Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance - *Brierre*
Nom d'usage - *Brierre*
Prénom - *Criss*
Adresse - *56 rue jean mermoz*

Titre professionnel visé

Concepteur développeur d'applications

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.

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

Sommaire

Exemples de pratique professionnelle

DOSSIER PROFESSIONNEL (DP)

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

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

Maquetter une application

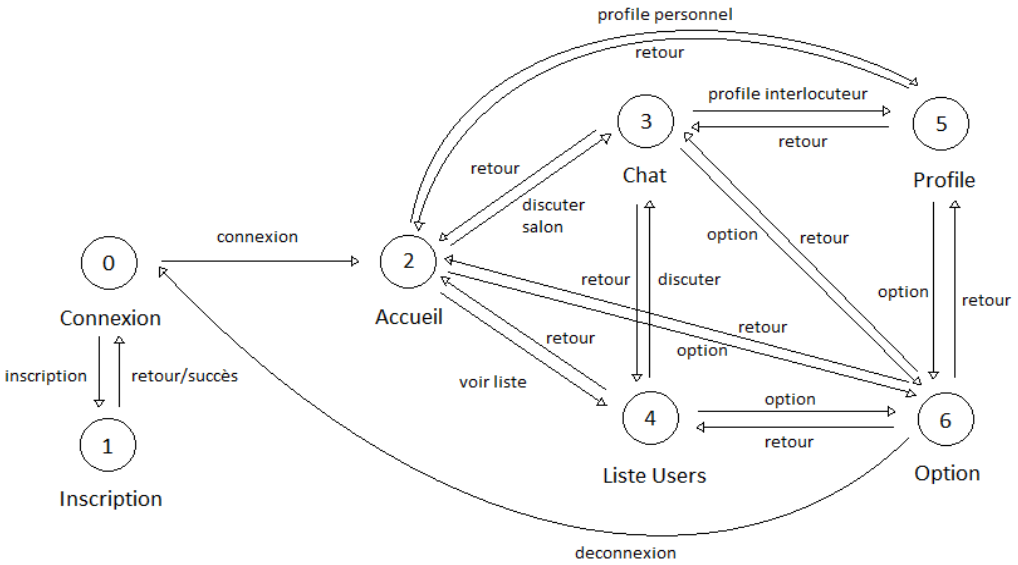
1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

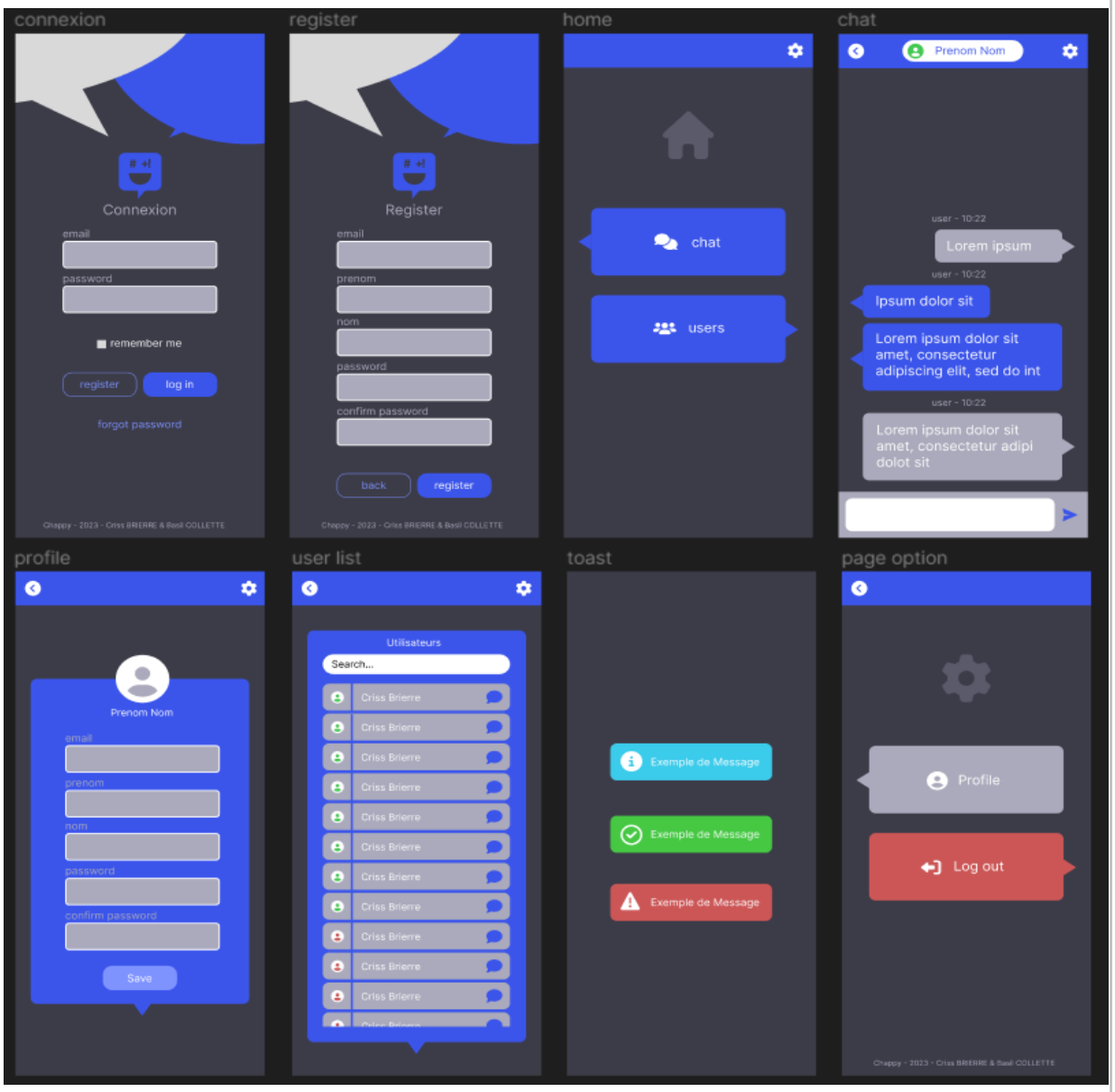
Le maquettage était pour nous l'occasion d'expérimenter les notions d'UI/UX. L'UI (Interface utilisateur) se réfère à l'aspect visuel et à l'interaction de l'application, tandis que l'UX (Expérience utilisateur) englobe l'ensemble des émotions et des perceptions ressenties par l'utilisateur lors de son utilisation.

Pour cela, nous avons utilisé Figma un outil de conception d'interface utilisateur (UI) basé sur le cloud, qui permet aux équipes de collaborer en temps réel sur la création de maquettes interactives, de prototypes et de spécifications de conception pour les applications web et mobiles.

J'ai utilisé des SVG trouvés sur le net pour servir d'icône à des boutons ou à titre informatif.

