



DOSSIER PROFESSIONNEL (DP)

Nom de naissance Bouguermouh
Nom d'usage Bouguermouh
Prénom Johan
Adresse 14 Boulevard Joachim Elie Vezien 13008 Marseille

Titre professionnel visé

Concepteur(trice) Développeur(se) 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
» Sent Application Mobile	p.	5
» Sokoban	p.	27
Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité	p.	31
» Sent Application Mobile	p.	31
Concevoir et développer la persistance des données en intégrant les recommandations de sécurité	p.	48
» Sent Application Mobile	p.	48
Documents illustrant la pratique professionnelle (facultatif)	p.	
Annexes (Si le RC le prévoit)	p.	
Déclaration sur l'honneur	p.	53

DOSSIER PROFESSIONNEL ^(DP)

EXEMPLES DE PRATIQUE

PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type

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

Exemple n°1 - Sent Application Mobile

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

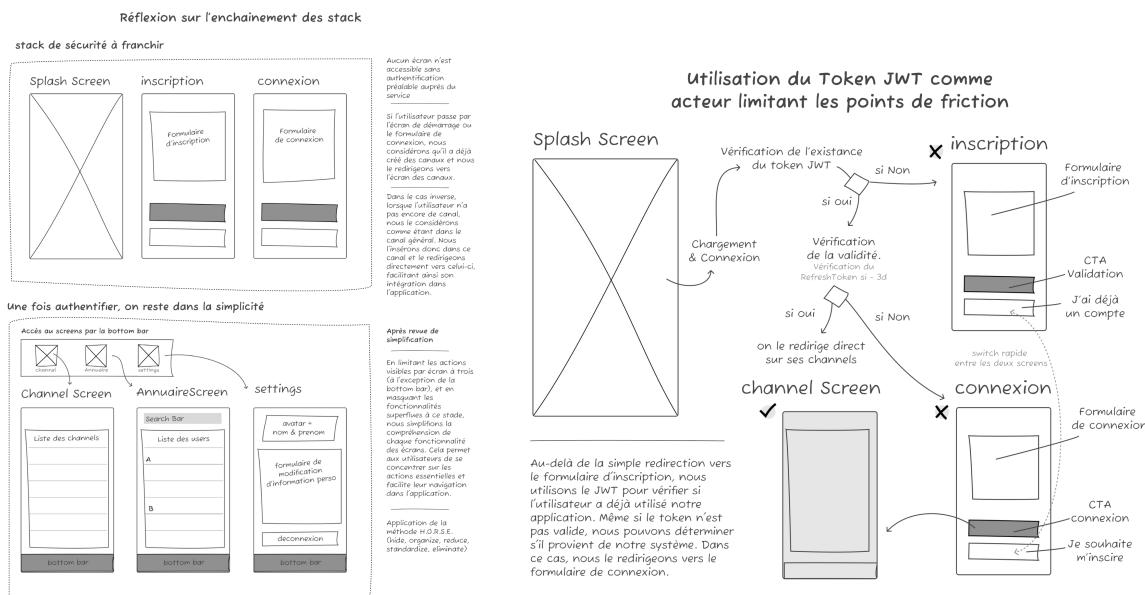
Au cours du projet académique qui s'est étendu sur une durée d'un mois, notre équipe composée de quatre personnes avait pour objectif de concevoir et développer une application mobile accompagnée d'un panneau d'administration. La première semaine du projet a été consacrée à la phase de conception et à l'organisation de l'équipe.

Maquetter une application

Lors de cette semaine, nous nous sommes attardés sur l'activité consistant à **maquetter une application**.

