



DOSSIER PROFESSIONNEL (DP)

Nom de naissance - MAGASSOUBA
Nom d'usage - MAGASSOUBA
Prénom - Ahmed Sékou
Adresse - 105 rue de la république , 13002 Marseille

Titre professionnel visé

Concepteur Développeur Informatique

MODALITÉ D'ACCÈS :

- 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.

DOSSIER PROFESSIONNEL (DP)

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

| | | |
|---|----|----|
| Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité | p. | 5 |
| - <i>Développement d'une application de chat-SENT</i> | p. | 5 |
| - <i>My Sokoban</i> | p. | 18 |
| Concevoir et développer la persistance des données en intégrant les recommandations de sécurité | p. | 22 |
| - <i>Développement d'une application de chat-SENT</i> | p. | 22 |
| - <i> </i> | p. | |
| - <i> </i> | p. | |
| Concevoir et développer une application multicouches répartie en intégrant les recommandation de sécurité | p. | 30 |
| - <i>Développement d'une application de chat-SENT</i> | p. | |
| - Intitulé de l'exemple n° 2 | p. | |
| - Intitulé de l'exemple n° 3 | p. | |
| Titres, diplômes, CQP, attestations de formation (<i>facultatif</i>) | p. | |
| Déclaration sur l'honneur | p. | |
| Documents illustrant la pratique professionnelle (<i>facultatif</i>) | p. | |
| Annexes (<i>Si le RC le prévoit</i>) | p. | |

DOSSIER PROFESSIONNEL ^(DP)

**EXEMPLES DE PRATIQUE
PROFESSIONNELLE**

DOSSIER PROFESSIONNEL (DP)

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

Exemple n°1 - Développement d'une application de chat-SENT

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

Dans le cadre de ma formation de développeur et concepteur d'application, et ma présentation de fin d'année, j'ai réalisé un projet regroupant toutes les compétences que j'ai pu acquérir durant mon cursus. Le projet est un chat sous forme d'application mobile, ce projet semblait très complet aussi bien dans la conception que dans le développement.

Sent est une app de chat mobile avec les fonctionnalités suivantes pour les utilisateurs

- Création d'un profil qui peut être modifié par l'utilisateur
- Envoie de message sur un channel public commun
- Envoie de message sur un channel privé
- Une annuaire avec possibilité de rechercher un utilisateur particulier
- un panel admin pour gérer les users et les messages

À la suite de la rédaction du cahier des charges, j'ai commencé par me renseigner sur le type de technologie à utiliser pour mener à bien ce projet. Je voulais une technologie que je maîtrisais un peu. Ainsi mon choix s'est porté sur Node.js qui est un environnement d'exécution de JavaScript côté serveur et Express qui est un Framework de node.js, vu lors de notre formation.

Node dispose d'une bibliothèque appelée Sequelize, c'est un ORM (mapping object-relational mapping) qui est un type de programme informatique qui se place en interface entre le programme applicatif et une base de données relationnelle pour simuler une base de données orientées objet.

Ensuite, j'ai réalisé le diagramme de cas d'utilisation. Qui permet de donner une vision globale du comportement fonctionnel de notre application mobile.

Au début du projet, le logo SENT et le splash screen ont été créés par notre équipe en utilisant Midjourney .

Après quelques navigations rapides sur les différents sites de type chat mobile , nous avons vu ce qui se faisait , ce que nous pouvions apporter de nouveau .

Nous avons décidé du style à donner à notre application couleur et font.

DOSSIER PROFESSIONNEL (DP)



Charte Graphique Sent

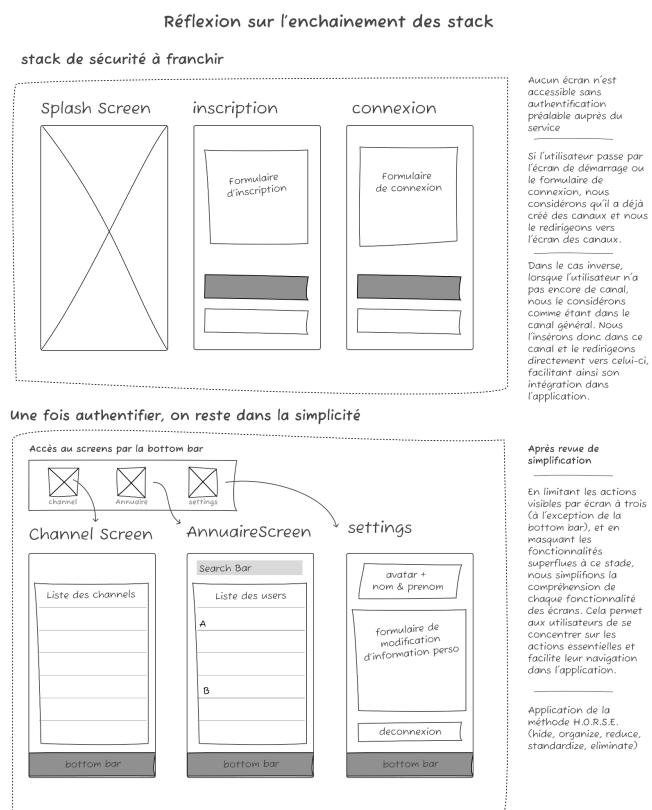
Couleurs

| | | | |
|--|---------|--|---------|
| ■ | #00C2FF | ■ | #A8B500 |
| ■ | #5B1B9B | ■ | #FFA022 |
| ■ | #000000 | | |

Police

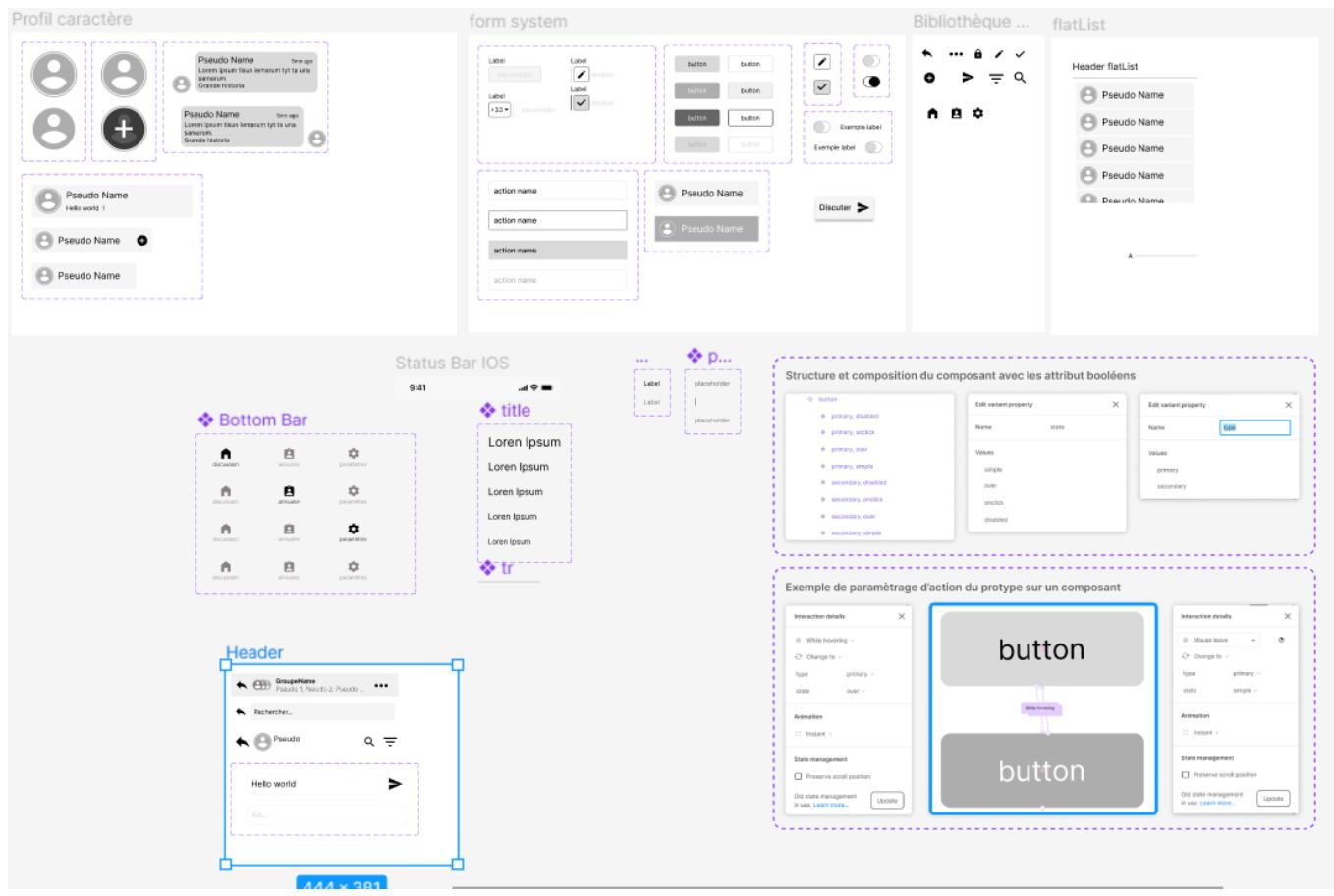
ROBOTO

Une fois tous ces éléments validés, j'ai mis en place ces éléments dans l'application pour proposer des enchaînements de maquettes :



DOSSIER PROFESSIONNEL (DP)

WIREFRAME: Design système et Maquette



DOSSIER PROFESSIONNEL (DP)