Le logo de l'application à été réalisé avec Paint.net.




connexion
Connexion
email
password
remember me
register
log in
forgot password
register
Register
email
prenom
nom
password
confirm password
back
register
home
chat
users
chat
Prenom Nom
user - 10:22
Lorem ipsum
user - 10:22
Ipsum dolor sit
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do int
user - 10:22
Lorem ipsum dolor sit amet, consectetur adipi dolot sit
profile
Prenom Nom
email
prenom
nom
password
confirm password
Save
user list
Utilisateurs
Search...
Criss Brierre
toast
Exemple de Message
Exemple de Message
Exemple de Message
page option
Profile
Log out

2. Précisez les moyens utilisés :

Figma (éditeur de graphiques vectoriels)

SVG trouvé sur le net

Conseil en UI/UX trouvé sur le net

3. Avec qui avez-vous travaillé ?

J'ai effectué ce travail avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02 Janvier 2023 au : 04 Janvier 2023

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL (DP)

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

Développer une interface utilisateur de type desktop – G.D.I URSOL

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

Au cours de ma formation en BTS SIO SLAM (Solutions logicielles applications métiers), j'ai effectué un stage de 6 semaines durant ma 2 -ème années à l'association Urgences et solidarités basé sur Aubagne.

Présentation de l'association :

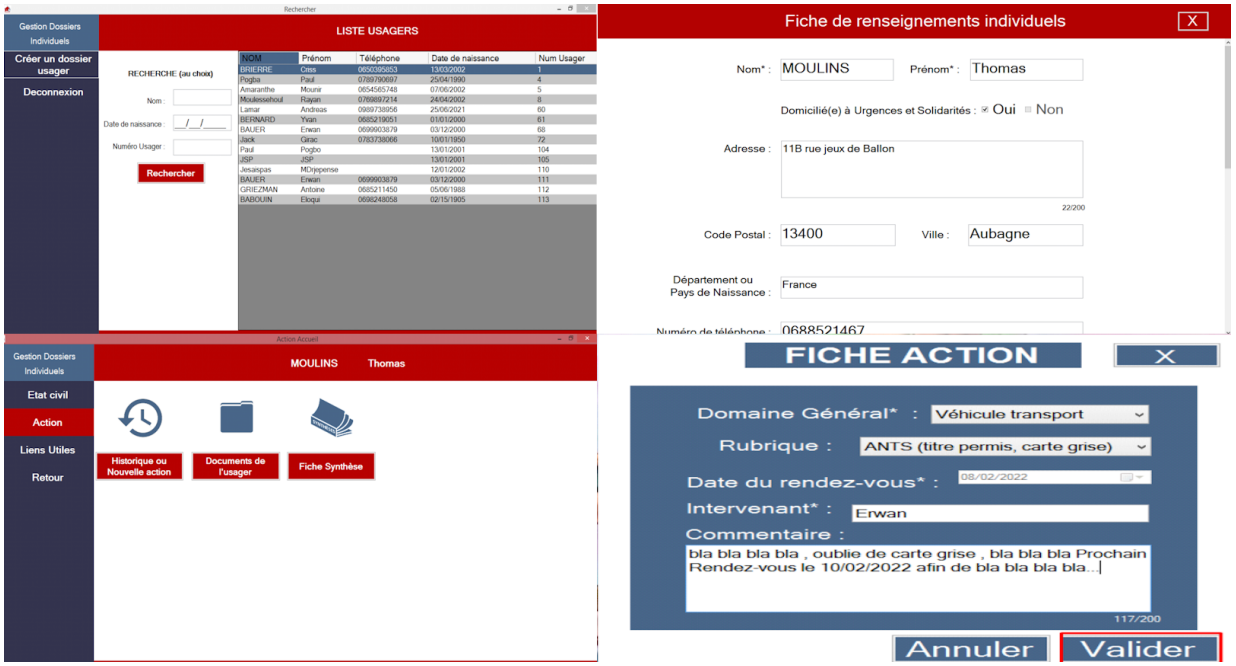
Urgence et solidarité est une association qui offre un soutien aux adultes sans domicile fixe ou en difficulté, en leur fournissant une **assistance administrative**, des **repas**, des **services d'hygiène** tels que le lavage du linge, l'accès aux douches et aux toilettes. Les bénévoles accordent une grande importance au respect de la dignité des personnes aidées. Il s'agit d'un lieu de vie et de rencontre accueillant des individus aux parcours variés, indépendamment de leur origine ou de leur religion.

Mission du stage :

L'association avait besoin d'une solution logicielle pour le pôle assistance administrative, en effet les bénévoles sont amenés à faire des démarches administratives pour les personnes ayant du mal à parler/écrire français.

C'est pour répondre à ce besoin que j'ai développé une solution logicielle permettant de stocker les informations personnelles des individus. Cette application a été conçue en utilisant les technologies WinForms, .NET Framework et le langage C#. J'ai également mis en place une base de données MySQL et établi une connexion FTP entre le client et le serveur pour le stockage de fichiers tels que des PDF et des documents Word. Enfin, j'ai créé un générateur de fiches synthèse qui récapitule les actions effectuées lors des rendez-vous avec les personnes aidées. Ce générateur permet de générer un fichier PDF pouvant ensuite être imprimé.

Aperçu visuel de l'application :



J'ai choisi d'utiliser des classes dans mon projet pour structurer efficacement mes données et regrouper les fonctionnalités associées, ce qui facilite la maintenance et la compréhension du code : une classe qui sert à interagir avec la base de données “LienBdd”, une classe qui sert à interagir avec le protocole Ftp pour les fichiers “LienFtp”.

LienBdd :

```
/// </summary>
17 références
public LienBdd()
{
    this.cn = new MySqlConnection("server=localhost; Database=gestiondossier ;uid=root ;pwd=; CharSet = utf8;");
    try
    {
        this.cn.Open();
        if (this.cn.State != ConnectionState.Open)
            return;
    }
    catch (MySqlException ex)
    {
        MessageBox.Show(ex.Message);
    }
}
/// <summary>
/// Méthode permettant de fermer la connexion
/// </summary>
0 références
public void FermerConnexion()
{
    this.cn.Close();
}

1 référence
public DataTable ObtenirDossier()
{
    String req = "select nom AS NOM, prenom AS Prénom, Tel AS Téléphone, date_naissance AS Date de naissance, num
    this.cde = new MySqlCommand(req, cn);
    da = new MySqlDataAdapter();
    da.SelectCommand = this.cde;
    dt = new DataTable();
    //Le DataAdapter da va se charger ensuite de remplir la DataTable
    da.Fill(dt);

    return dt;
}
```

LienFtp :

```
class LienFtp
{
    private string Username = "criss"; //test
    private string Server = "ftp://127.0.0.1"; //ftp://192.168.1.11
    private string Password; //ursol
    private string path = @"C:\Users\Public\Fichiers récupérés\";


    4 références
    public bool CreateFTPDirectory(string namedirectory)
    {
        FtpWebResponse response;

        try
        {
            //create the directory
            FtpWebRequest requestDir = (FtpWebRequest)FtpWebRequest.Create(Server + "/fichier/" + namedirectory);
            requestDir.Method = WebRequestMethods.Ftp.MakeDirectory;
            requestDir.Credentials = new NetworkCredential(Username, Password);
            requestDir.UsePassive = true;
            requestDir.UseBinary = true;
            requestDir.KeepAlive = false;
             response = (FtpWebResponse)requestDir.GetResponse();
            Stream ftpStream = response.GetResponseStream();

            ftpStream.Close();
            response.Close();

            return true;
        }
        catch (WebException ex)
        {
             response = (FtpWebResponse)ex.Response;
            if (response.StatusCode == FtpStatusCode.ActionNotTakenFileUnavailable)
            {
                response.Close();
                return true;
            }
            else
            {
                response.Close();
```

2. Précisez les moyens utilisés :

- Visual studio 2019
- Wamp Server

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec Erwan Bauer

4. Contexte

Nom de l'entreprise, organisme ou association : *Urgences et solidarités*

Chantier, atelier, service : *Cliquez ici pour taper du texte.*

Période d'exercice : Du : *02 Janvier* au : *11 Février*

5. Informations complémentaires *(facultatif)*

Si je pouvais revenir en arrière, j'utiliserais aussi le protocole FTPS plutôt que le protocole FTP car il chiffre les données transférées, c'est donc plus sécurisé.