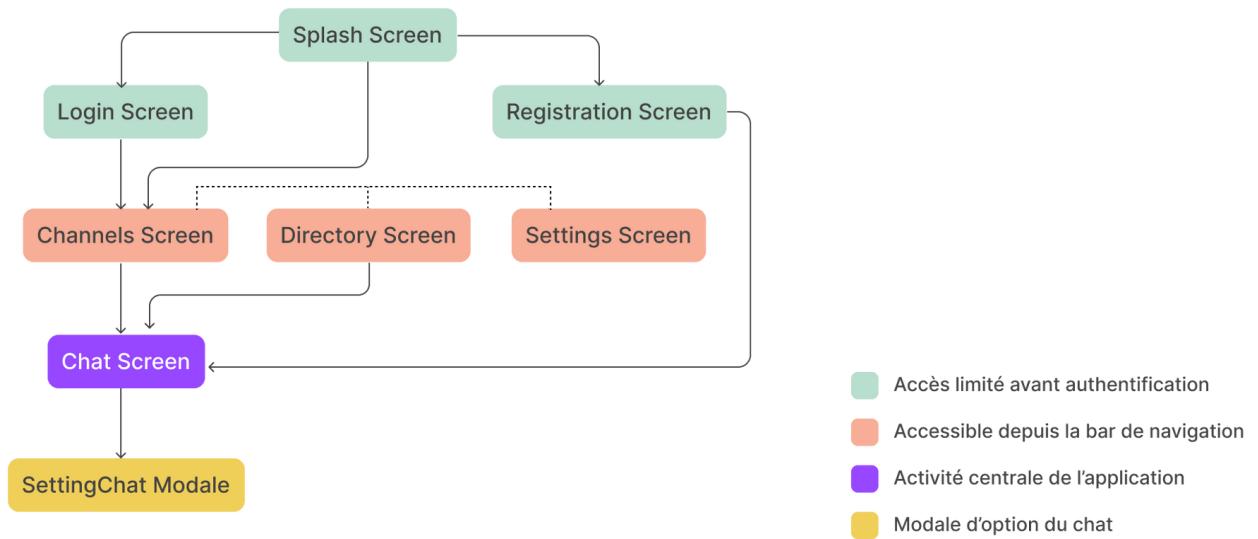
1. Nous avons effectué préalablement un **zoning** de manière à ce que la maquette respecte les principes de sécurisation d'une interface utilisateur.

- > Le zoning comprend la nécessité d'établir une interface permettant d'authentifier l'utilisateur.
- > Les informations sensibles et personnelles d'autres utilisateurs ne sont pas accessibles ou sont masquées, conformément aux principes de minimalité.
- > Selon les normes du RGPD, une demande d'autorisation explicite doit être présentée.
- > Un utilisateur doit obligatoirement s'authentifier pour accéder à d'autres interfaces de l'application, en dehors de l'écran d'accueil, de l'enregistrement ou du module de connexion.