pad -> phone

pad -> text

pad -> edit text

Splash Screen

Inscription

connexion

Home

Profil

annuaire

public chat

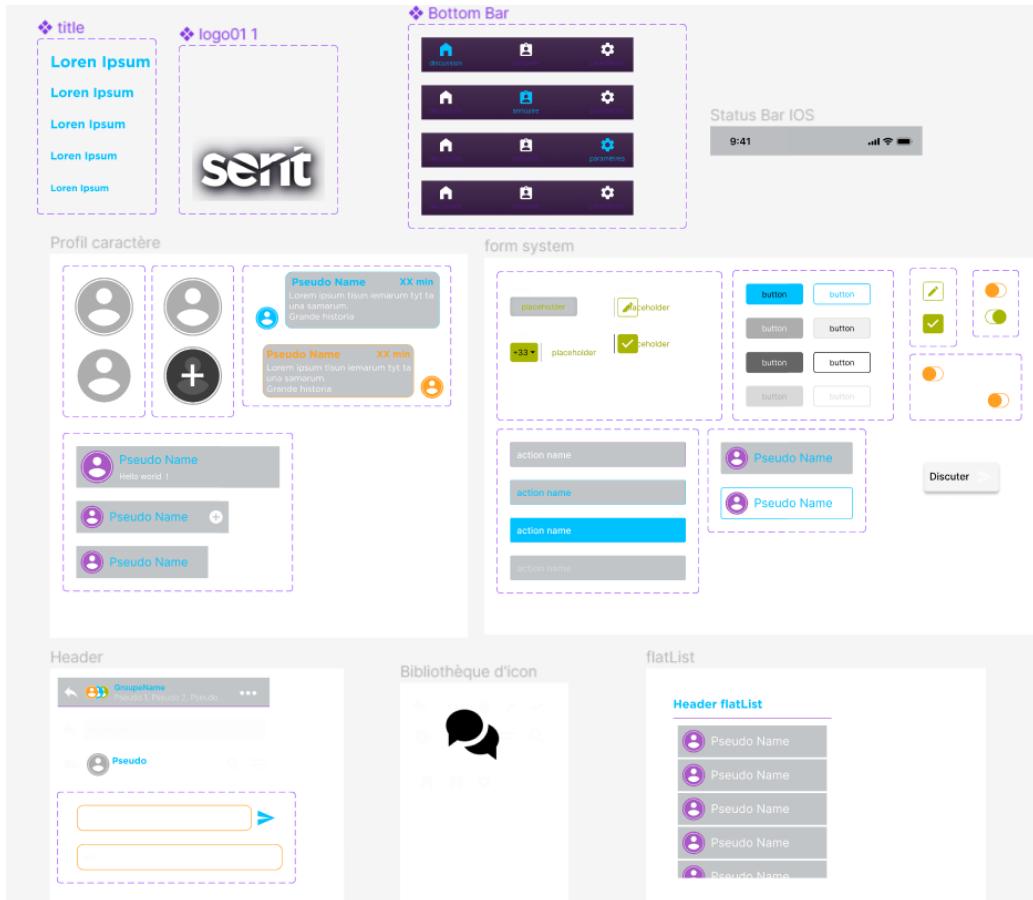
chat one by one

chat group

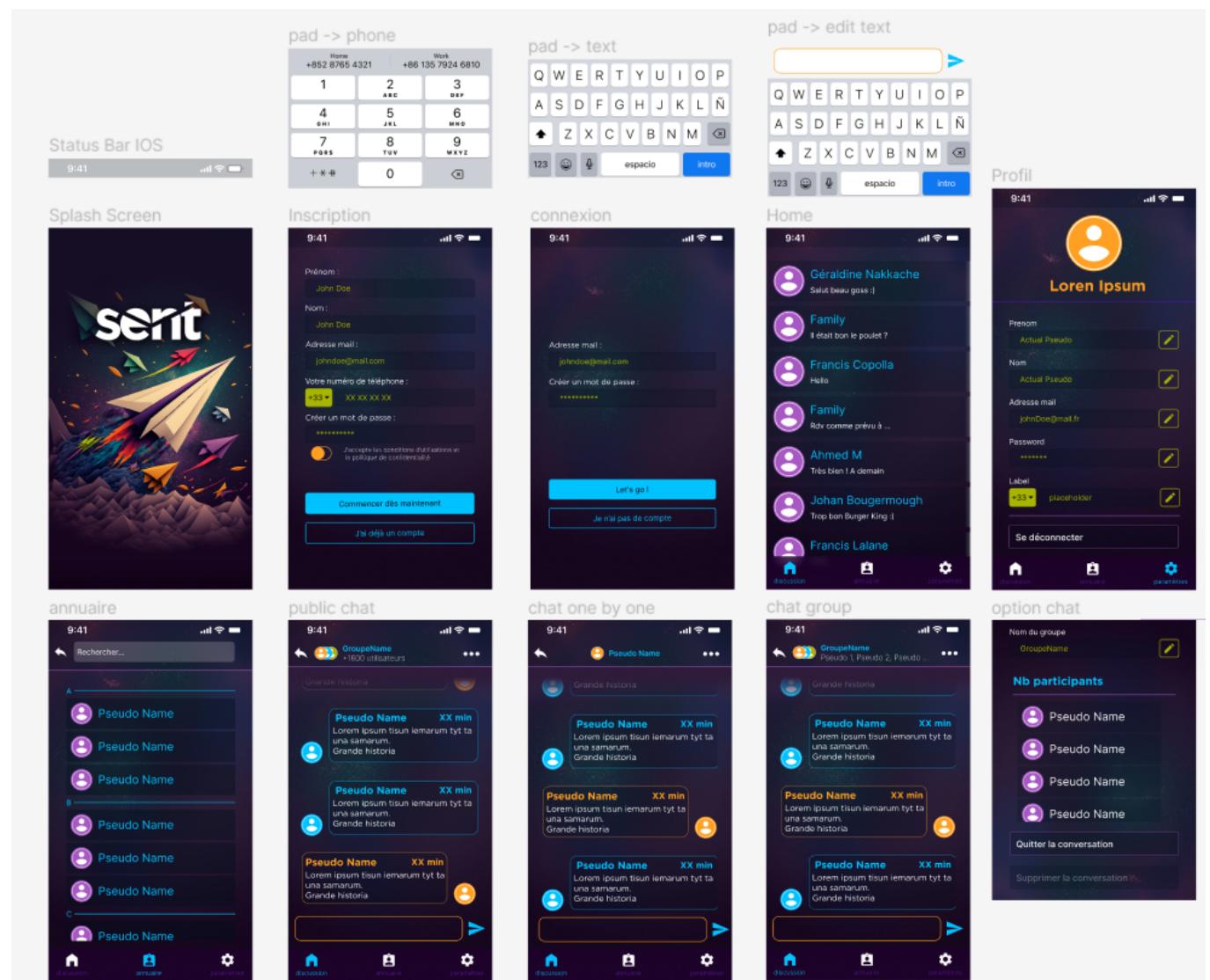
option chat

DOSSIER PROFESSIONNEL (DP)

Haute fidélité: Design système et maquette



DOSSIER PROFESSIONNEL (DP)



Lors du développement d'une application Node.js avec Sequelize, les composants d'accès aux données sont responsables de la gestion de l'interaction entre l'application et la base de données relationnelle. Sequelize facilite cette tâche en fournissant une couche d'abstraction ORM qui permet de manipuler les données à l'aide d'objets JavaScript plutôt que d'écrire des requêtes SQL manuellement.

Pour commencer, il est nécessaire d'installer Sequelize et les dépendances associées via NPM (Node Package Manager). Cela peut être fait en exécutant la commande suivante dans le terminal :

```
npm install sequelize sequelize-cli mysql2
```

Une fois Sequelize installé, la première étape consiste à configurer la connexion à la base de données.

DOSSIER PROFESSIONNEL (DP)

Pour configuration de la connexion à la base de donnée nous avons spécifié les détails de connexion tels que l'hôte, le port, le nom de la base de données, le nom d'utilisateur et le mot de passe.

```
/*
 * CONNEXION A LA BASE DE DONNEES *
 */

let sequelize = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  "", //process.env.DB_PASSWORD_MACOS,

  {
    host: process.env.DB_HOST,
    dialect: "mysql",
    dialectOptions: {
      charset: "utf8mb4",
    },
    logging: false,
  }
);
```

Ensuite, nous avons créé des modèles qui correspondent aux tables de la base de données. Les modèles définissent la structure de chaque table et les relations entre les tables, si nécessaire. Vous pouvez utiliser la ligne de commande Sequelize (sequelize-cli) pour générer les fichiers de modèle correspondants à partir de la ligne de commande ou les créer manuellement en utilisant la syntaxe Sequelize.

Dans chaque modèle, nous avons défini les attributs correspondant aux colonnes de la table et les associations avec d'autres modèles, le cas échéant. Nous avons spécifié les types de données, les contraintes, les relations "one to one", "one to many" ou "many to many", ainsi que d'autres options de configuration spécifiques à Sequelize.

DOSSIER PROFESSIONNEL (DP)

```
*****  
/* MIS EN PLACE DES RELATIONS */  
*****  
  
const db = {};  
db.sequelize = sequelize;  
db.User = Users(sequelize);  
db.Channel = Channels(sequelize);  
db.Participant = Participants(sequelize);  
db.Conversation = Conversations(sequelize);  
  
db.UserhasMany(db.Participant, {  
  foreignKey: "userId",  
  // onDelete: "cascade",  
  // onUpdate: "cascade",  
  // hooks: true,  
});  
db.Participant.belongsTo(db.User, {  
  foreignKey: "userId",  
});  
  
db.ChannelhasMany(db.Participant, {  
  foreignKey: "channelId",  
  //  // onDelete: "cascade",  
  //  // onUpdate: "cascade",  
  //  // hooks: true,  
});  
db.Participant.belongsTo(db.Channel, {  
  foreignKey: "channelId",  
});  
  
db.ParticipanthasMany(db.Conversation, {  
  foreignKey: "participantId",  
  //  // onDelete: "cascade",  
  //  // onUpdate: "cascade",  
  //  // hooks: true,  
});  
db.Conversation.belongsTo(db.Participant, {  
  foreignKey: "participantId",  
});      You, il y a 5 mois * fix datafixture for arll entity
```

Le code ci dessous vérifie si la connexion à la base de donnée c'est effectué avec succès

```
*****  
/* CONNEXION A LA BDD ET LANCEMENT DU SERVER */  
*****  
  
Db.sequelize  
  .authenticate()  
  .then(() => {  
    console.log("Connection has been established successfully...");  
  })  
  .then(() => {  
    console.log("Synchronizing models with database...");  
    app.listen(process.env.APP_PORT, () => {  
      console.log(  
        `Server is running on port ${process.env.APP_PORT} : http://localhost:${process.env.APP_PORT}`  
      );  
    });  
  })  
  .catch((err) => {  
    console.log("Unable to connect to the database...");  
    console.log(err);  
  });      You, il y a 6 mois * setup database ...
```

DOSSIER PROFESSIONNEL (DP)

Une fois les modèles définis, nous pouvons utiliser les méthodes fournies par Sequelize pour effectuer des opérations CRUD sur la base de données. Par exemple, vous pouvez créer un nouvel enregistrement, le lire, le mettre à jour ou le supprimer en utilisant les méthodes correspondantes sur le modèle associé.

Voici un exemple d'utilisation de Sequelize pour récupérer un utilisateur:

Dans l'exemple ci-dessous, grâce à Sequelize qui fournit des méthodes et des opérations simples pour effectuer les opérations CRUD (Create, Read, Update, Delete) sur les données de la base de données, on peut utiliser l'une de ces méthodes le `findById` qui comme son nom l'indique prend en paramètre un identifiant et retourne l'utilisateur qui correspond.

```
/*
 *      GET ONE USER
 */
const getOne = async (req, res) => {
  const { userId } = req.params;
  Users.findById(userId, {
    attributes: [
      exclude: [
        "password",
        "createdAt",
        "updatedAt",
        "deletedAt",
        "phone",
        "email",
        "role",
      ],
    ],
  })
    .then((user) => {
      if (user) {
        const message = "Utilisateur trouvé";
        return res.status(200).json({ message, data: user });
      } else {
        const message = "Utilisateur non trouvé";
        res.status(404).json({ message, data: user });
      }
    })
    .catch((err) => {
      const message = "Erreur serveur. Veuillez réessayer ultérieurement";
      res.status(500).json({ message, data: err });
    });
};
```

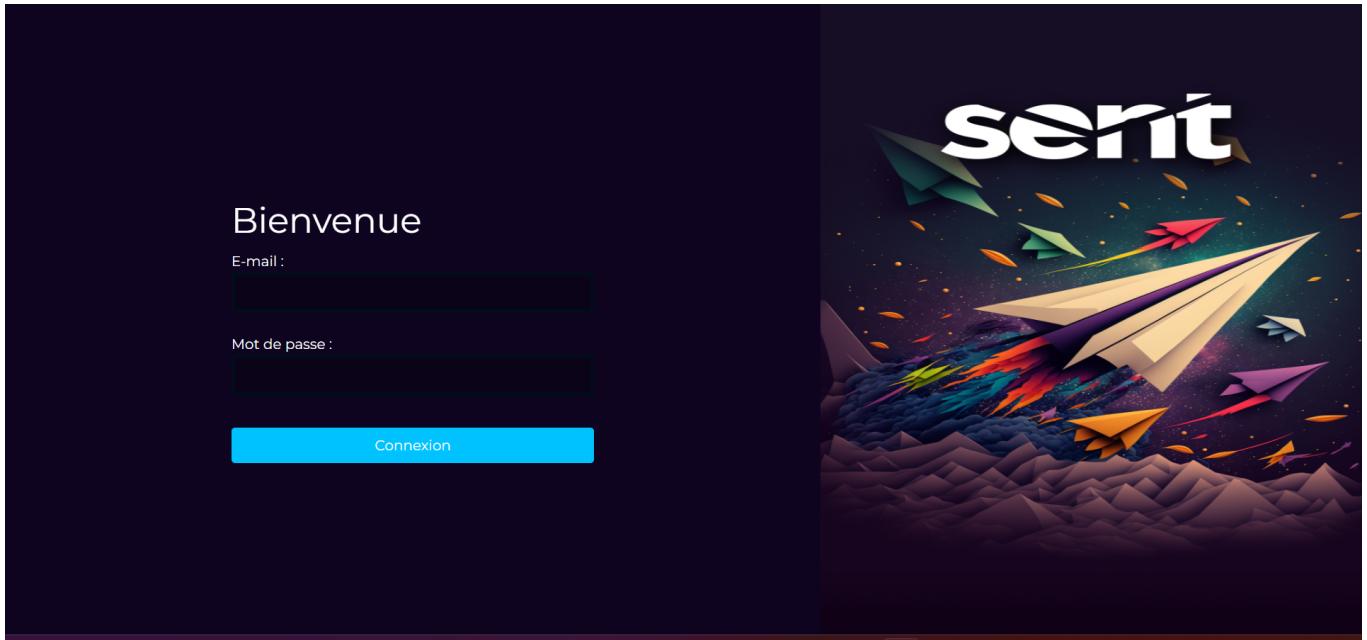
Dans cet exemple on a la méthode de récupération d'un utilisateur (`user`), on y spécifie les données qu'on veut récupérer dans la table `users` et on traite le renvoie de la réponse au client selon le cas.

L'exemple ci-dessus correspond à l'end-point appelé pour la interface utilisateur.

Il est important de noter que Sequelize offre de nombreuses autres fonctionnalités telles que les jointures, les transactions, les migrations, les validations des données, etc. Ces fonctionnalités peuvent être utilisées pour construire des composants d'accès aux données plus avancés et répondre aux

DOSSIER PROFESSIONNEL (DP)

besoins spécifiques de votre application.