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

Développer des composants d'accès aux données - Chappy

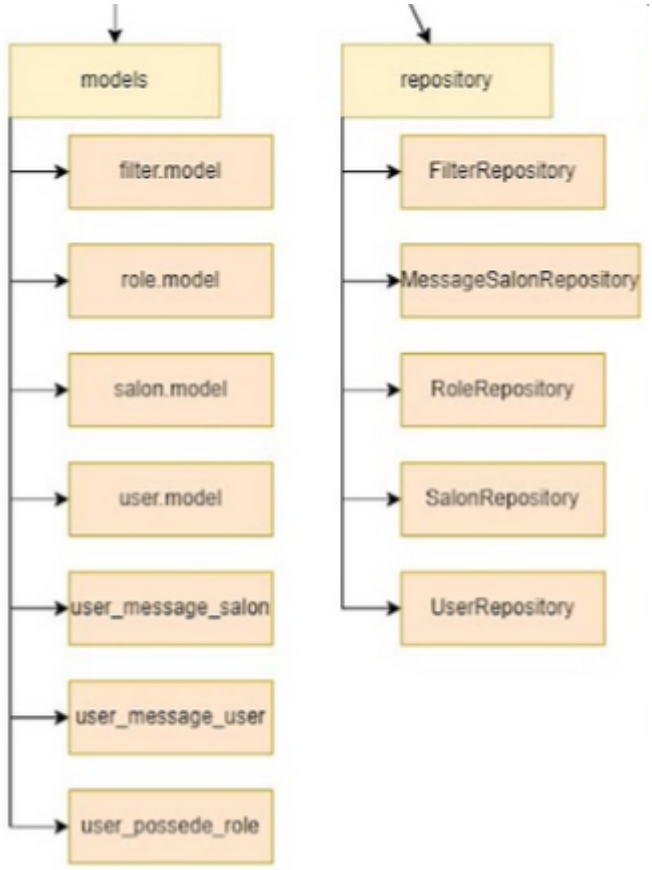
1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour que notre application puisse interagir avec la base de données, nous avons choisi d'utiliser un ORM : Sequelize. Notre choix s'est porté vers l'usage d'un ORM car cela permet de simplifier et d'automatiser la manipulation des données en tant que couche d'abstraction (entre la base de données et l'application). Ils s'occupent aussi de sanitizer les valeurs utilisées pour éviter les injections SQL.

Nous avons donc choisi l'ORM Sequelize pour sa popularité et sa simplicité. Nous avons défini des modèles dans le dossier "Models", qui sont tout simplement la représentation de nos tables de la base de données mais sous forme de classe.

Par la suite, l'accès aux données se font dans le dossier "repositories" de l'application, c'est ici que toutes nos méthodes y sont renseignées, chaque modèle à un repository qui le représente (exemple le model "user.model" à son propre repository "UserRepository").

Arborescence composant d'accès aux données :



Model “user.model” :

```
module.exports = (sequelize) => {

    const roleModel = require("./role.model")(sequelize);
    const user_possede_role = require("./user_possede_role.model")(sequelize);

    const userModel = sequelize.define("user", {
        idUser: {
            primaryKey: true,
            autoIncrement: true,
            type: Sequelize.NUMBER,
            field: 'pk_id_user',
            allowNull: false
        },
        email: {
            type: Sequelize.STRING,
            allowNull: false
        },
        prenom: {
            type: Sequelize.STRING,
            allowNull: false
        },
        nom: {
            type: Sequelize.STRING,
            allowNull: false
        },
        password: {
            type: Sequelize.STRING,
            allowNull: false
        },
        createdAt: {
            type: Sequelize.DATE,
            allowNull: false,
            field: 'created_at'
        }
    }, {
        tableName: 'user',
        updatedAt: false,
    });

    userModel.belongsToMany(roleModel, {
        through: user_possede_role,
        as: 'roles',
        onDelete: 'RESTRICT',
        onUpdate: 'RESTRICT'
    });

    roleModel.belongsToMany(userModel, {
        through: user_possede_role,
        as: 'users',
        onDelete: 'RESTRICT',
        onUpdate: 'RESTRICT'
    });

    return userModel;
```

2. Précisez les moyens utilisés :

- Visual studio Code

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice : Du : 02/01/2023 au : *05/01/2023*

5. Informations complémentaires *(facultatif)*

Activité-type 1

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

- Développer la partie front-end d'une interface utilisateur web – Chappy

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

Chappy étant une application de chat, un panel admin doit être disponible facilement aux administrateurs pour permettre des actions rapides. Il permet de gérer les utilisateurs, de modérer les discussions et de surveiller l'activité, assurant ainsi un environnement sûr et efficace pour les utilisateurs.

Dans cette démarche d'accessibilité nous avons opté pour une interface web, pour l'accès depuis de nombreux devices sans installation.

Cette interface a été développée en React.js puisque très similaire à React Native que nous utilisions pour l'application, l'intérêt est donc d'utiliser le même stack de compétence pour toutes les parties du projet.

Pour générer les grilles de données sur la page web, j'ai choisi d'utiliser la bibliothèque Ant Design (antd) qui est une bibliothèque javascript (donc utilisable en React) de composants d'interface utilisateur réactifs et prêts à l'emploi pour le développement web. Par souci de rapidité et de simplicité.

Pour pouvoir accéder à l'application d'administration, il faut se connecter en utilisant un email et un password et posséder les droits "administrateur", En effet les routes utilisées par le client admin sont des routes du middleware "Admin", qui vérifie que ce droit est possédé. C'est par l'utilisation du JWT Token qui a été reçu lors de la connexion, que l'API détermine s'il s'agit d'un admin qui effectue la requête.

