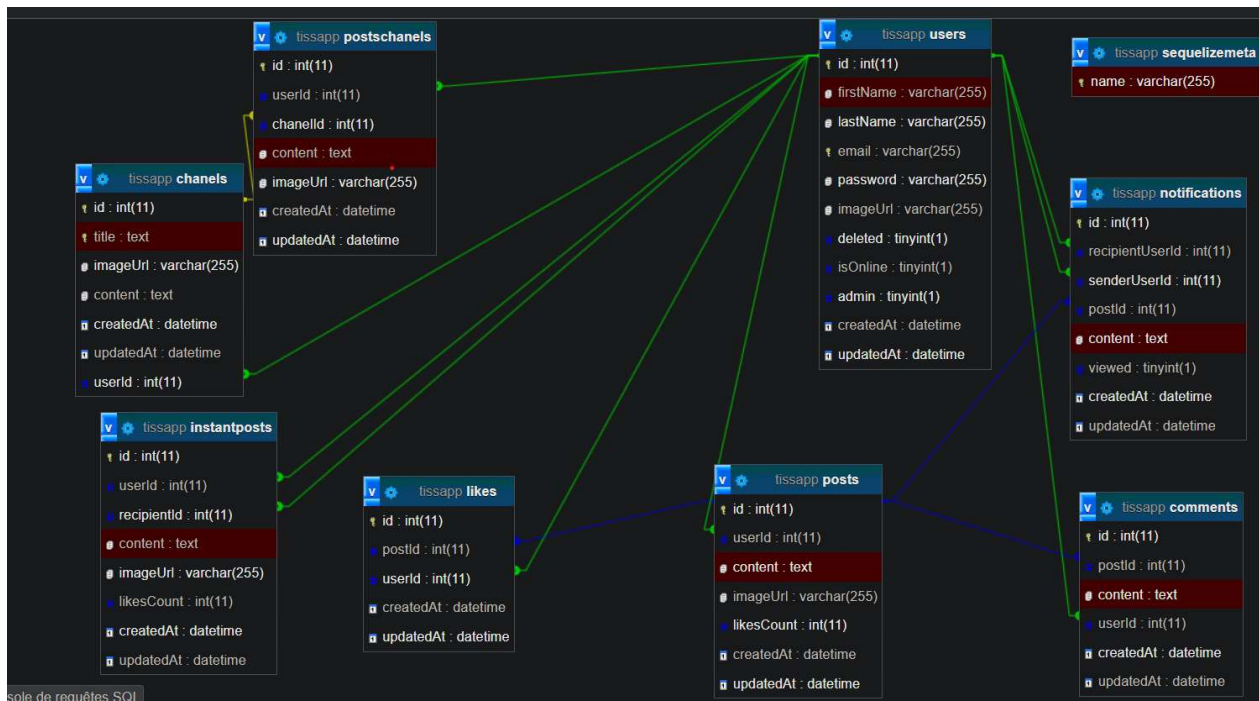
MLD

Diagram de MLD Chat



DOSSIER PROFESSIONNEL (DP)

Représentation de la base de données



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.

DOSSIER PROFESSIONNEL ^(DP)

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 automatically.
     */
    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 ^(DP)

2. Précisez les moyens utilisés :

J'ai utilisé :

- VScode
- Node js
- sequelize
- MySQL
- LucidChart

3. Avec qui avez-vous travaillé ?

J'ai collaboré avec [Salim Ouari](#) 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 et projet

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 déployer sur un hébergeur (plesk)

Cet exemple me permet de valider la compétence :


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

Pour déployer mon portfolio sur plesk , j'ai créé mon domaine en générant un nom de domaine.

Websites & Domains >

Hosting Settings for awesome-boyd.82-165-185-52.plesk.page ▾

This is where you configure website hosting settings and select the features available for your site.

Domain name *	www. <input type="text" value="awesome-boyd.82-165-185-52"/>
	<small>For example, example.com</small>
Hosting type	Website [Change]
Website status	Active [Change]
Document root *	 / <input type="text" value="awesome-boyd.82-165-185-52"/>
	<small>The path to the website home directory.</small>
Preferred domain *	<div><input type="radio"/> www.awesome-boyd.82-165-185-52.plesk.page</div> <div><input checked="" type="radio"/> awesome-boyd.82-165-185-52.plesk.page</div> <div><input type="radio"/> None</div>
	<small>Select the URL (either with or without the www. prefix) to which site visitors will be redirected via a SEO-safe HTTP 301 redirect.</small>

DOSSIER PROFESSIONNEL (DP)

Une fois mon nom de domaine, j'ai téléchargé mon dossier de portfolio directement sur le serveur Plesk, dans le répertoire correspondant à mon nom de domaine. Cela me permet d'héberger mon portfolio en ligne et de le rendre accessible aux visiteurs via mon site web.