L'utilisateur interagit avec l'interface de connexion pour le panel administrateur. Une fois les champs remplis, il clique sur le bouton connexion.

connexion de l'utilisateur :

Lorsque l'utilisateur interagit avec l'interface de connexion, la méthode handleSubmit est appelée pour gérer les données soumises. Cette méthode effectue des vérifications sur les champs du formulaire à l'aide de la bibliothèque validator. Elle vérifie notamment si les champs sont correctement remplis et si le format de l'e-mail est valide.

```
const handleSubmit = () => {
  const errors = {};

  if (isEmpty(email)) {
    errors.email = "* Veuillez remplir le champ E-mail";
  } else if (!isEmail(email)) {
    errors.email = "* Veuillez saisir une adresse valide";
  }

  if (isEmpty(password)) {
    errors.password = "* Veuillez remplir le champ password";
  }

  setFormErrors(errors);

  if (Object.keys(errors).length === 0) {
    login(email, password);
  }
}
```

Si aucune erreur n'est détectée dans les champs du formulaire, la méthode login est exécutée.

DOSSIER PROFESSIONNEL (DP)

```
const login = async (email1, password1) => {
  const data = {
    email: email1,
    password: password1,
  };
  const auth = await signInApi.signIn(data);
  console.log(auth);
  const { status } = auth;
  const { token } = auth.data;
  const { message } = auth.data;

  if (status === 200 && auth.data.data.role === "ADMIN") {
    localStorage.setItem("user", token);
    navigate("/orders");
  }
  else {
    console.log("Nope :) =>", message);
    setError(message);
  }
};

const signIn = async (data1) => {
  console.log(data1);
  const data = await axios
    .post(`http://localhost:3000/api/users/authenticate`, data1)
    .catch((err) => {
      return err.response;
    });
  return data;
};

export default { signIn };
```

Cette méthode est responsable d'appeler le service approprié pour exécuter la requête nécessaire afin de connecter l'utilisateur. Cela implique l'appel de l' API backend et l'exécution d'une requête vers la base de données.

Une fois que l'utilisateur est connecté avec succès, il peut être redirigé vers la page d'accueil de l'application.

2. Précisez les moyens utilisés :

❖ Figma

Figma est un outil de conception d'interface utilisateur (UI) et de prototypage collaboratif basé sur le cloud. Il offre une plateforme complète pour la conception, la collaboration et le partage de maquettes d'interfaces utilisateur interactives.

❖ Environnement de développement intégré (IDE) **Visual Studio Code**

Visual Studio Code est un éditeur de code source populaire, offrant une interface utilisateur intuitive, une prise en charge de nombreux langages, une extensibilité grâce aux extensions, une intégration native avec Git, des fonctionnalités de débogage avancées, un terminal intégré et des outils de productivité pour les développeurs.

DOSSIER PROFESSIONNEL (DP)

❖ SQL

SQL est utilisé dans une grande variété d'applications et de systèmes de gestion de bases de données (SGBD), tels que MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite, etc. Il est essentiel pour stocker, récupérer et manipuler efficacement les données dans les bases de données relationnelles.

❖ MySql

MySQL est largement utilisé dans de nombreuses applications et systèmes, allant des sites web et des applications d'entreprise aux systèmes de gestion de contenu (CMS) tels que WordPress. En tant que SGBDR populaire et open source, MySQL offre une performance élevée, une stabilité et une flexibilité pour la gestion des données.

❖ Node.js

Node.js est un environnement d'exécution côté serveur basé sur le moteur JavaScript V8 de Google Chrome. Il permet d'exécuter du code JavaScript côté serveur, contrairement à son utilisation traditionnelle côté client dans les navigateurs web.

❖ Express

Express est un framework web minimaliste pour Node.js qui facilite le développement d'applications web côté serveur. Il est conçu pour être simple, flexible et performant.

❖ React Native

React Native est un framework open source développé par Facebook qui permet de créer des applications mobiles pour iOS et Android en utilisant le langage JavaScript et la bibliothèque React.

❖ Expo

Expo est une plateforme de développement qui fournit des outils et des services pour créer des applications mobiles avec React Native.

❖ Documentation

→ Documentation officiel react native

<https://reactnative.dev/>

→ Documentation officiel expo

<https://expo.dev/>

→ Documentation officiel NodeJs

<https://nodejs.org/en/docs>

→ Documentation officiel Express

<https://expressjs.com/fr/>

→ Le routage avec Express

<https://expressjs.com/fr/guide/routing.html>

→ Intégration de bases de donnée avec ExpressJs

<https://expressjs.com/fr/guide/database-integration.html>

DOSSIER PROFESSIONNEL (DP)

Ce projet m'a permis d'acquérir les compétences du référentiel:

- Maquetter une application
- Développer les composant d'accès au données
- Développer la partie front-end d'une interface utilisateur web
- Développer la partie back-end d'une interface utilisateur web

3. Avec qui avez-vous travaillé ?

Boris TIKHOMIROFF

Johan Bouguermouh

Cyril Porez

4. Contexte

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

Chantier, atelier, service ▶ *Sent*

Période d'exercice ▶ Du : 02/01/2023 au : 27/01/2023

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

Exemple n°2 - My Sokoban

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

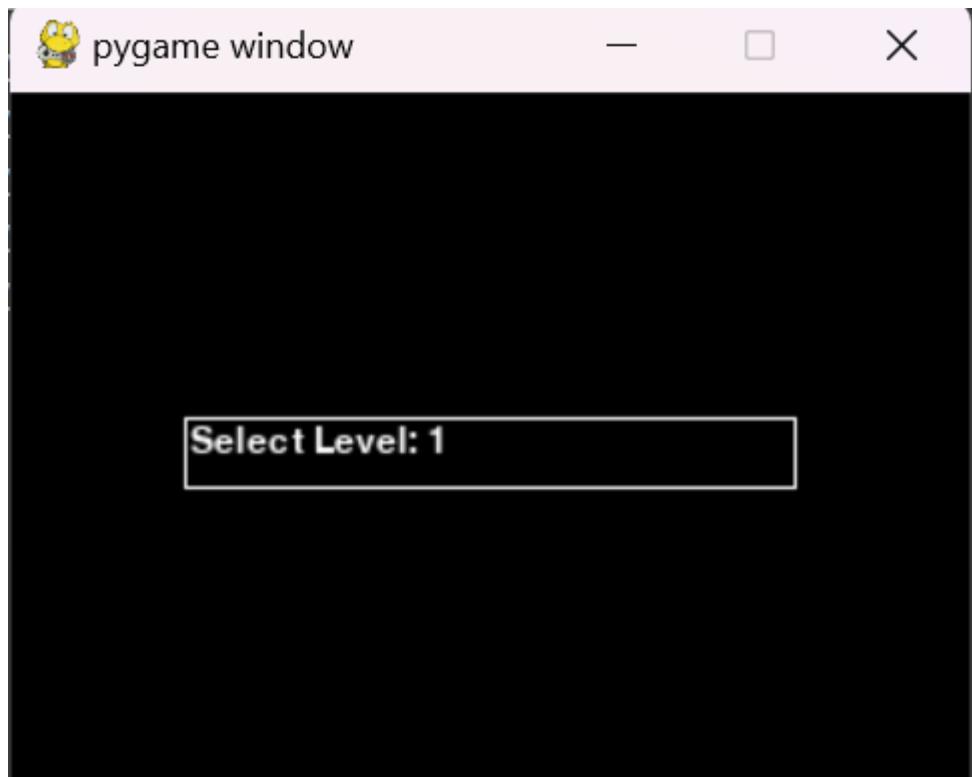
Dans le cadre de ma formation de développeur et concepteur d'application, et ma présentation de fin d'année, j'ai dû réaliser une application de type desktop. Le projet My Sokoban est un jeu où le contexte est de déplacer des caisses jusqu'à l'endroit désigné sur la map.

Pour réaliser ce projet j'ai utilisé les technos suivantes:

- ❖ **Python** pour programmer l'appli
- ❖ **Pygames** pour l'interface graphique

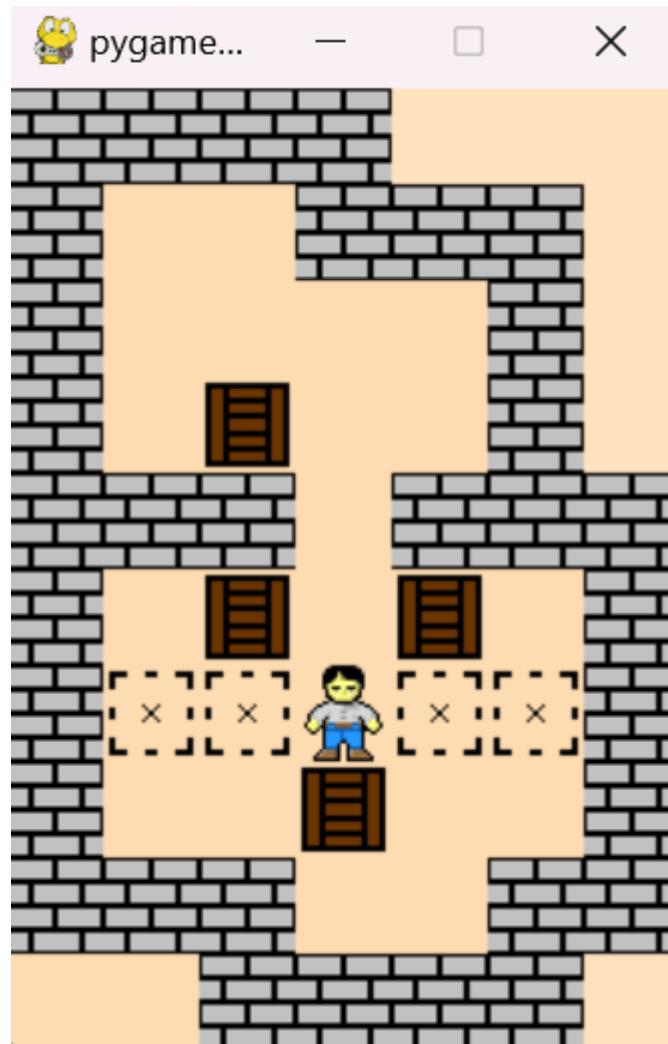
Pour ce faire, il fallait créer une architecture avec un modèle de donnée (une classe). Dans cette classe j'ai créé des fonctions qui gèrent les événements de clavier, le déplacement des caisses et du personnage, la création de la map en fonction du niveau choisi.

Lorsque l'on démarre l'appli on doit choisir un niveau entre 1 et 50:



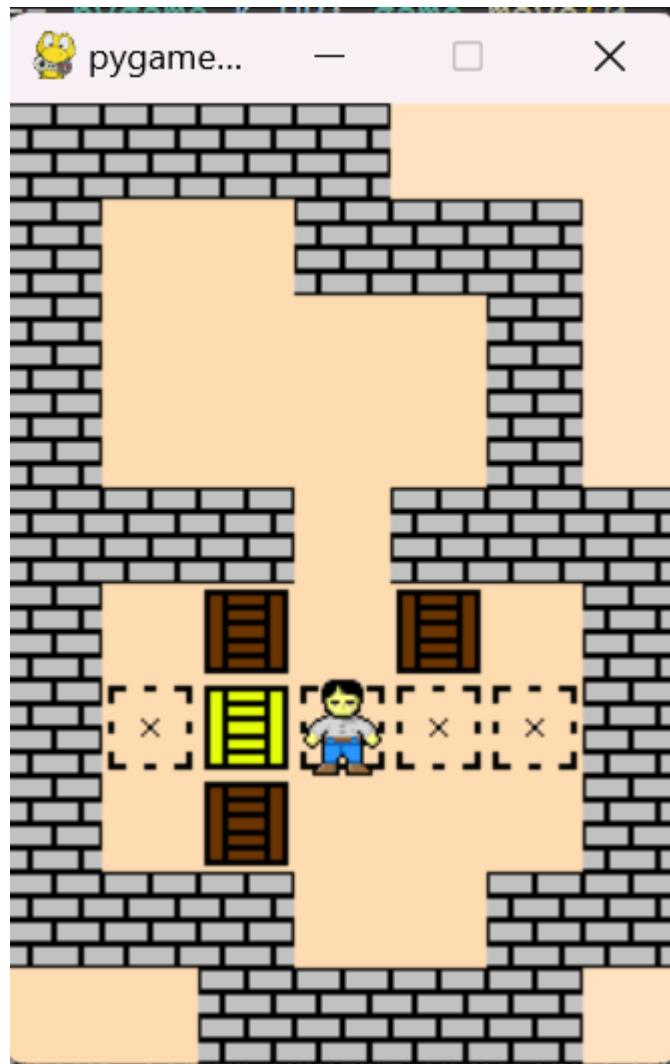
Ensuite, une fois que l'on a choisi le niveau (en l'occurrence le niveau 1), on peut voir apparaître le niveau choisi:

DOSSIER PROFESSIONNEL (DP)



On peut voir la map avec les caisses, le personnage qui va déplacer les caisses et leurs emplacements. Une fois que l'on a placé une caisse à son objectif, elle change de couleur pour devenir dorée:

DOSSIER PROFESSIONNEL (DP)



Il ne reste plus qu'à placer les autres caisses jusqu'à réussir le niveau.