Composant Auth :

```
export default function AuthComponent(props) {
  const [state, setState] = useState({
    connexionInputs: { email: "", password: "" },
  });

  const navigate = useNavigate();

  const authSubmit = async (e) => {
    try {
      e.preventDefault();
      let resultToken = await apiHttpRequest("user/login/", "POST", null, state.connexionInputs);

      if (resultToken) {
        StoreService.storeData("user", resultToken.user);
        StoreService.storeData("jwttoken", resultToken.token);
        navigate("/admin");
      }
    } catch (err) {
      throw new Error(err);
    }
  };

  const updateInput = (inputName, value) => {
    try {
      const newConnexionInputs = InputService.setInputStates(
        state.connexionInputs,
        inputName,
        value
      );

      setState({
        ...state,
        connexionInputs: newConnexionInputs,
      });

    } catch (err) {
      console.log(err);
    }
  };

  return <AuthTemplate authSubmit={authSubmit} updateInput={updateInput} />;
}
```

ainsi qu'un extrait du template :

```jsx
import styles from "./auth.style.jsx";
import { TextInput, View, Image, TouchableHighlight, Text } from "react-native";
import { Checkbox } from "react-native-paper";

export default function AuthTemplate(props) {
  return (
    <View style={styles.container}>
      <View style={styles.containerHeader}>
        <View style={styles.triangleRight}></View>
        <View style={styles.triangleLeft}></View>
        <View style={styles.circleLeft}></View>
        <View style={styles.circleRight}></View>
      </View>

      <View style={styles.containerForm}>
        <Image style={styles.logo} source={require("@assets/img/logo.png")} />
        <Text style={styles.title}>Connexion</Text>

        <TextInput
          style={styles.inputEmail}
          onChangeText={(e) => {
            props.updateInput("email", e);
          }}
          title="Email"
          placeholder="Email"
          placeholderTextColor="white"
          value={props.rememberMe.email}
        />
        <TextInput
          style={styles.inputPassword}
          onChangeText={(e) => {
            props.updateInput("password", e);
          }}
          title="Password"
          placeholder="Password"
          placeholderTextColor="white"
          secureTextEntry={true}
          value={props.rememberMe.password}
        />

        <View style={styles.containerCheckBox}>
          <Checkbox
            color="white"
            uncheckedColor="white"
            status={props.rememberStatus === true ? "checked" : "unchecked"}
            onPress={() => props.setRememberStatusCheck()}
            onValueChange={(value) => this.toggleRememberMe(value)}
          />
```

2. Précisez les moyens utilisés :

- Visual studio Code

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ► Du : 02/02/2023 au : *09/02/2023*

5. Informations complémentaires *(facultatif)*

Activité-type 1

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

Développer la partie back-end d'une interface utilisateur web – Chappy

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

Puisque les mêmes opérations seront effectuées à la fois pour mon application web et mobile, j'ai opté pour la création d'une API afin d'éviter la duplication de la logique métier. J'ai divisé ma logique en modules distincts, ce qui a amélioré la lisibilité du code et facilite sa maintenance pour les futures versions. J'ai également évité les redondances en créant des fonctions et des services. Ainsi, la structure de mon code est conçue pour être facilement maintenable et réutilisable.

Fonctionnement de l'api

Lorsque le client envoie une requête sur mon API, le fichier server.js analyse l'url et renvoie sur le router correspondant, ces routes appellent ensuite des middlewares qui vont exécuter la logique métier qui elle-même appelle les repositories pour pouvoir insérer, modifier et accéder aux données, les middlewares contiennent aussi un bloc try catch qui permet de renvoyer une réponse en fonction de l'état de la requête, cette réponse est au format JSON.

Dans le cas du client admin web, la route admin référence un routeur contenant plusieurs middlewares routes qui vont alors passées par le pré processus d'authentification du middleware admin, on utilise ainsi le plein potentiel de centralisation des middlewares.

Les méthodes sont commentées pour permettre une rapide compréhension et se repérer dans la maintenance du code:

```
/**
 * check if user is authentified using jwt token
 * set the user object returned, in the res.locals
 */
const isAuthorized = async (req, res, next) => {
    try {
        if (!req.params.idUser) {
            res.status(406).send('idUser GET params is missing');
            return;
        }
        res.locals.userId = parseInt(req.params.idUser);

        if (res.locals.authentifiedUser.idUser != res.locals.userId
            && !UserRepository.isAdmin(res.locals.authentifiedUser)
        ) {
            res.status(406).send('not_authorized');
            return;
        }
        next();
    } catch(err) {
        console.log(err);
        res.status(500).send('error_during_authorization_check');
    }
}

/**
 * check if all inputs required for login are send in body
 */
const loginInputsAreSent = (req, res, next) => {
    const userFields = req.body;

    if (!userFields.email || !userFields.password) {
        res.status(406).send('list of needed inputs is required');
        return;
    }
    next();
}
```

Aussi, puisque cette API est prévue pour être utilisée par plusieurs utilisateurs en même temps, dans le cadre de la programmation POO nous avons adopté un patern nommé “Singletonpour éviter la redondance d’instance similaire en mémoire. En effet, les repository seront strictement les mêmes pour chacun, alors autant utiliser la même adresse mémoire pour chaque requête simultanées.

```
let instance = null;

class UserRepository {

    connexion;
    userModel;
    roleModel;

    constructor() {
        if (!instance) {
            this.connexion = require('../database/sequelize');

            this.userModel = require("../models/user.model")(this.connexion);
            this.roleModel = require("../models/role.model")(this.connexion);

            instance = this;
        }

        return instance;
    }
```

2. Précisez les moyens utilisés :

- Visual studio Code
- Thunder Client (un module VS-Code servant de Postman like)

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02/02/2023 au : *09/02/2023*

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

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

- Concevoir une base de données - Chappy

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

Lors de la création d'un projet d'application mobile de chat en temps réel, il est indispensable d'avoir un moyen de persister les données,il faut faire le choix d'une base de données, nous avons donc réfléchi à quel SGBD utiliser.

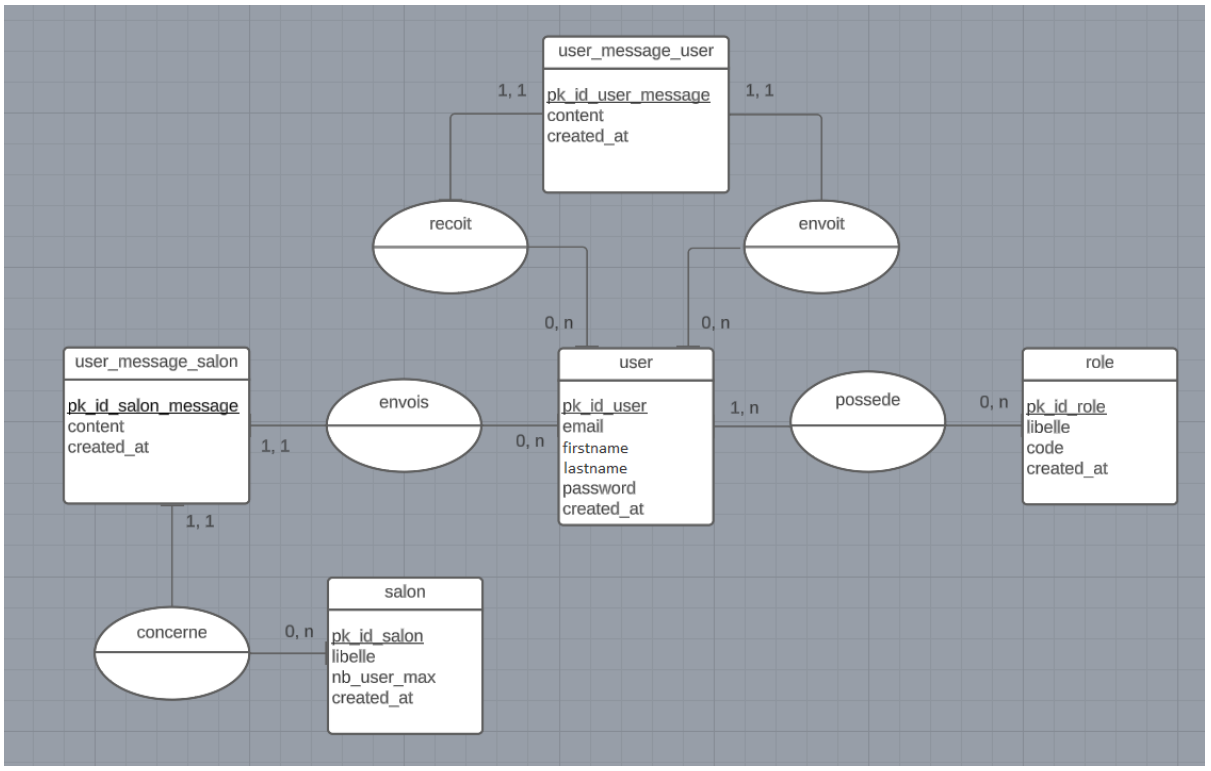
Base de données relationnelle :