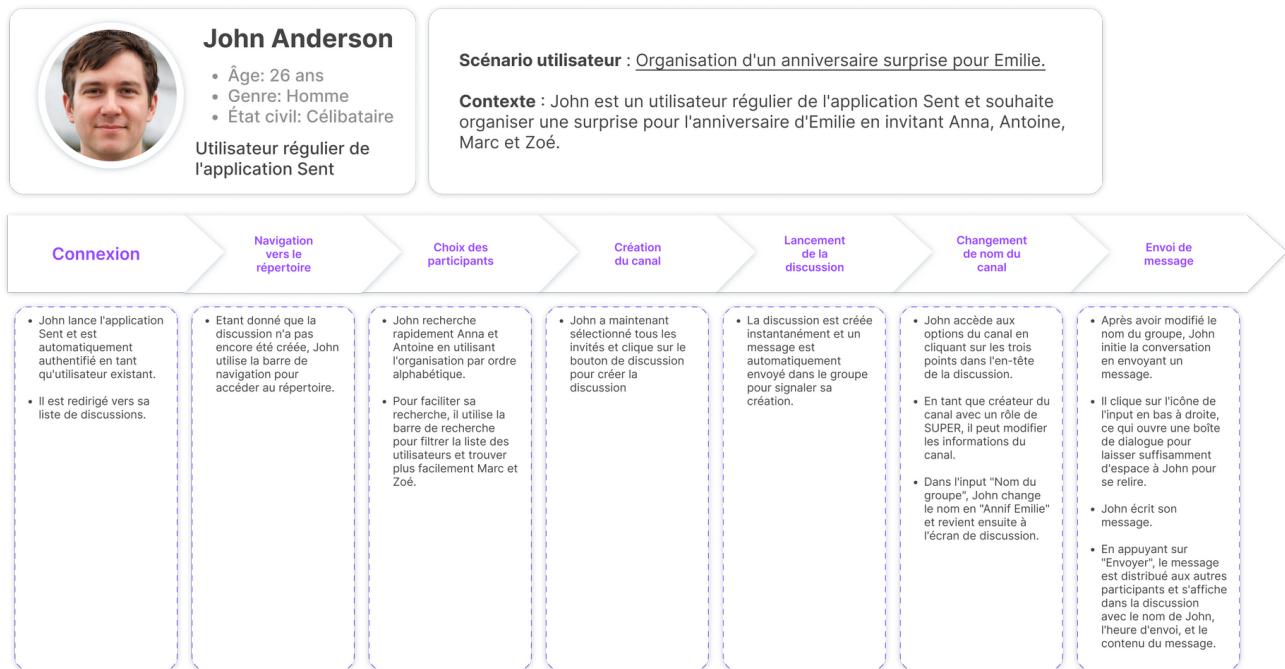


DOSSIER PROFESSIONNEL (DP)

2. Nous avons formalisé l'enchaînement des écrans pas un schéma

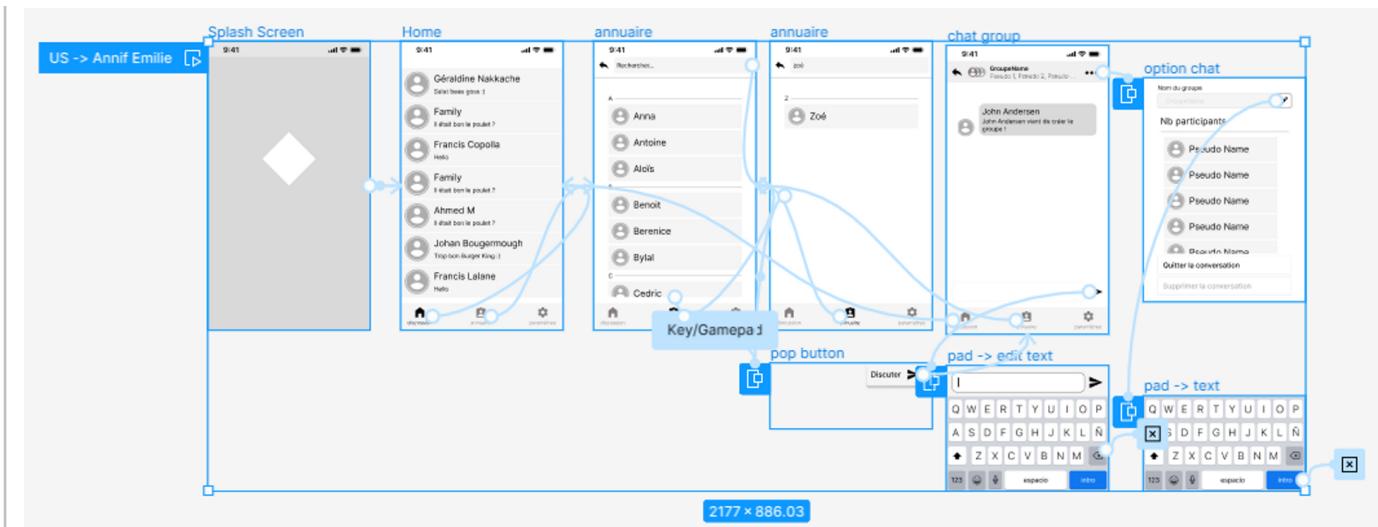


3. Nous nous sommes assuré que la maquette prend en compte les spécificités fonctionnelles décrites dans les scénarios utilisateur



exemple de scénario utilisateur

DOSSIER PROFESSIONNEL (DP)



ScreenShot de l'interface d'activité Figma sur le wireframe lors du maquettage du scénario en question

3. Une fois le wireframe fonctionnel. Nous avons pu passer à la maquette haute fidélité

The high-fidelity mockup displays the following screens:

- Splash Screen:** Shows the Senit logo and a "Continuer" button.
- Inscription:** Shows fields for "Prénom", "Nom", "Email", "Mot de passe", and "Confirmer mot de passe".
- connexion:** Shows fields for "Email" and "Mot de passe".
- Home:** Shows a list of contacts: Géraldine Nakache, Family, Francis Copolla, Family, etc.
- Profil:** Shows a profile card for "Loren Ipsum" with edit icons.
- annuaire:** Shows a list of contacts with edit icons.
- public chat:** Shows a list of users in a public chat.
- chat one by one:** Shows a list of users in a one-on-one chat.
- chat group:** Shows a list of users in a group chat.
- option chat:** Shows options for a group chat.