2. Précisez les moyens utilisés :

Pour réaliser ce projet j'ai utilisé:

- ❖ Python pour la programmation
- ❖ La librairie Pygames pour l'interface graphique
- ❖ Des images pour les caisses, les murs et l'ouvrier.

Ce projet m'a permis d'acquérir les compétences du référentiel:

- Développer une interface utilisateur de type desktop

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet seul.

4. Contexte

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

Chantier, atelier, service ➔ *My Sokoban.*

Période d'exercice ➔ Du : **17/04/2023** au : **22/04/2023**

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Activité-type 2 Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Exemple n° 1 - Développement d'une application de chat-SENT

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

Dans le cadre de ma formation de développeur et concepteur d'application, et ma présentation de fin d'année, j'ai dû réaliser un projet regroupant toutes les compétences que j'ai pu acquérir durant mon cursus. Le projet est un projet de chat sous forme d'application mobile, ce projet semblait très complet aussi bien dans la conception que dans le développement.

Pour réaliser cette activité type , j'ai divisé l'activité en plusieurs phases qui sont les suivantes:

1. Phase d'analyse

Après le recensement des besoins vus précédemment. J'ai réalisé le diagramme de cas d'utilisation. À l'aide de draw.oi.

Ce qui m'a amené à établir des règles de gestion suivantes :

- Un Utilisateur participe au minimum un et au maximum plusieurs channels;
- Un channel a au minimum un et au maximum plusieurs participants ;
- Un message est envoyé au minimum par un utilisateur et au maximum par utilisateur ;
- Un utilisateur peut envoyer au minimum 0 message et au maximum plusieurs messages;
- Un channel peut contenir au minimum 0 et au maximum plusieurs messages
- Un message appartient au minimum à un et au maximum à un channel ;

Nous avons déterminé les données élémentaires, les éventuelles données calculées, et les données composées en précisant leurs types.

Ce qui m'a permis de composer rapidement le dictionnaire de données.

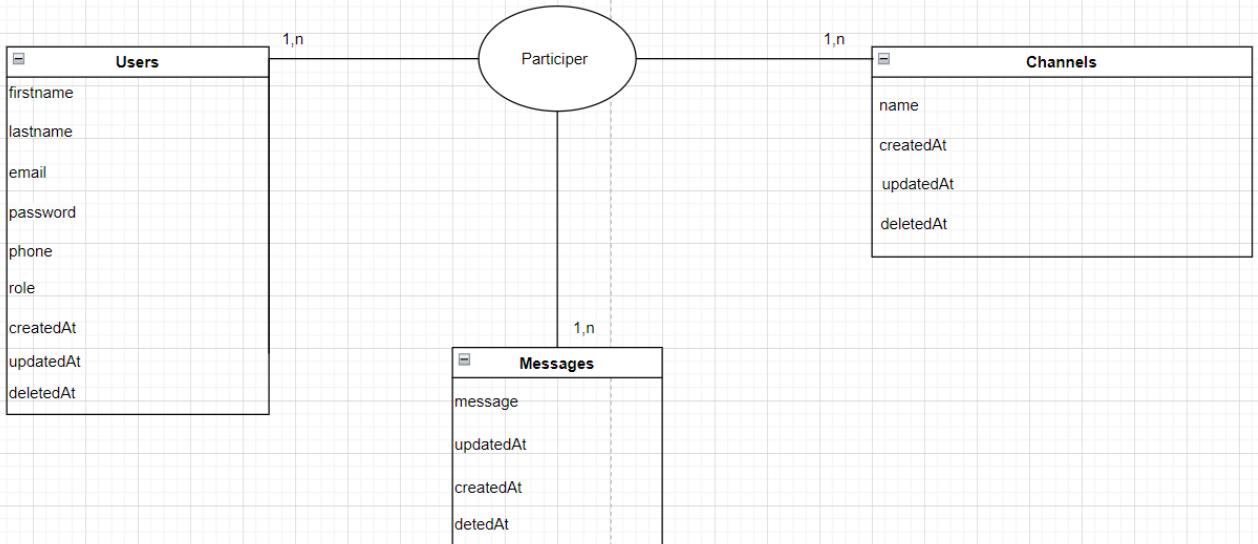
2. Phase de conception technique

Durant cette phase, J'ai mis en place les différents modèles, en m'appuyant et épurant le dictionnaire de données.

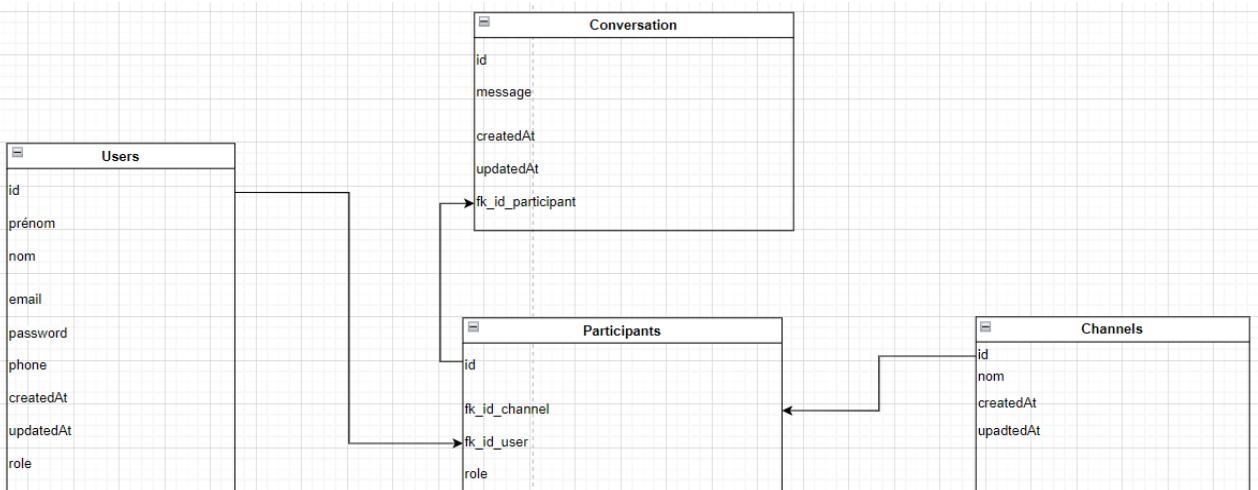
Voici les différents modèles que j'ai réalisés dans l'ordre suivant :

- I. **Le modèle conceptuel de données** du type modèle entité-association. Qui permet de décrire le système d'information à l'aide d'entités.

DOSSIER PROFESSIONNEL (DP)

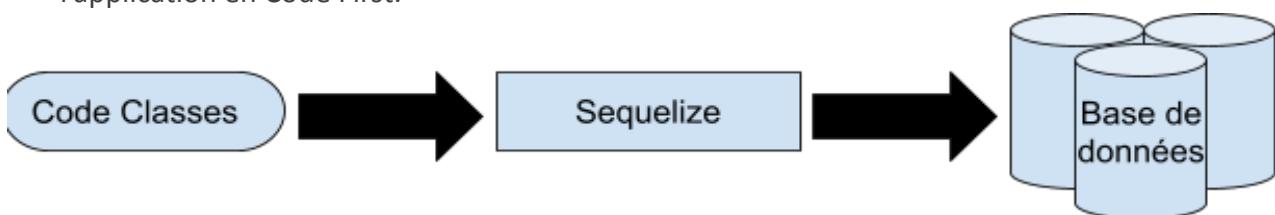


- II. **Modèle logique de données**, qui consiste à décrire la structure selon laquelle les données seront stockées dans la base de données. Voici le modèle logique de données modélisant notre projet SENT.



- III. **Le modèle physique** est un mixte entre le modèle conceptuel de données et le modèle logique de données. Je représente chaque entité et leurs attributs en précisant leur type (varchar(..), int ...). Mais je représente aussi les clés étrangères.

- IV. **Le diagramme de classes**, à partir de toutes ces modélisations. Je réalise le plus important d'entre eux car il représente l'ensemble des objets de notre système. De plus, j'ai conçu l'application en Code First.



3. Phase de réalisation

DOSSIER PROFESSIONNEL (DP)

● Utilisation de L' ORM sequelize

Sequelize est une bibliothèque JavaScript basée sur les promesses qui facilite l'interaction avec les bases de données relationnelles dans des applications Node.js. Elle fournit une interface conviviale pour effectuer des opérations de création, lecture, mise à jour et suppression (CRUD) sur la base de données en utilisant des objets JavaScript plutôt que d'écrire des requêtes SQL manuellement.

Voici quelques points importants sur Sequelize :

- ORM (Object-Relational Mapping)
- Prise en charge de plusieurs dialectes de bases de données
- Modèles et migrations
- Relations entre les modèles
- Opérations CRUD
- Validation des données
- Hooks et événements
- Transactions et contrôle des versions

Sequelize simplifie l'interaction avec les bases de données relationnelles dans les applications Node.js en fournissant une couche d'abstraction et en utilisant des objets JavaScript familiers pour effectuer des opérations sur la base de données. Cela permet aux développeurs de se concentrer davantage sur la logique métier plutôt que sur les détails de la gestion de la base de données.

Connexion à la base de données

Pour commencer j'ai installé le paquet npm sequelize `npm install sequelize`

Ensuite j'ai installé le paquet npm mysql2 `npm install mysql2` qui est un pilote spécifique à MySQL et permet à Sequelize de se connecter, de communiquer et de travailler avec une base de données MySQL. Il assure une compatibilité optimale avec les fonctionnalités et les spécificités de MySQL.

Il existe plusieurs paquets permettant une connexion à une base de données, j'ai choisi mysql. C'est la solution la plus simple et rapide à mettre en place pour interagir avec une base de données sql en nodejs.

j'importe les modules nécessaire

```
import { Sequelize } from "sequelize";
```

DOSSIER PROFESSIONNEL (DP)

je crée une instance sequelize en spécifiant les information de connexion: nom de la base de données, nom d'utilisateur et mot de passe. Ensuite on lui passe le host et le dialect

```
/*
 * CONNEXION A LA BASE DE DONNEES *
 */

let sequelize = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  "", //process.env.DB_PASSWORD_MACOS,

  {
    host: process.env.DB_HOST,
    dialect: "mysql",
    dialectOptions: {
      charset: "utf8mb4",
    },
    logging: false,
  }
);
```

Avec l'orm sequelize on peut gérer les relations entre les tables de la base de données. Sequelize met à disposition des méthodes comme pour l'exemple ci dessous , j'utilise la méthode hasMany et belongsTo

const db = {} ; Ici, un objet db est créé pour stocker les différentes tables de la base de données.

db.sequelize = sequelize; : L'objet sequelize, qui représente la connexion à la base de données, est assigné à la propriété sequelize de l'objet db. Cela permet d'accéder à l'instance de Sequelize dans d'autres parties de l'application via db.sequelize.

DOSSIER PROFESSIONNEL (DP)

```
*****  
/* MIS EN PLACE DES RELATIONS */  
*****  
  
const db = {};  
db.sequelize = sequelize;  
db.User = Users(sequelize);  
db.Channel = Channels(sequelize);  
db.Participant = Participants(sequelize);  
db.Conversation = Conversations(sequelize);  
  
db.UserhasMany(db.Participant, {  
  foreignKey: "userId",  
  // onDelete: "cascade",  
  // onUpdate: "cascade",  
  // hooks: true,  
});  
db.Participant.belongsTo(db.User, {  
  foreignKey: "userId",  
});  
  
db.ChannelhasMany(db.Participant, {  
  foreignKey: "channelId",  
  // onDelete: "cascade",  
  // onUpdate: "cascade",  
  // hooks: true,  
});  
db.Participant.belongsTo(db.Channel, {  
  foreignKey: "channelId",  
});  
  
db.ParticipanthasMany(db.Conversation, {  
  foreignKey: "participantId",  
  // onDelete: "cascade",  
  // onUpdate: "cascade",  
  // hooks: true,  
});  
db.Conversation.belongsTo(db.Participant, {  
  foreignKey: "participantId",  
});  
  You, il y a 5 mois * fix datafixture for all entity
```

je charge mes fixtures dans le fichier mock.js dans la base de données avec la méthode create de sequelize pour avoir un jeux de données

```
*****  
/* SYNCHRONISATION DES MODELS */  
*****  
// force: true  
sequelize  
  .sync({ force: true }) //  
  .then(() => {  
    You, il y a 6 mois * feat: user ro  
    users.map(user) => {  
      db.User.create(user);  
    };  
    channels.map(channel) => {  
      db.Channel.create(channel);  
    };  
    participants.map(participant) => {  
      db.Participant.create(participant);  
    };  
    conversations.map(conversation) => {  
      db.Conversation.create(conversation);  
    };  
    console.log("Database is connected...");  
  })  
  .catch((err) => {  
    console.log("Error connecting to database...");  
    console.log(err);  
  });  
  export default db;
```

je vérifie enfin si la connexion à la base de données peut être établie avec les informations fournies.