Une base de données relationnelle présente divers avantages: elle offre une structure organisée basée sur des tables et des relations claires entre elles, facilitant la gestion des données, les contraintes d'intégrité garantissent la cohérence et la validité des données. Les relations entre les tables maintiennent la cohérence des données. Elles facilitent également l'intégration et la jointure de données provenant de différentes sources. Les SGBD assurent la durabilité des données et permettent le contrôle de la redondance grâce aux contraintes d'unicité .

Ces avantages correspondent exactement au besoin dans notre contexte d'application de messagerie dans laquelle les messages sont liés aux utilisateurs par exemple.

Conception de la base de données :

Pour définir les tables ainsi que leurs relations dans la base de données, il faut faire ce qu'on appelle un MCD (**Modèle Conceptuel de données**) qui va grâce à aux cardinalités définir le type de relations entre-elles, ensuite il faut faire un MLD (**Modèle logique de données**) qui va concrétiser cela en définissant les tables de la base de données.



Nous avons défini 6 tables dans notre MLD

User : correspond aux utilisateurs de l'application

Role : correspond aux rôles des utilisateurs

Salon : correspond aux salons de l'application

User_possede_role : correspond aux différents rôles d'un user

User_message_salon : correspond aux messages envoyés par les utilisateurs dans un salon

User_message_user : correspond aux messages privés de l'application

Après avoir conçu le MLD, il a fallu développer le script SQL afin de créer la base de données.

2. Précisez les moyens utilisés :

- Lucid Chart pour le MCD et MLD
- WampServer
- Moteur MySql

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02/01/2023 au : *04/01/2023*

5. Informations complémentaires *(facultatif)*

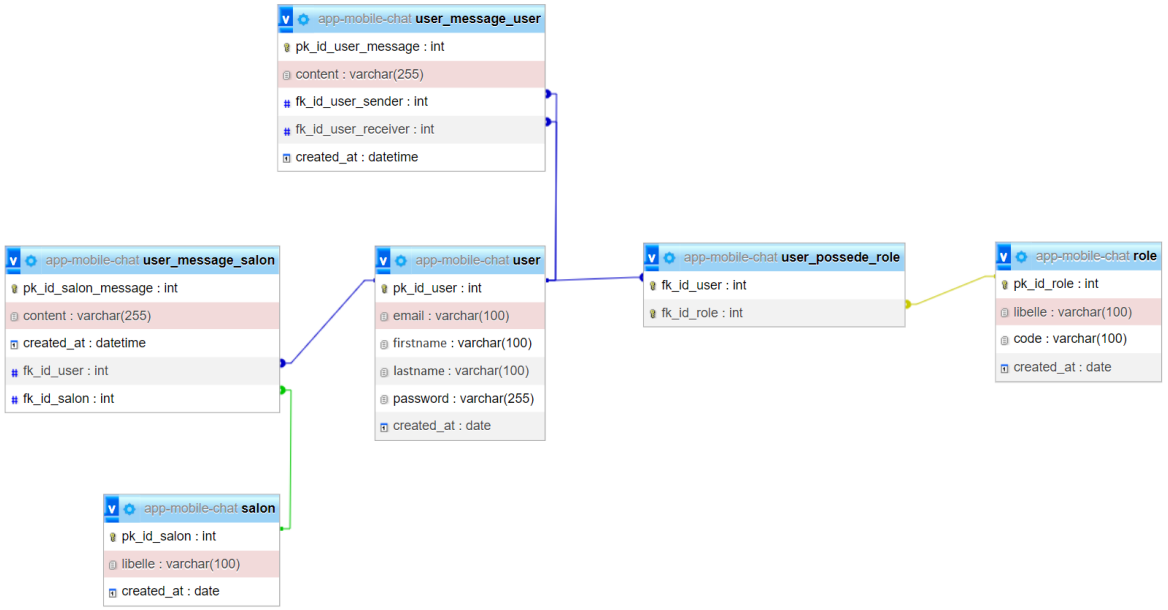
Activité-type 2

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

- Mettre en place une base de données - Chappy

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

Voici le modèle physique de donnée d'après la base qui a été créée conformément aux modèles précédemment définis :



Nous avons aussi généré un script qui permet de recréer la base avec son exécution.
Aperçu :

```sql
DROP TABLE IF EXISTS `role`;
CREATE TABLE IF NOT EXISTS `role` (
  `pk_id_role` int NOT NULL AUTO_INCREMENT,
  `libelle` varchar(100) NOT NULL,
  `code` varchar(100) NOT NULL,
  `created_at` date NOT NULL,
  PRIMARY KEY (`pk_id_role`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

--
-- Déchargement des données de la table `role`
--

INSERT INTO `role` (`pk_id_role`, `libelle`, `code`, `created_at`) VALUES
(1, 'User', 'user', '2022-06-16'),
(2, 'Admin', 'admin', '2022-11-03');

-- --------------------------------------------------------

--
-- Structure de la table `salon`
--

DROP TABLE IF EXISTS `salon`;
CREATE TABLE IF NOT EXISTS `salon` (
  `pk_id_salon` int NOT NULL AUTO_INCREMENT,
  `libelle` varchar(100) NOT NULL,
  `created_at` date NOT NULL,
  PRIMARY KEY (`pk_id_salon`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

--
-- Déchargement des données de la table `salon`
--

INSERT INTO `salon` (`pk_id_salon`, `libelle`, `created_at`) VALUES
(1, 'Général', '2023-01-10');
```

Pour la sauvegarde des données, nous avons pensé à un système d'importation des données avec une tâche planifiée.

2. Précisez les moyens utilisés :

- phpmyadmin
- MySql

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02/01/2023 au : *04/01/2023*

5. Informations complémentaires *(facultatif)*

Activité-type 2

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

- Développer des composants dans le langage d'une base de données - Chappy

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

Certaines informations sont dites "sensibles" comme ce qui permet d'identifier une personne physique de manière unique, ou des informations à caractères personnelles. Dans notre base de donnée nous considérons que le mot-de-passe est une donnée sensible car il permettrait de pouvoir s'authentifier de manière frauduleuse et ainsi accéder à certaines infos et opérations ce qui ne devrait pas être possible.