Annotations provide developer details for each screen:

- Inscription:** Includes notes for "Pour l'input numero de téléphone", "Pour l'input mot de passe", and "CTA (en bas)".
- connexion:** Includes notes for "Pour l'input email" and "Pour l'input mot de passe".
- channelList:** Includes notes for "CTA (en haut)" and "Pour le filtre".
- profil:** Includes notes for "Pour l'input nom et prénom", "Pour l'input adresse", "Pour l'input ville", "Pour l'input code postal", and "Pour l'input pays".
- annuaire:** Includes notes for "Pour l'input recherche" and "Pour l'input filtre".
- public chat:** Includes notes for "Pour l'input message".
- chat one by one:** Includes notes for "Pour l'input message".
- chat group:** Includes notes for "Pour l'input message".
- option chat:** Includes notes for "Nom du groupe", "Nb participants", and "Quitter la conversation".

Legend at the bottom:

- Red dashed box: Insertion d'élément natif du téléphone
- Purple dashed box: Screens haute fidélité
- Cyan dashed box: Detail technique pour faciliter le développement

✓ L'ensemble du processus d'activité suivante couvre les compétences professionnels attendue **Maquetter sur application**

Développer des composants d'accès aux données

1.Lors de la réalisation de cette application, nous avons développé des composants d'accès

DOSSIER PROFESSIONNEL (DP)

aux données répondant aux fonctionnalités décrites dans le dossier de conception technique.
Afin de faciliter les tâches de chaque développeur, nous avons préalablement déterminé l'ensemble des accès nécessaires à l'exploitation des ressources. Pour ce faire, nous avons utilisé des tickets contenant un descriptif détaillé en format Markdown.

Information sur la route

Route	Méthode HTTP	Params URL	Authentification	Action
/userId	GET		false	Récupérer les informations de l'utilisateur connecté

- Route : <http://localhost:3000/api/users/:userId>

Peut contenir les filtres suivant : (1) `:name=` | Retourner le nom d'un utilisateur `:username = [STRING]` recherche sur le `firstName+ 'lastName` de l'utilisateur. (2) `:limit=` permet de modifier les limite de remonté des informations. Limit de bases des utilisateurs est à 50.

- Success Response

```
{  
  "message": "Il y a X des utilisateurs",  
  "count": "int",  
  "length": "int",  
  "data": [  
    {  
      "id": 1,  
      "firstname": "Ahmed",  
      "lastname": "Magassouba"  
    },...  
  ]  
}
```

Screenshot d'un ticket Github servant à la réalisation du développement d'un composant d'accès au donnée

2. Nous avons notamment fait en sorte que des tests unitaires soient associés aux composants, avec une double approche fonctionnelle et sécurité.

```
> app-mobile-chat@1.0.0 test  
> jest  
  
PASS  test/participantMe.test.js  
  ✓ Récupération des channels avec un token invalide (151 ms)  
  ✓ Récupération des channels de l'utilisateur connecté (23 ms)  
  ✓ Vérifie si les données sont au bon (10 ms)  
  
Test Suites: 1 passed, 1 total  
Tests:       3 passed, 3 total  
Snapshots:  0 total  
Time:        0.999 s, estimated 1 s  
Ran all test suites.  
PS C:\Users\bougu\sent-chat\back> 
```

DOSSIER PROFESSIONNEL (DP)

Screenshot du passage de test sur la route participant/me

```

const axios = require("axios");
const jwt = require("jsonwebtoken");
require("dotenv").config();

const token = jwt.sign({ id: 5, role: "GUEST" }, process.env.JWT_KEY, {
  expiresIn: process.env.JWT_EXPIRES_IN,
});

test("Vérifie si les données sont au bon format", async () => {
  const response = await axios.get(
    "http://localhost:3000/api/participants/me",
    {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    }
  );

  const channels = response.data.data;
  // Vérifiez que les données sont bien un tableau
  expect(Array.isArray(channels)).toBe(true);

  // Vérifiez chaque channel dans les données
  channels.forEach((channel) => {
    expect(channel.sendAt).toBeDefined();
    expect(channel.sendAt).toMatch(
      /\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}\.\d{3}Z/
    );
    expect(typeof channel.channelId).toBe("number");
    expect(typeof channel.channelName).toBe("string");
    expect(typeof channel.LastMessage).toBe("string");
  });
});

```

exemple du code du test unitaire en question

Nous avons notamment testé systématiquement les routes de manière à s'assurer de leur efficacité.