DOSSIER PROFESSIONNEL (DP)

```
*****  
/* CONNEXION A LA BDD ET LANCEMENT DU SERVER */  
*****  
Db.sequelize  
  .authenticate()  
  .then(() => {  
    console.log("Connection has been established successfully...");  
  })  
  .then(() => {  
    console.log("Synchronizing models with database...");  
    app.listen(process.env.APP_PORT, () => {  
      console.log(`  
        Server is running on port ${process.env.APP_PORT} : http://localhost:\${process.env.APP\_PORT}`  
      );  
    });  
  })  
  .catch((err) => {  
    console.log("Unable to connect to the database...");  
    console.log(err);  
  });  
  You, il y a 6 mois • setup database ...
```

2

2. Précisez les moyens utilisés :

- ❖ Environnement de développement intégré (IDE) **Visual Studio Code**
Visual Studio Code est un éditeur de code source populaire, offrant une interface utilisateur intuitive, une prise en charge de nombreux langages, une extensibilité grâce aux extensions, une intégration native avec Git, des fonctionnalités de débogage avancées, un terminal intégré et des outils de productivité pour les développeurs.
- ❖ **SQL**
SQL est utilisé dans une grande variété d'applications et de systèmes de gestion de bases de données (SGBD), tels que MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite, etc. Il est essentiel pour stocker, récupérer et manipuler efficacement les données dans les bases de données relationnelles.

DOSSIER PROFESSIONNEL (DP)

❖ MySql

MySQL est largement utilisé dans de nombreuses applications et systèmes, allant des sites web et des applications d'entreprise aux systèmes de gestion de contenu (CMS) tels que WordPress. En tant que SGBDR populaire et open source, MySQL offre une performance élevée, une stabilité et une flexibilité pour la gestion des données.

❖ Node.Js

Node.js est un environnement d'exécution côté serveur basé sur le moteur JavaScript V8 de Google Chrome. Il permet d'exécuter du code JavaScript côté serveur, contrairement à son utilisation traditionnelle côté client dans les navigateurs web.

❖ Express

Express est un framework web minimaliste pour Node.js qui facilite le développement d'applications web côté serveur. Il est conçu pour être simple, flexible et performant.

❖ Documentation

- Documentation officielle Sequelize
<https://sequelize.org/docs/v6/getting-started/>
- Documentation officiel NodeJs
<https://nodejs.org/en/docs>
- Documentation officiel Express
<https://expressjs.com/fr/>
- Le routage avec Express
<https://expressjs.com/fr/guide/routing.html>
- Intégration de bases de donnée avec ExpressJs
<https://expressjs.com/fr/guide/database-integration.html>

Ce projet m'a permis d'acquérir les compétences du référentiel:

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

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL (DP)

Johan Bouguermouh

Boris TIKHOMIROFF

Cyril Porez

4. Contexte

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

Chantier, atelier, service ▶ *Sent*

Période d'exercice ▶ Du : *02/01/2023* au : *29/01/2023*

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Concevoir et développer une application multicouches répartie en intégrant les recommandation de sécurité

Exemple n° 1 - *Développement d'une application de chat-SENT*

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

L'application Sent, présentée dans l'activité précédente, offre une réponse complète pour valider l'Activité-type 3.

Analyse:

Dans le cadre de ma formation à la Plateforme, pour l'obtention de mon titre RNCP Concepteur Développeur Application, j'ai réalisé une application mobile en React Native. Pour ce faire, j'ai utilisé le sujet comme base de cahier des charges.

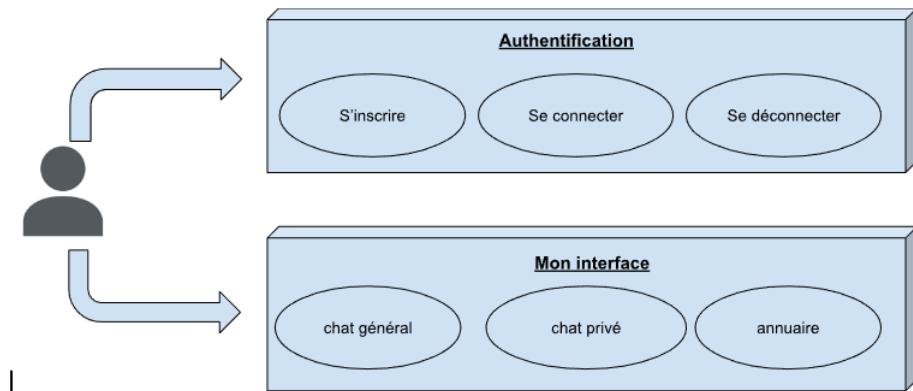


Figure - Diagramme de cas d'utilisation

● Scénario :

L'utilisateur se connecte, si l'utilisateur n'a pas de token d'authentification, il est redirigé automatiquement sur la page inscription. Après l'inscription ou la connexion, l'utilisateur est directement dirigé sur le chat général. Il peut converser sur le chat général ou il peut accéder à l'annuaire des utilisateurs afin de lancer une conversation privé en choisissant un ou plusieurs participants.

Conception technique:

Dans cette étape, nous avons mis en place le diagramme d'activité détaillés (Figure), le diagramme de classes. De plus, nous avons défini l'architecture physique et logicielle de l'application.

DOSSIER PROFESSIONNEL (DP)

L'application se base sur le design pattern MVC (Model – View - Controller), dont la couche model représente la partie de l'application qui exécute la logique métier, ce qui signifie qu'elle est responsable de la récupération des données. La vue retourne une présentation des données venant du model. Elle est responsable de l'utilisation des informations dont elle dispose pour produire une interface de présentation. Et enfin la couche Controller qui gère les requêtes des utilisateurs, il est responsable de retourner une réponse avec l'aide mutuelle des couches Model et Vue.

Comme technologies, nous avons décidé d'utiliser:

Pour notre Back end

- Node Js qui est un environnement d'exécution côté server pour JavaScript et offre une architecture orientée événements, un écosystème de modules riche, des performances élevées, une bonne scalabilité et une communauté active. Ces avantages en font un choix attrayant pour le développement d'applications web rapides, évolutives et efficaces.

- Express qui est un framework de développement web populaire utilisé avec Node.js et offre simplicité, flexibilité, performances et un large écosystème, ce qui en fait un choix populaire pour développer des applications web avec Node.js

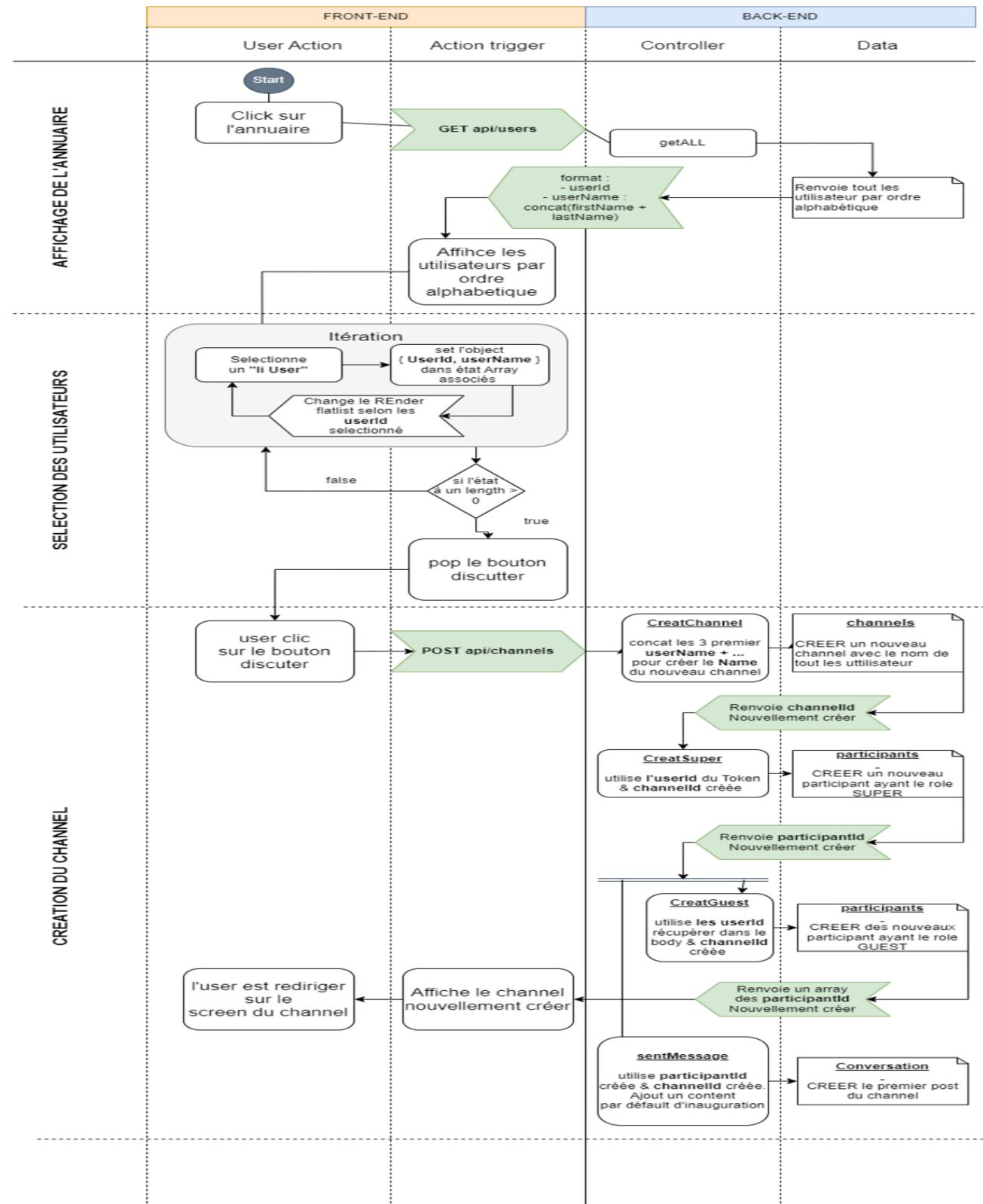
Pour notre Front end

- React Native qui est un framework open source développé par Facebook. Il permet de créer des applications mobiles multiplateformes en utilisant JavaScript et la bibliothèque React. Ses avantages comprennent le développement multiplateforme, des performances élevées, la réutilisation de code, un développement rapide, un écosystème actif et la fonctionnalité de Hot Reloading. Ces avantages en font un choix populaire pour le développement d'applications mobiles efficaces et rapides.

- Expo qui est une plateforme de développement d'applications mobiles multiplateformes qui simplifie le processus de développement, permet un développement rapide et offre un accès facile aux fonctionnalités natives des appareils. Ses avantages comprennent la simplification du processus de développement, le développement multiplateforme, l'accès aux fonctionnalités natives, l'expérience de développement rapide, le hot reloading, la publication simplifiée et une communauté active.

DOSSIER PROFESSIONNEL (DP)

DIAGRAMME D'ACTIVITE - CREATION DE CHANNEL



DOSSIER PROFESSIONNEL (DP)

Réalisation:

- Composants métier:

Nous avons développé un composant de chat privé. Un annuaire est mis en place afin de lister tous les utilisateurs par ordre alphabétique. Lorsque l'utilisateur connecté sélectionne dans l'annuaire un utilisateur minimum, un bouton discuté apparaît. Une fois cliqué, un channel se génère. Les différents utilisateurs liés à ce channel peuvent discuter.

```
const channelCreated = async () => {
  const response = await createChannel(selectedId);
  console.log("response", response.data.data.channel.channelId);
  setSelectedId([]);
  navigation.reset({
    index: 0,
    routes: [{ name: "Channels" }],
  });
  navigation.navigate("Channels", {
    screen: "Channel",
    params: { channelId: response.data.data.channel.channelId },
  });
};

useEffect(() => {
  You, il y a 6 mois
  const fetchData = () => {
    if (userList.length > 0) {
      setData(userList.data);
    }
  };
  fetchData();
}, [userList]);
```



```
<SectionList
  style={styles.flatlistContainer}
  sections={userList.data}
  keyExtractor={({item, index}) => item + index}
  renderItem={({item}) => (
    <Pressable
      key={item.userId}
      style={[
        styles.pessableContainer,
        {
          backgroundColor:
            selectedId.filter((user) => user.userId === item.userId)
            .length > 0
            ? "#FFFFF3"
            : "#FFFFFF00",
          borderwidth:
            selectedId.filter((user) => user.userId === item.userId)
            .length > 0
            ? 1
            : 0,
          bordercolor: colorsChart.primary,
          borderradius: 10,
        },
      ]}>
      You, il y a 6 mois * update branch annuailelist and recuperer
      onPress={() => {
        handleSelect(item);
      }}
      onLongPress={() => {
        showModal(item);
        console.log("onpressitem", item);
      }}
    >
      <ListComponent
        avatar=""
        pseudoName={item.userName}
        lastMessage=""
      />
      <Pressable>
    </Pressable>
  )}>
```