Pour améliorer la sécurité, nous ne persistons en base les mot-de-passe que sous une forme hachée. Pour ce faire, avant l'ajout ou la modification d'un user, le processus passe par un middleware qui s'occupe entre autres du hachage.

```
/**
 * process before a user persist
 * valorise the fields 'created_at' and 'roles', hash password
 */
const prePersist = async (req, res, next) => {
    try {
        //valorisation de la date de creation
        req.body.createdAt = new Date();

        //hashage du password
        req.body.password = await bcrypt.hash(req.body.password, 10);

        //valorisation du roles si aucun n'est spécifié
        if (!req.body.roles) {
            req.body.roles = [1];
        }

        next();
    } catch (err) {
        console.log(err);
        res.status(500).send('error_during_prePersist_user');
    }
}
```

Pour assurer l'intégrité des données, certains processus de vérification sont mis en place avant toute opération d'insertion ou de modification. Par exemple pour s'inscrire comme nouvel utilisateur, tous les champs du formulaire d'inscription doivent avoir été remplis et sous un format correct, aussi l'identifiant unique (ici l'adresse mail) ne doit pas déjà avoir été utilisé.

```
/**
 * check if all inputs required for register are send in body
 */
const registerInputsAreSent = (req, res, next) => {
    const userFields = req.body;

    if (
        !userFields.email
        || !userFields.prenom
        || !userFields.nom
        || !userFields.password
    ) {
        res.status(406).send('list of needed inputs is required');
        return;
    }

    next();
}

/**
 * check if the email send in body is already used by a user
 */
const userDoesntExists = async (req, res, next) => {
    if (req.body.email) {
        const userAllreadyExists = await UserRepository.exists(req.body.email);

        if (!res.locals.userId) {
            if (userAllreadyExists) {
                res.status(406).send('Error during registration. This email is already used');
                return
            }

        } else {
            if (
                req.body.email
                && userAllreadyExists != null
                && userAllreadyExists.idUser != res.locals.userId
            ) {
                res.status(406).send('Update Failed, this email is already used');
                return;
            }
        }
    }

    next();
}
```

Si les prérequis ne sont pas valides, une exception est levée, l'opération n'est pas effectuée et un message d'erreur compréhensible est envoyé à l'utilisateur pour l'informer de la problématique rencontrée.

2. Précisez les moyens utilisés :

- bcrypt
- Express.js pour les statuts de réponse
- Node pour les middlewares, routes

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02/01/2023 au : *04/01/2023*

5. Informations complémentaires *(facultatif)*

Activité-type 3

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

- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement - Chappy

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

Pour mener à bien notre projet Chappy, nous avons mis en place une collaboration efficace en utilisant une combinaison d'outils de gestion de projet et de développement.

Diagramme de gantt :
Pour planifier et suivre notre progression, nous avons opté pour l'utilisation d'un diagramme de Gantt, qui nous a permis d'avoir une vue d'ensemble des différentes tâches et des délais associés. Le diagramme de Gantt nous a également aidé à identifier les dépendances entre les différentes tâches, ce qui a été crucial pour la planification de notre projet.

Trello :
Pour la gestion quotidienne de nos tâches, nous avons choisi d'utiliser Trello, un outil de gestion de projet en ligne. Trello nous a permis de créer des listes de tâches, des cartes et des étiquettes pour organiser et prioriser nos activités. Grâce à son interface intuitive, nous avons pu suivre facilement l'état d'avancement de chaque tâche, attribuer des responsabilités aux membres de l'équipe et collaborer de manière transparente.

Outils de versionning Git :
En ce qui concerne le contrôle de version et la gestion du code source, Git a été notre outil central. Nous avons utilisé Git pour héberger notre dépôt de code source et gérer les différentes branches de développement. L'utilisation de branches nous a permis de travailler simultanément sur des fonctionnalités distinctes sans affecter la version principale du projet. Les fonctionnalités ont ensuite été fusionnées via des pull requests, permettant une approche collaborative pour la validation et l'intégration du code. Nous avons choisis la méthodologie de versionning **trunk based development** qui consiste à créer une branche pour chaque feature développé.

Wamp server :
Pour la mise en place de notre environnement de développement, nous avons utilisé Wamp (Windows, Apache, MySQL, PHP). Wamp nous a offert un serveur web local pour tester notre application en cours de développement. Il nous a également permis de gérer notre base de données et d'exécuter notre application dans un environnement similaire à celui de production.

Editeur de code (visual studio code) :
Enfin, notre environnement de développement a été enrichi par l'utilisation de Visual Studio Code, un éditeur de code léger et puissant. Nous avons apprécié la richesse de ses fonctionnalités, telles que la coloration syntaxique, l'autocomplétion, ses extensions et l'intégration avec Git. Cela nous a permis de développer de manière productive et efficace tout en maintenant la cohérence et la qualité du code.

Gant.

-
- Trello.
- Vs Code
- Wamp
- Git
- Communication technique oral

2. Précisez les moyens utilisés :

- Lucid Chart pour le MCD et MLD
- WampServer
- Moteur MySql

3. Avec qui avez-vous travaillé ?

Avec mon binome Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02/01/2023 au : *04/01/2023*

5. Informations complémentaires *(facultatif)*

Activité-type 3

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

- Concevoir une application - Chappy

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

- L'accès aux messages et aux infos user sont réservé aux concernés et au admins
- On ne peut accéder à rien si on n'est pas co
- Mettre un exemple d'un bout de doc qu'on a respecté et compris
- L'architecture technique (parlé des responsabilités des différentes couches de l'appli)

Pour concevoir notre projet d'application mobile Chappy, nous nous sommes basés à 100% sur le sujet donné par notre centre de formation, c'était notre cahier des charges. Concernant le développement de l'api, il y était stipulé 3 types de routes dans le sujet :

- Les routes ouvertes à toutes
- Les routes accessibles uniquement aux users identifiés
- Les routes accessibles uniquement aux admins

Il nous fallait donc une gestion des rôles solides pour pouvoir répondre à ce besoin de sécurité.
Pour illustrer le développement de cette partie, j'ai décidé de prendre en exemple la partie d'accès aux messages privés et aux informations de l'utilisateur.

<u>Les messages privés :</u>
Les utilisateurs de l'application ont la possibilité de consulter les messages privés qu'ils ont avec les différents utilisateurs de l'application, ils peuvent donc consulter uniquement leurs propres messages privés, tandis que les administrateurs de l'application ont la possibilité d'accès à tous les messages privées de l'application.
Pour assurer cela nous avons donc deux routes : /getdiscussion/:idUserReceiver et /getall/usermessage.

middleware appelée par la route accessible par l'admin:

```
/**
 * retourne tout les UserMessages
 * GET http://127.0.0.1:3000/admin/getall/usermessage'
 */
const getAllUserMessages = async (req, res, next) => {
    try {
        const privateMessages = await MessageUserRepository.getAll();

        res.status(200);
        res.send(privateMessages);
        next();

    } catch (err) {
        console.log(err);
        res.status(500).send('error during getting all private messages');
    }
}
```

middleware appelée par la route accessible par l'utilisateur :

```
/**
 * get all message from salon
 * GET http://127.0.0.1:3000/messageuser/getdiscussion/:idUserReceiver/
 */
const getDiscussion = async (req, res, next) => {
    try {
        const idUserSender = res.locals.authentifiedUser.idUser;
        const idUserReceiver = req.params.idUserReceiver;

        const wheres = [
            {fk_id_user_sender: {[Op.in]: [idUserReceiver, idUserSender]}},
            {fk_id_user_receiver: {[Op.in]: [idUserReceiver, idUserSender]}}
        ]

        if (idUserSender != idUserReceiver) wheres.push({fk_id_user_sender: {[Op.not]: Sequelize.col('fk_id_user_receiver')}});

        let messagesUser = await MessageUserRepository.getDiscussion({[Op.and]: wheres});

        res.status(200);
        res.send(messagesUser);
        next();

    } catch (err) {
        console.log(err);
        res.status(500).send('error during getting private discussion');
    }
}
```