Après m'être connecté à mon compte Plesk, j'ai accédé à la section "Fichiers" dans mon panneau de contrôle. Ensuite, j'ai navigué jusqu'au répertoire racine de mon nom de domaine.

J'ai procédé à l'importation de mon dossier portfolio dans ce répertoire. J'ai veillé à ce que tous mes fichiers HTML, CSS, JavaScript, images et autres ressources soient inclus dans le dossier. Il est essentiel que les fichiers de mon portfolio soient directement présents dans le répertoire racine du nom de domaine, sans sous-répertoire supplémentaire. Cela garantit que mon portfolio sera correctement accessible lorsque les visiteurs accèderont à mon site web.

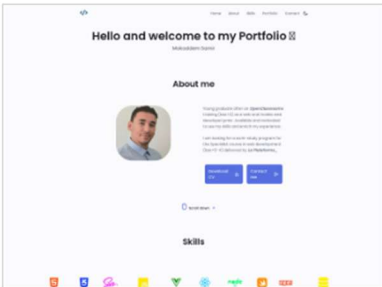
File Manager for [awesome-boyd.82-165-185-52.plesk.page](#)

The screenshot displays the Plesk File Manager interface for the domain [awesome-boyd.82-165-185-52.plesk.page](#). On the left, a sidebar shows the 'Home directory' with a list of folders including .cache, .composer, .config, .nodenv, .npm, .phpenv, .pki, .revisium_antivirus_c, .ssh, .trash, .wp-cli, .wp-toolkit, .yarn, amazing-goldwasser., and awesome-boyd.82-16. The main area shows a table of files and folders in the 'Home directory'.

<input type="checkbox"/>	Name ↑	Modified	Size	Permissions	User	Group
	..	June 18, 2023 06:28 PM		rwX --X --	samir.mokaddem	psaserv
<input type="checkbox"/>	.git	June 18, 2023 06:29 PM		rwX r-X r-X	samir.mokaddem	psacln
<input type="checkbox"/>	.vscode	June 18, 2023 06:29 PM		rwX r-X r-X	samir.mokaddem	psacln
<input type="checkbox"/>	assets	June 18, 2023 06:29 PM		rwX r-X r-X	samir.mokaddem	psacln
<input type="checkbox"/>	blog-php-master	June 18, 2023 06:29 PM		rwX r-X r-X	samir.mokaddem	psacln
<input type="checkbox"/>	Blog-symfony-project-main	June 18, 2023 06:30 PM		rwX r-X r-X	samir.mokaddem	psacln
<input type="checkbox"/>	Calculatrice-React-LaPlateforme-main	June 18, 2023 06:30 PM		rwX r-X r-X	samir.mokaddem	psacln
<input type="checkbox"/>	mokaddem_7_01052022-main	June 18, 2023 06:30 PM		rwX r-X r-X	samir.mokaddem	psacln
<input type="checkbox"/>	index.html	June 18, 2023 06:32 PM	21.3 KB	rw- r-- r--	samir.mokaddem	psacln

DOSSIER PROFESSIONNEL ^(DP)

awesome-boyd.82-165-185-52.plesk.page Active Website



[Open in web](#) [Preview](#)

Disk Usage 0 MB

Traffic 0 MB/month

[Web Statistics SSL/TLS](#)

Dashboard

- Hosting & DNS
- Mail

Files & Databases

- Connection Info
for FTP, Database
- File Manager
- Databases
- FTP Access
- Backup & Restore
- Website Copying

Dev Tools

- PHP Settings
Version 8.1.13
- Logs
- Applications
- Git
- PHP Composer
- Install WordPress
- Node.js
- SEO Toolkit
- Website Importing

Security

- SSL/TLS Certificates
Security can be improved
- Password-Protected Directories
- Web Application Firewall
- Advisor

Website at [awesome-boyd.82-165-185-52.plesk.page](#)
IP address 82.165.185.52 System user samir.mokaddem
[Add description](#)

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

J'ai utilisé :
- plesk

3. Avec qui avez-vous travaillé ?

Moi

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)

DOSSIER PROFESSIONNEL ^(DP)

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.

DOSSIER PROFESSIONNEL ^(DP)

Déclaration sur l'honneur

Je soussigné(e) MOKADDEM Samir ,

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 09/06/2023

Pour faire valoir ce que de droit.

Signature : ***Mokaddem Samir***

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL ^(DP)

ANNEXES

(Si le RC le prévoit)