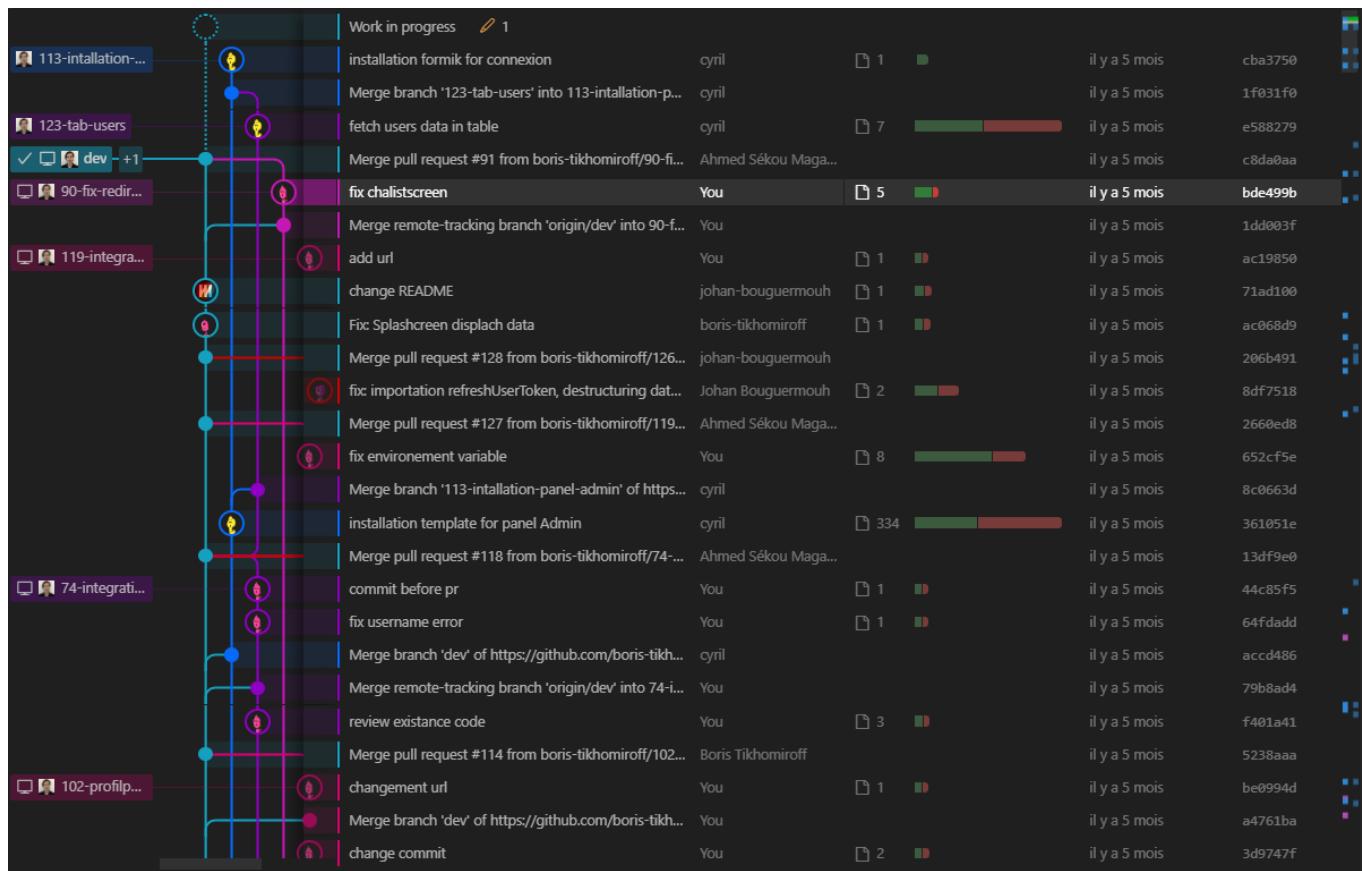


```
{selectedId.length > 0 && (
  <Pressable
    style={styles.buttonContainer}
    onPress={() => {
      channelCreated();
    }}
  >
    <View style={styles.button}>
      <Text style={styles.buttonText}>Discuter</Text>
      <Image
        style={styles.image}
        source={require("../.../assets/send.png")}
      />
    </View>
  </Pressable>
)>
  You, il y a 6 mois * update branch annuailelist and recuperer
```

DOSSIER PROFESSIONNEL (DP)

Outil collaboratif et version:

Ce projet, a été réalisé en groupe, il était indispensable de se répartir les tâches pour le bon avancement du projet. Dans cette optique, nous avons mis en place un repository du projet sur GitHub, GitHub est un système de contrôle de version distribué. GitHub permet aux développeurs de collaborer sur des projets de développement de logiciels, de partager du code source, de suivre les problèmes et de gérer les versions de leur code.



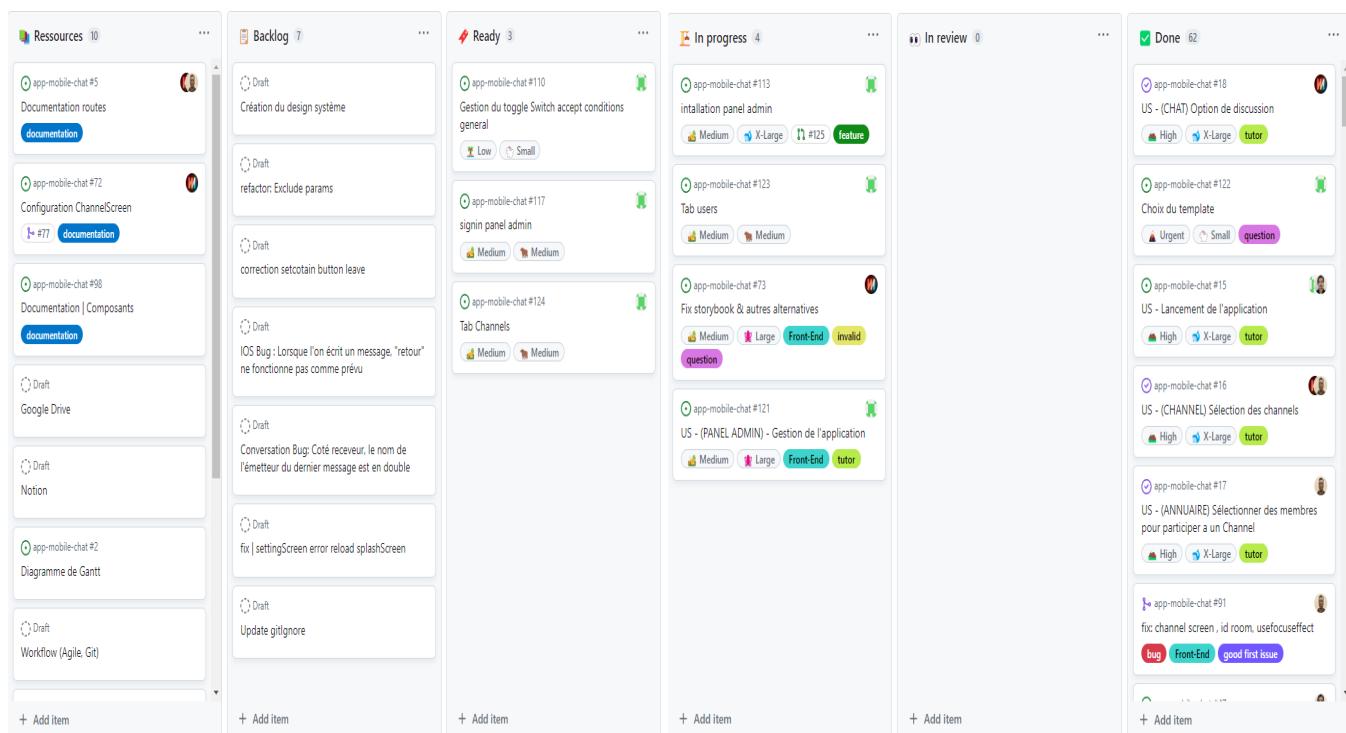
DOSSIER PROFESSIONNEL (DP)

Gestion du projet

Durant notre mois de développement, nous avons mis en place des daily meeting, pour parler de l'avancée des fonctionnalités attribuées à chacun chaque matin à huit heures. Chaque semaine le chef de projet changé, afin que chaque membre du groupe puisse vivre cette expérience.

Nous avons utilisé le Kanban de GitHub, c'est une fonctionnalité intégrée à GitHub qui permet de visualiser et de gérer les tâches d'un projet sous forme de tableau Kanban.

En utilisant le Kanban de GitHub, nous pouvions organiser notre flux de travail de manière visuelle, suivre l'état des tâches, gérer les priorités et collaborer efficacement pour mener à bien notre projets. Cela favorise une meilleure coordination et une plus grande transparence au sein de l'équipe de développement.



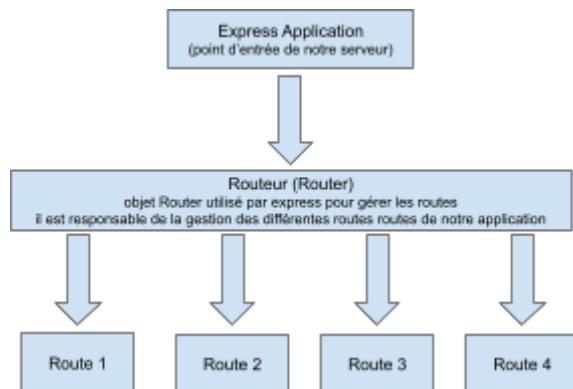
L'API suit une approche MVC (Modèle-Vue-Contrôleur) où chaque composant a un rôle spécifique



Architecture de l'API

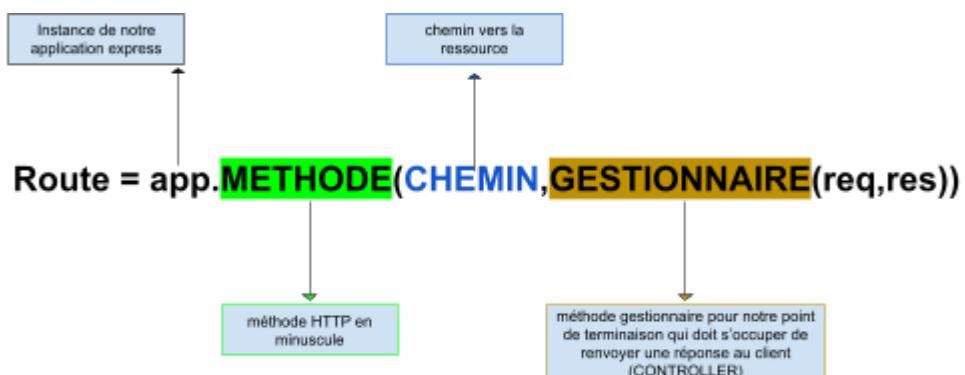
DOSSIER PROFESSIONNEL (DP)

Afin de mettre en place le routage, j'ai utilisé le routeur de express js.



ExpressJs met à disposition un middleware qui gère facilement le routage du projet.

Pour déclarer une nouvelle route à express , on peut le résumer à cette exemple :



Comme expliqué plus haut mon but a été de rendre mon code facilement modifiable et maintenable j' ai donc divisé cette partie .

La première chose que j'ai faite a été la séparation des routes et le code d'implémentation de celles-ci.

```
***** You, il y a 6 mo  
/* ROUTES D'APPLICATION */  
*****  
app.use("/api/users", userRoutes);  
app.use("/api/conversations", conversationRoutes);  
app.use("/api/participants", participantsRoutes);  
app.use("/api/channels", channelsRoutes);
```

DOSSIER PROFESSIONNEL (DP)

Le middleware use de cet exemple permet de spécifier le routeur qui doit être utilisé en fonction d'une route appelée. Comme on peut le voir dans cet exemple, si l'URL '/api/users' est appelée , mon routeur userRoutes est utilisé. App est une instance de express et la fonction use permet d'associer un routeur dans lequel on trouve toutes les routes pour cette ressource. Dans les routeurs une classe express.Router est instancié. C'est un middleware niveau routeur. Un middleware niveau routeur fonctionne de la même manière qu'un middleware. Grâce à la méthode route() de celui-ci le schéma de la route est défini. Ainsi je peux utiliser les méthodes get , post, put, patch et delete afin de pouvoir effectuer différentes actions en fonction de la méthode choisie.

```
import { Router } from "express";  
  
const router = Router();  
  
router.get("/", getAll);
```

Dans les codes ci-dessus , j'ai importé la classe Router de express , grâce à cette classe , je peux faire appelle aux différentes méthodes HTTP. Comme on peut le voir, la méthode utilisée est le get. Le premier paramètre de la fonction est l'URL , le deuxième est la fonction de gestionnaire de la route qu' on trouvera dans le controller associé à la route.