Les infos de l'utilisateur :

Les utilisateurs de l'application ont la possibilité de consulter le profil d'un utilisateur s' il le souhaite mais ils n'auront accès qu'au nom et au prénom de la personne. Tandis que l'admin aura lui la possibilité d'avoir accès à l'email et à la date de création du profil en question. La route qui assure cela est /detail :

```
/**
 * send informations of a user by id
 * GET http://127.0.0.1:3000/user/auth/:idUser/detail
 */
router.get('/detail', async (req, res, next) => {
    try {
        let user;
        if (await UserRepository.isAdmin(res.locals.authentifiedUser)) {
            user = await UserRepository.getNestedFilteredByFilters({pk_id_user: res.locals.userId});
        } else {
            user = await UserRepository.getFilteredById(res.locals.userId);
        }

        user = getUsersState([user])[0];

        res.status(200);
        res.send(user);
        next();
    } catch (err) {
        console.error(err);
        res.status(500).send('error during getting user detail');
    }
});
```

A part les routes register et login qui sont nécessaire à l'inscription et connexion d'un utilisateur , aucunes des routes de l'application sont accessibles pour les personnes non connectées, le middleware "isAuthentified" permet d'assurer cette fonction :

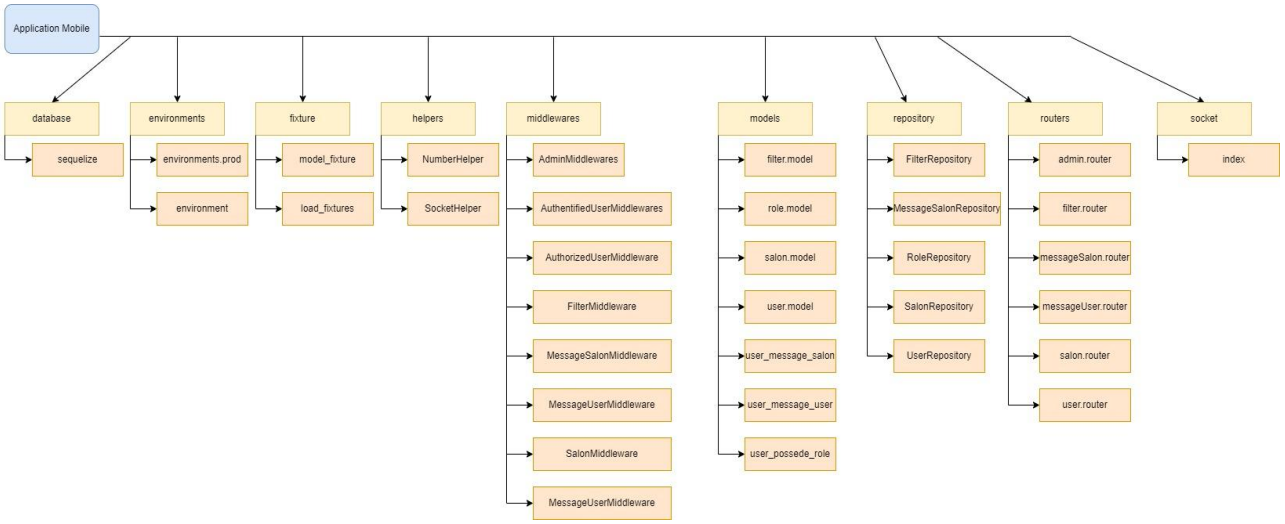
```
/**
 * check if user is authentified using jwt token
 * set the user object returned, in the res.locals
 */
const IsAuthentified = async (req, res, next) => {
    try {
        const authentifiedUser = await UserRepository.getAuthentifiedUser(req);

        if (!authentifiedUser) {
            res.status(401).send('not_authentified');
            return;
        }

        res.locals.authentifiedUser = authentifiedUser;
        next();

    } catch(err) {
        console.log(err);
        res.status(500).send('error_during_authentification_check');
    }
}
```

En effet, celui-ci vérifie et décode le jwt de l'emetteur pour récupérer l'utilisateur par son id ainsi que ses rôles.

Pour garantir une utilisation efficace de mon backend à la fois pour mon application web et mobile, j'ai divisé ma logique en modules distincts, ce qui a amélioré la lisibilité du code et facilite sa maintenance pour les futures versions. J'ai également évité les redondances en créant des

fonctions et des services. Ainsi, la structure de mon code est conçue pour être facilement maintenable et réutilisable.

Arborescence



J'ai suivi une architecture n-tier. Le principe de cette architecture est la séparation des préoccupations pour éloigner la logique métier des routes de l'API.

Les différentes couches de l'application sont séparées :
- Le routeur,
- Middleware qui gère aussi la couche métier de l'application
- La couche data (les modèles, et repository).

Mon back end est donc composer des dossiers suivants :

- **database:** permet d'accéder au fichier database.js qui gère la connexion à la base de données, pour éviter des surcharges de mémoire nous avons décider d'y incorporer à celui-ci un Singleton qui permet de créer une seul instance de connexion
- **fixture:** permet de créer les fixtures de l'application en utilisant sequelize donc en accord avec model.
- repository : permet d'accéder et d'insérer des données en base.
- **helpers :** permet d'accéder au fichier de helper qui héberge de la logique utilisé à divers endroit de l'application, donc ce fichier centralise de la logique utilisé plusieurs fois.
- **Middleware :** contient les fichiers avec les middlewares des routes, les middlewares contient aussi la logique métier de l'application.
- modèle : contenant les modèles de toutes les tables.

- routes : contenant toutes les routes de l'application.
- **socket :** gère le socket de l'application en y faisant la connexion et en y définissant des events sockets.

Fonctionnement de l'api

Lorsque le client envoie une requête sur mon API, le fichier server.js analyse l'url et renvoie sur le router correspondant,ces routes font appeler des middlewares qui vont exécuter la logique métier qui elle même appelle les repositories pour pouvoir insérer modifier et accéder à la données, les middlewares contiennent aussi un bloc try catch qui permet de renvoyer une réponse en fonction de l'état de la requête, cette réponse est au format JSON .

Pour pouvoir utiliser les fonctionnalités de relation entre les entités de sequelize ,nous nous sommes intéressées à la documentation de celui-ci. Elle montre 5 types de relations possibles :

Relation **hasOne** : La relation hasOne représente une association un-à-un entre deux tables, où chaque enregistrement dans la table source (le modèle d'origine) est associé à un seul enregistrement dans la table cible (le modèle associé).

Relation **belongsTo** : La relation belongsTo est l'inverse de hasOne. Elle représente une association un-à-un où chaque enregistrement de la table source appartient à un seul enregistrement de la table cible.

Relation **hasMany** : La relation hasMany représente une association un-à-plusieurs (ou un-à-plusieurs) entre deux tables. Cela signifie qu'un enregistrement du modèle source peut être associé à plusieurs enregistrements du modèle cible.

Relation **belongsToMany** : La relation belongsToMany représente une association plusieurs-à-plusieurs entre deux tables. Cela signifie que chaque enregistrement de la table source peut être associé à plusieurs enregistrements de la table cible, et vice versa. Cela est généralement mis en œuvre en utilisant une table de liaison qui enregistre les correspondances entre les deux tables.

Documentation de sequelize : https://sequelize.org/docs/v6/core-concepts/assocs/

Comprendre et utiliser cela était important étant données que le projet utilise une base de données relationnelles, définir des relations directement dans les modèles sequelize permet de donc d'automapper directement les données des relations dans celui-ci ci-besoin.

Par exemple avec le modèle user_message_salonModel :

```
const Sequelize = require("sequelize");

module.exports = (sequelize) => {

    const salonModel = require("./salon.model")(sequelize);
    const userModel = require("./user.model")(sequelize);

    var user_message_salonModel = sequelize.define("user_message_salon", {
        idUserMessageSalon: {
            primaryKey: true,
            autoIncrement: true,
            type: Sequelize.NUMBER,
            field: 'pk_id_salon_message',
            allowNull: false
        },
        content: {
            type: Sequelize.STRING,
            allowNull: false
        },
        createdAt: {
            type: Sequelize.DATE,
            allowNull: false,
            field: 'created_at'
        },
        idUser: {
            type: Sequelize.INTEGER,
            allowNull: false,
            field: 'fk_id_user'
        },
        idSalon: {
            type: Sequelize.INTEGER,
            allowNull: false,
            field: 'fk_id_salon'
        }
    }, {
        tableName: 'user_message_salon',
        updatedAt: false
    });

    user_message_salonModel.belongsTo(userModel, {
        foreignKey: {
            name : 'idUser',
            field: 'fk_id_user'
          },
          as: 'userSender'
    });

    user_message_salonModel.belongsTo(salonModel, {
        foreignKey: {
            name:'idSalon',
            field: 'fk_id_salon'
          },
          as: 'salon'
    });

    return user_message_salonModel;
};
```

2. Précisez les moyens utilisés :

- Visual studio code
- WampServer
- Moteur MySql
- Documentation de sequelize

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02/01/2023 au : *04/01/2023*

5. Informations complémentaires *(facultatif)*

Activité-type 3

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

- Développer des composants métier - Chappy

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

La couche métier de l'application Chappy est composée d'un type de middleware nommé "Application level".

Les Application level Middlewares s'occupent d'appeler des fonctions dans les repositories pour faire des requêtes à la base de données au besoin, définir le statut de la réponse retournée et exécute un try catch pour renvoyer en cas d'erreur un message digeste pour l'utilisateur.

Pour donner un exemple visuel, voici le middleware MessageSalonMiddleware:

```
 */
const getDiscussion = async (req, res, next) => {
    try {
        const messagesSalon = await MessageSalonRepository.getDiscussion({fk_id_salon: req.params.idSalon});

        res.status(200);
        res.send(messagesSalon);
        next();
    } catch (err) {
        console.log(err);
        res.status(500).send('error during getting room messages');
    }
}

const getAllMessage = async(req,res,next) => {
    try{
        const getAllMessage = await MessageSalonRepository.getAllMessage();
        res.status(200);
        res.send(getAllMessage);
        next();
    }catch(err){
        console.log(err)
        next(err);
    }
}
```

Pour assurer le côté défensif de nos middleware, la validité des données est importantes afin de s'assurer que les données récoltées sont celles attendues.

Le middleware registerInputsValidations permet de vérifier que les inputs sont bien remplis et check si l'email est au bon format, si ce n'est pas le cas une erreur 406 est retourné.

```
/**
 * pass middleware if register inputs are valid
 */
const registerInputsValidation = (req, res, next) => {
    const userFields = req.body;

    if (
        !userFields.email
        || !userFields.prenom
        || !userFields.nom
        || !userFields.password
    ) {
        res.status(406).send('list of needed inputs is required');
        return;
    }

    const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
    if (emailRegex.test(email)) {
        res.status(406).send('the email address is not in the correct format');
        return;
    }

    next();
}
```

Pour les tests unitaires de l'API nous avons utilisé Mocha qui est est un framework de test JavaScript largement utilisé, ainsi que Chai qui est aussi une librairie qui fournit une syntaxe expressive et flexible pour effectuer des assertions.
Le principe Arrange-Act-Assert (Mise en place, Agir, Vérification) est une convention de nommage et d'organisation pour écrire des tests unitaires. La structure AAA permet de séparer clairement les différentes parties du test, ce qui facilite la compréhension du comportement testé et des attentes.

Exemple de test unitaire, sur l'obtention d'un user d'après un certain numéro d'identifiant:

```
describe('getById', () => {
  it('should return correct user depending of given id', async () => {

    //ARRANGE
    const USER_ID = 1;

    //ACT
    let result = await UserRepository.getById(USER_ID);

    //ASSERT
    expect(result).to.be.an('object');
    expect(result).to.have.property('idUser', 1);
    expect(result).to.have.property('email', 'user@gmail.com');
    expect(result).to.have.property('prenom', 'user');
    expect(result).to.have.property('nom', 'user');
    expect(result).to.have.property('createdAt', '2022-05-05');

  });

});
```

2. Précisez les moyens utilisés :

- Lucid Chart pour le MCD et MLD
- WampServer pour pouvoir suler phpmyadmin
- Moteur MySql

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ► Du : 02/01/2023 au : *04/01/2023*

5. Informations complémentaires *(facultatif)*

Activité-type 3

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

- Préparer et exécuter les plan de test d'une application

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

```
PS D:\VisualStudio Repo\LAPLATEFORME\appmobilechat\api-app-mobile-chat> npm test

> api-node-js@0.0.0 test
> mocha test/**/*.spec.js



  getAll
    1) should return all users


  13 passing (37ms)
  22 failing

  1) getAll
       should return all users:
     AssertionError: expected { idUser: 1, …(5) } to have property 'idUser' of '1', but got 1
      at Context.<anonymous> (test\usermiddlewares.spec.js:17:28)
      at process.processTicksAndRejections (node:internal/process/task_queues:95:5)
```

```
PS D:\VisualStudio Repo\LAPLATEFORME\appmobilechat\api-app-mobile-chat> npm test

> api-node-js@0.0.0 test
> mocha test/**/*.spec.js



  getById
    ✓ should return correct user depending of given id (57ms)
    …

  45 passing (65ms)
```

2. Précisez les moyens utilisés :

- Mocha
- Chai
- terminal

3. Avec qui avez-vous travaillé ?

Avec mon binôme Basil Collette

4. Contexte

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

Période d'exercice ▸ Du : 02/01/2023 au : *04/01/2023*

5. Informations complémentaires *(facultatif)*

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] Cliquez ici pour taper du texte. ,

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 Cliquez ici pour taper du texte. le Cliquez ici pour choisir une date

pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

ANNEXES

(Si le RC le prévoit)