Utilisation des Middlewares

```
const checkIsAdmin = (req, res, next) => {  
  const { role } = req.decoded;  
  if (role !== "ADMIN") {  
    const message = `Vous n'avez pas les droits d'administration.`;  
    return res.status(401).json({ message });  
  }  
  
  next();  
};
```

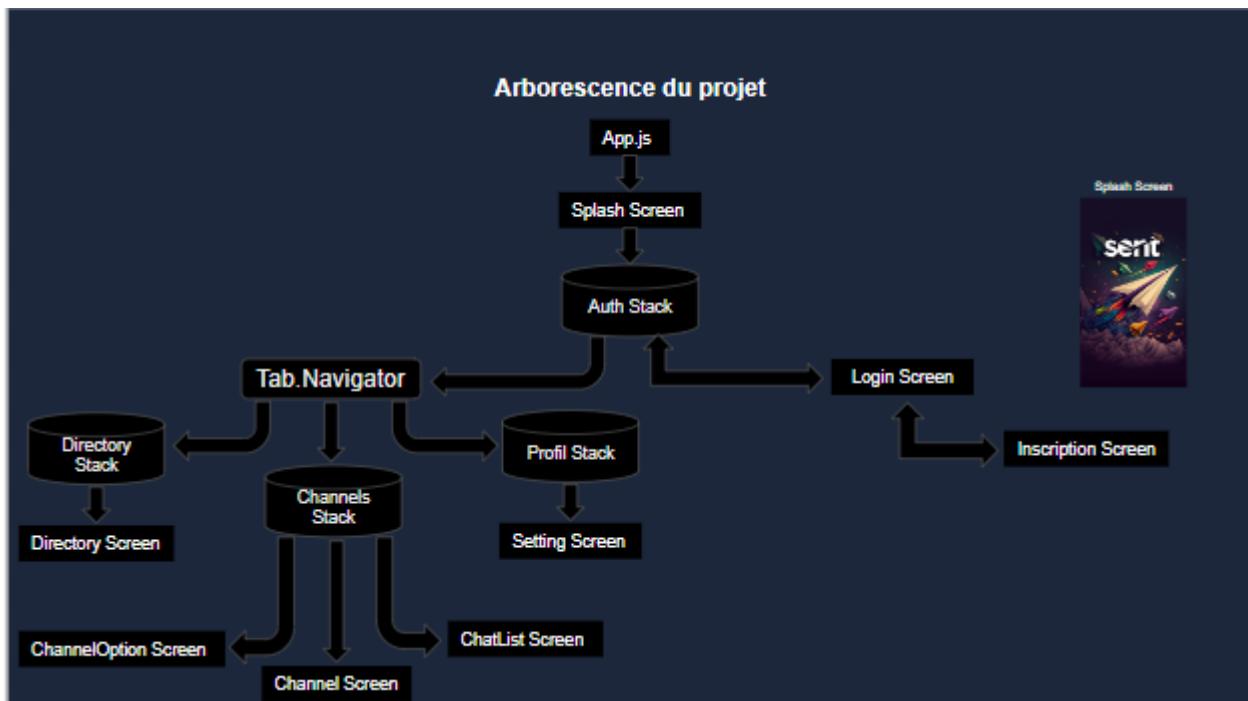
Ce middleware permet comme son nom l'indique la vérification du rôle d'un utilisateur.

La route sera inaccessible si le rôle utilisateur n'est pas '**ADMIN**'

DOSSIER PROFESSIONNEL (DP)

```
router.put("/", checkToken, checkIsAdmin, updateAll);
```

Notre application mobile a été conçue avec React Native



Page d'inscription et de connexion

DOSSIER PROFESSIONNEL (DP)

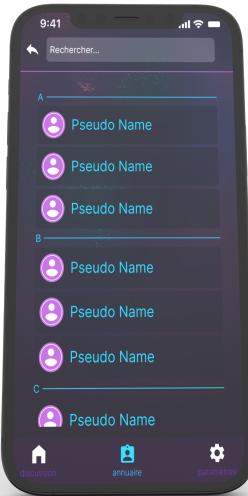


A l'ouverture de l'application, l'appli vérifie si un token d'authentification existe. Si il est présent, l'utilisateur est dirigé vers le chat général. Sinon l'utilisateur sera envoyé vers la page de connexion.

Un chat général sur lequel l'utilisateur sera dirigé après sa connexion et un chat privé qui pourrait être créée par l'utilisateur



DOSSIER PROFESSIONNEL (DP)



L'annuaire avec la liste de tous les utilisateurs



Liste des conversations auxquelles l'utilisateur participe et le dernier message envoyé dans chaque conversation

DOSSIER PROFESSIONNEL (DP)

Une page de profil

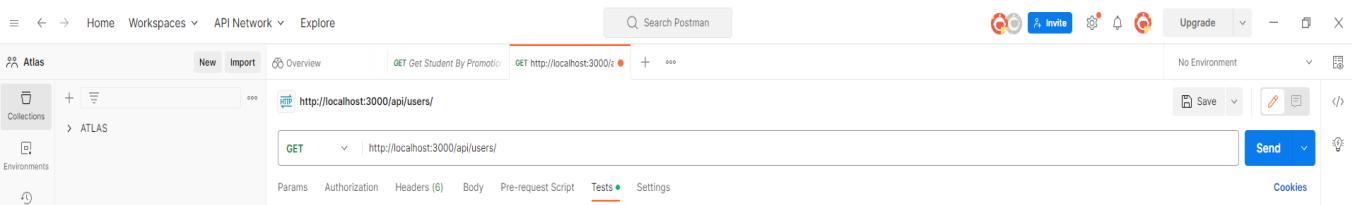


DOSSIER PROFESSIONNEL (DP)

Test fonctionnel d'api avec Postman

Nous souhaitons tester si l'API renvoie les données correctement . Nous avons testé nos Url d'API.

On s'est servi de postman pour réaliser cette tâche



Exemple d'url: **GET** http://localhost:3000/api/users/

Pour récuperer l'ensemble des utilisateur inscrit sur notre app mobile

Ensute on clique sur l'onglet "Tests" juste en dessous de la zone de requête.

Dans la section des scripts de test, nous avons écrit du code JavaScript pour effectuer des assertions sur la réponse de l'API.

DOSSIER PROFESSIONNEL (DP)

```
// Vérification du code de statut de la réponse
pm.test("Statut de la réponse", function () {
    pm.response.to.have.status(200);
});

// Vérification du format de la réponse en JSON
pm.test("Format de la réponse", function () {
    pm.response.to.be.json();
});

// Vérification du contenu de la réponse
pm.test("Contenu de la réponse", function () {
    var jsonData = pm.response.json();

    pm.test("Vérification de la réponse", function () {
        pm.expect(jsonData).to.not.be.null;
        pm.expect(jsonData).to.not.be.undefined;
        // Autres assertions...
    });

    pm.test("Vérification des propriétés", function () {
        pm.expect(jsonData.data[0]).to.have.property('id').that.is.not.null.and.not.undefined;
        // Autres assertions...
    });

    var expectedValue = "some value";
    pm.test("Vérification de la valeur", function () {
        pm.expect(jsonData.data[0].firstname).to.equal('Cyril');
        // Autres assertions...
    });

    // Vérification du nombre d'utilisateurs renvoyés
    pm.expect(jsonData.data).to.have.length(12);

    // Vérification des propriétés des utilisateurs
    pm.expect(jsonData.data[0]).to.have.property('id');
    pm.expect(jsonData.data[0]).to.have.property('firstname');
    pm.expect(jsonData.data[0]).to.have.property('lastname');
});
});
```

- nous vérifions que la réponse de l'API a un code de statut 200
- qu'elle est au format JSON
- on vérifie le contenu de la réponse

DOSSIER PROFESSIONNEL (DP)

- Vérification de la réponse pour voir si elle est nulle ou pas .
- On vérifie si les propriété de la réponse ne sont pas vide ou undefined
- On vérifie si réponse correspond au résultat attendu
- qu'elle contient les détails corrects pour l'utilisateur avec l'ID 1

Body Cookies Headers (8) Test Results (5/6)

All Passed Skipped Failed

| | |
|------|--|
| PASS | Statut de la réponse |
| PASS | Format de la réponse |
| PASS | Vérification de la réponse |
| PASS | Vérification des propriétés |
| FAIL | Vérification de la valeur Assertion Error: expected 'Ahmed' to equal 'Cyril' |
| PASS | Contenu de la réponse |

Body Cookies Headers (8) Test Results (6/6)

All Passed Skipped Failed

| | |
|------|-----------------------------|
| PASS | Statut de la réponse |
| PASS | Format de la réponse |
| PASS | Vérification de la réponse |
| PASS | Vérification des propriétés |
| PASS | Vérification de la valeur |
| PASS | Contenu de la réponse |

```
var expectedValue = "some value";
pm.test("Vérification de la valeur", function () {
    pm.expect(jsonData.data[0].firstname).to.equal(['Ahmed']);
    // Autres assertions...
});
```

Test Unitaire avec jest

Nous souhaitons tester si chaque unité fonctionne correctement de manière isolée. Nous effectuons des tests unitaires. Ils garantissent la stabilité et la fiabilité du code en détectant rapidement les erreurs et en facilitant la maintenance du système. Les tests unitaires contribuent ainsi à améliorer la qualité du logiciel et à assurer un développement itératif plus efficace. Pour ce faire nous utilisons le framework jest.

DOSSIER PROFESSIONNEL (DP)

on voulait tester nos endpoint, voici l'exemple suivant effectué une route:`http://localhost:3000/api/users/me`.

Nous testons le statut de la requête.

```
import axios from "axios";
import * as jwt from "jsonwebtoken";
import { config } from "dotenv";      You, maintenant • Uncommitted changes

config();
const url = "http://localhost:3000/api/users/me";

// Création du token
const token = jwt.sign({ id: 1, role: "GUEST" }, process.env.JWT_KEY, {
  expiresIn: process.env.JWT_EXPIRES_IN,
});

test("Vérifie que la requête GET vers /api/me renvoie un statut 200 avec un jeton Bearer", async (
  const config = {
    headers: {
      Authorization: `Bearer ${token}`,
    },
  };

  const response = await axios.get(url, config);
  expect(response.status).toBe(200);
));
```

Voici le résultat obtenu, la validation des tests.

```
> app-mobile-chat@1.0.0 test
> jest

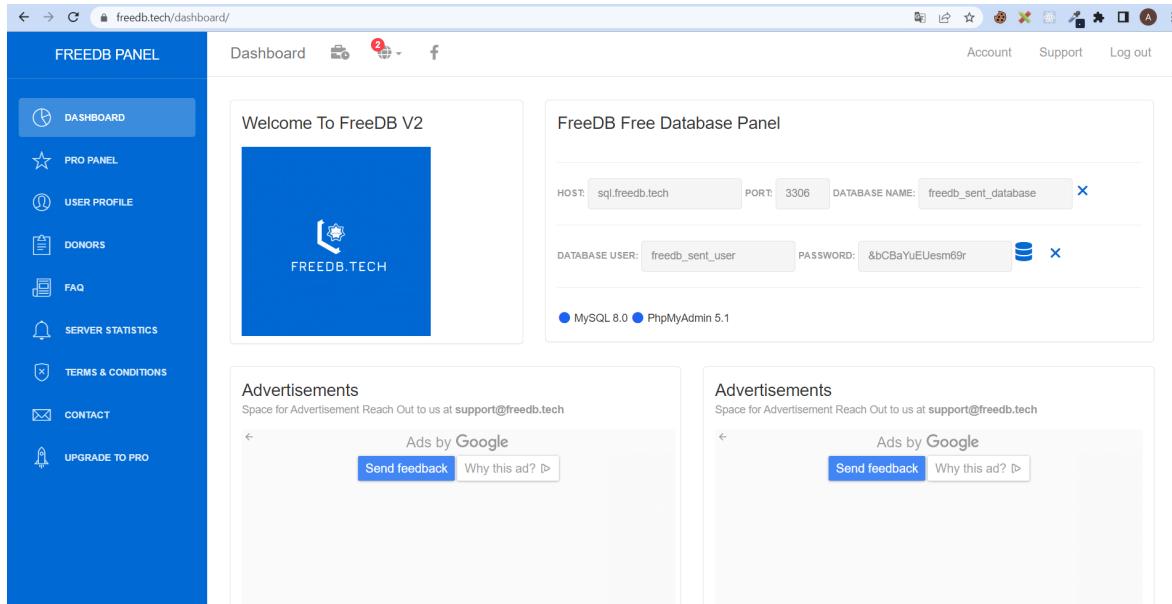
PASS  tests/example.test.js
PASS  tests/userControllers.test.js

Test Suites: 2 passed, 2 total
Tests:       4 passed, 4 total
Snapshots:  0 total
Time:        1.656 s, estimated 2 s
Ran all test suites.
PS C:\Users\magas\Desktop\back> []
```

DOSSIER PROFESSIONNEL (DP)

Déploiement application

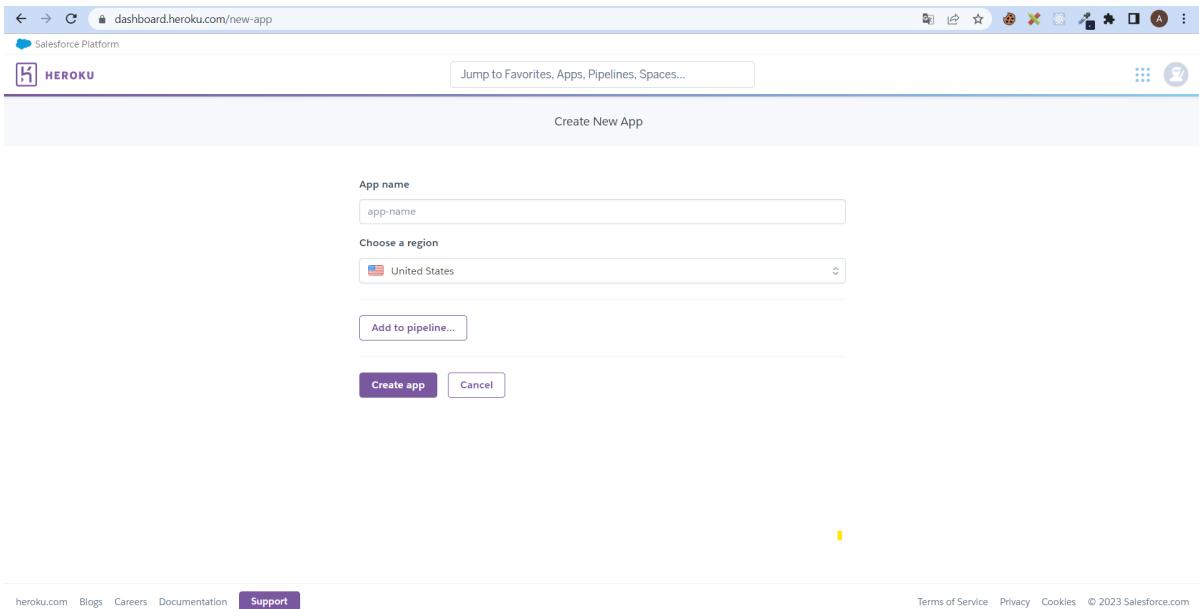
Nous avons commencé par mettre en ligne notre base de données sur FREEDB TECH.



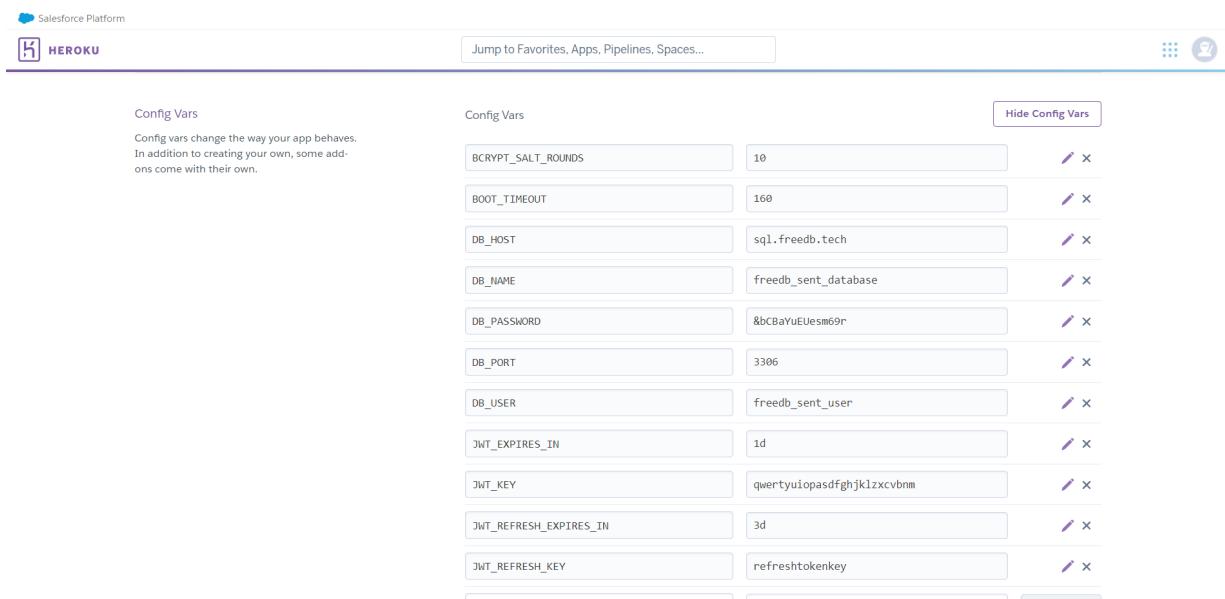
The screenshot shows the Freedb.tech dashboard at freedb.tech/dashboard/. The left sidebar is titled 'FREEDB PANEL' and includes links for Dashboard, PRO PANEL, USER PROFILE, DONORS, FAQ, SERVER STATISTICS, TERMS & CONDITIONS, CONTACT, and UPGRADE TO PRO. The main content area has a 'Welcome To FreeDB V2' banner with the Freedb.Tech logo. To the right is the 'FreeDB Free Database Panel' with fields for HOST (sql.freedb.tech), PORT (3306), DATABASE NAME (freedb_sent_database), DATABASE USER (freedb_sent_user), and PASSWORD (&bCBAYuUEsm69r). Below these fields are MySQL 8.0 and PhpMyAdmin 5.1 links. There are also two advertisement slots for Google Ads.

Nous avons choisi Heroku comme système de déploiement pour notre backend. l'image ci-dessous montre la simplicité de création l'app.

DOSSIER PROFESSIONNEL (DP)



On passe l'étape de config ou on définit nos variables d'environnements



On pousse notre back sur le repo Git créé et fourni par heroku pour notre projet (voir l'image du dessous), cette action mène au déploiement de notre projet sur heroku

DOSSIER PROFESSIONNEL (DP)

The screenshot shows the Heroku Platform interface. At the top, there's a header with the Salesforce Platform logo, the Heroku logo, and a search bar. Below the header, the app name 'api-sent' is selected. The main navigation bar includes links for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. The 'Settings' tab is currently active.

In the 'App Information' section, the app name is set to 'api-sent'. The region is 'Europe', stack is 'heroku-22', framework is 'Node.js', and the Heroku git URL is '<https://git.heroku.com/api-sent.git>'.

Below the app information, there's a terminal window showing a JSON response from the API endpoint. The response indicates there are 12 users, with a sample of four users listed:

```
// 20230720211633
// https://api-sent-e38ddc20b501.herokuapp.com/api/users

1 {
  "message": "Il y a 12 des utilisateurs",
  "count": 12,
  "length": 12,
  "data": [
    {
      "id": 1,
      "firstname": "Ahmed",
      "lastname": "Magassouba"
    },
    {
      "id": 8,
      "firstname": "Blade",
      "lastname": "Ranger"
    },
    {
      "id": 2,
      "firstname": "Boris",
      "lastname": "TIKHOMIROFF"
    },
    {
      "id": 3,
      "firstname": "Cyril",
      "lastname": "Porez"
    },
    {
      "id": 11,
      "firstname": "Francois",
    }
  ]
}
```

L'exemple ci dessus nous montre un exemple de requête avec notre app déployé en utilisant son url

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

❖ Figma

Figma est un outil de conception d'interface utilisateur (UI) et de prototypage collaboratif basé sur le cloud. Il offre une plateforme complète pour la conception, la collaboration et le partage de maquettes d'interfaces utilisateur interactives.

❖ Environnement de développement intégré (IDE) Visual Studio Code

Visual Studio Code est un éditeur de code source populaire, offrant une interface utilisateur intuitive, une prise en charge de nombreux langages, une extensibilité grâce aux extensions, une intégration native avec Git, des fonctionnalités de débogage avancées, un terminal intégré et des outils de productivité pour les développeurs.

❖ Git

Git est un système de contrôle de version distribué qui permet de suivre les modifications apportées au code source d'un projet. Il facilite la collaboration, la gestion des branches, la fusion des modifications, la gestion des conflits et la gestion des référentiels distants. Git est largement utilisé dans l'industrie du développement logiciel pour gérer efficacement le code source et faciliter le travail en équipe.

❖ GitHub

GitHub est une plateforme web qui facilite la gestion de projets de développement de logiciels en utilisant Git. Elle offre des fonctionnalités de contrôle de version, de collaboration, de suivi des problèmes, d'intégration continue, de gestion des permissions et d'intégration avec des outils tiers. GitHub est largement utilisé par la communauté de développement logiciel pour héberger, partager et collaborer sur des projets open source et privés.

❖ Le Kanban de GitHub

En utilisant le Kanban de GitHub, nous pouvions organiser visuellement les tâches de notre projet, suivre leur progression, gérer les assignations et collaborer efficacement avec les membres de l'équipe. Cela facilite la planification, la gestion des priorités et la coordination des efforts de développement.

❖ SQL

SQL est utilisé dans une grande variété d'applications et de systèmes de gestion de bases de données (SGBD), tels que MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite, etc. Il est essentiel pour stocker, récupérer et manipuler efficacement les données dans les bases de données relationnelles.

❖ MySql

MySQL est largement utilisé dans de nombreuses applications et systèmes, allant des sites web et des applications d'entreprise aux systèmes de gestion de contenu (CMS) tels que WordPress. En tant que SGBDR populaire et open source, MySQL offre une performance élevée, une stabilité et une flexibilité pour la gestion des données.

DOSSIER PROFESSIONNEL (DP)

❖ **Node.Js**

Node.js est un environnement d'exécution côté serveur basé sur le moteur JavaScript V8 de Google Chrome. Il permet d'exécuter du code JavaScript côté serveur, contrairement à son utilisation traditionnelle côté client dans les navigateurs web.

❖ **Express**

Express est un framework web minimaliste pour Node.js qui facilite le développement d'applications web côté serveur. Il est conçu pour être simple, flexible et performant.

❖ **React Native**

React Native est un framework open source développé par Facebook qui permet de créer des applications mobiles pour iOS et Android en utilisant le langage JavaScript et la bibliothèque React.

❖ **Expo**

Expo est une plateforme de développement qui fournit des outils et des services pour créer des applications mobiles avec React Native.

❖ **Expo Go**

Expo Go est une application compagnon qui permet de prévisualiser et de tester les applications développées avec Expo sur des appareils mobiles réels. Expo Go facilite le processus de développement et permet une prévisualisation en temps réel des applications Expo sur iOS et Android.

❖ **Google Drive**

Google Drive est un service populaire pour le stockage en ligne et la gestion des fichiers. Il offre une solution pratique pour stocker, synchroniser, partager et collaborer sur des fichiers. Ils nous a permis de partager des documentations pour que toutes l'équipe puisse en profiter.

❖ **Notion**

Notion est une plateforme de productivité tout-en-un qui permet aux utilisateurs de créer, d'organiser et de collaborer sur différents types de contenus tels que des notes, des tâches, des bases de connaissances, des tableaux, des calendriers et bien plus encore.

❖ **Documentation**

→ Documentation officiel react native

<https://reactnative.dev/>

→ Documentation officiel expo

<https://expo.dev/>

→ Documentation officiel NodeJs

<https://nodejs.org/en/docs>

DOSSIER PROFESSIONNEL (DP)

- Documentation officiel Express
<https://expressjs.com/fr/>
- Le routage avec Express
<https://expressjs.com/fr/guide/routing.html>
- Les middlewares
<https://medium.com/@younessbnnj/expressjs-les-middlewares-a89c9a560ce5#:~:text=Les%20fonctions%20middlewares,-Un%20des%20concepts&text=Dans%20le%20cas%20d%27express,%C3%A0%20une%20autre%20fonction%20middleware.>
- Intégration de bases de donnée avec ExpressJs
<https://expressjs.com/fr/guide/database-integration.html>

Ce projet m'a permis d'acquérir les compétences du référentiel:

- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
- Concevoir une application
- Développer des composants métier
- Construire une application organisée en couche
- Développer une application mobile
- Préparer et exécuter des plan de test d'une application
- Préparer et exécuter le déploiement d'une application

3. Avec qui avez-vous travaillé ?

Johan Bouguermouh

Boris TIKHOMIROFF

Cyril Porez

4. Contexte

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

Chantier, atelier, service ➤ **SENT.**

Période d'exercice ➤ Du : 02/01/2023 au : 29/01/2023

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL ^(DP)

DOSSIER PROFESSIONNEL^(DP)

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. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné(e) Magassouba Ahmed sékou ,

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 20/07/2023

pour faire valoir ce que de droit.

Signature : Ahmed magassouba

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)