Méthode de la requête

Route Testée

GET http://localhost:3000/api/participants/me Send

Query Headers 2 Auth 1 Body Tests Pre Run

None Basic Bearer OAuth 2 NTLM AWS

Bearer Token

eyJhbGciOiUzI1NlslNsCcI6ikpXCVj9eyJpZC16NSwicm9sZSI6IiVRViIiCjpxYXQiOjE20DkzODEmMTEslmV4tC16MTY4OTMMTA3Mx0H2pIyH2EfjWMjWmIwS_wvzraRLL6w0J1xy2HtIgIQ

Token Prefix Bearer

Status: 200 OK Size: 622 Bytes Time: 29 ms

Response Headers Cookies Results Docs

```

1  {
2   "message": "Liste des channels où l'utilisateur id = 5 participe",
3   "length": 2,
4   "data": [
5     {
6       "sendAt": "2023-01-16T11:00:53.000Z",
7       "channelId": 3,
8       "channelName": "Jhon Doe, Jeanne Doe...",
9       "LastMessage": "Jhon Doe : Jhon Doe vient de créer cette discussion !"
10    },
11    {
12      "sendAt": "2023-01-16T11:00:52.000Z",
13      "channelId": 2,
14      "channelName": "Jhon Doe, Ahmed Magassouba, Boris TIKHOMIROFF",
15      "LastMessage": "Ahmed Magassouba : On pourrait essayé de faire autrement de manière à s'assurer que tout fonctionne bien !"
16    },
17    {
18      "sendAt": "2023-01-16T11:00:49.000Z",

```

PROBLEMS OUTPUT TERMINAL SQL CONSOLE GITLENS DEBUG CONSOLE

POST /api/users/authenticate
GET /api/participants/me
AUTH TIME Sat Jul 15 2023 02:30:20 GMT+0200 (heure d'été d'Europe centrale)
[15/07/2023 02:30:20] GET /me
req_decoded { id: 5, role: 'USER', iat: 1689381011, exp: 1689381071 }

Status: 200 OK Size: 622 Bytes Time: 29 ms

Response Headers Cookies Results Docs

Documentation

Récupérer tous les channels où l'utilisateur connecté participe

Information sur la route

Route	Méthode HTTP	Params URL	Authentification	Action
/participants/me	GET		true	Récupérer tous les channels où l'utilisateur connecté participe

- arrow_right: TOKEN REQUIRED
- Pour être effectuer convenablement cette requête doit allé récupérer le dernier message envoyé associé à l channelId dans la table conversation .
- Success Response:

```
{
  "message": "Liste des channels où l'utilisateur id = 5 participe",
  "length": "int",
  "data": [
    {
      "sendAt": "DATETIME",
      "channelId": "int",
      "channelName": "string",
      "LastMessage": "string"
    },
    ...
  ]
}
```

screenshot des test d'API

DOSSIER PROFESSIONNEL (DP)

3. Nous avons veillé à ce que **les composants d'accès à la base de données respectent les règles de sécurisation reconnues** en utilisant une librairie Object-Relational Mapping (ORM) qui met régulièrement à jour ses principes de sécurité.

- > validation des entrées et la prévention des attaques par injection SQL
- > gestion des transactions



4. Nous avons accentué nos démarches de recherche afin de **résoudre les problèmes techniques liés à son utilisation et de mettre en œuvre de nouvelles fonctionnalités** en s'intéressant aux bonnes pratiques.

Ce fut notamment le cas avec la mise en place de deux token **JWT** en utilisant la documentation officielle : <https://jwt.io/introduction> ainsi que les recommandation d'usage par **auth0** : <https://auth0.com/docs/get-started/architecture-scenarios/mobile-api/part-1>

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

//Création du refresh token
const refreshToken = jwt.sign(
  { id: user.id, role: user.role },
  process.env.JWT_REFRESH_KEY,
  { expiresIn: process.env.JWT_REFRESH_EXPIRES_IN }
);
```

Extrait de code sur la fonction permettant de signer un JWT

DOSSIER PROFESSIONNEL (DP)

5. Nous nous sommes notamment concentrés sur la veille des vulnérabilités connues, ce qui nous a permis d'identifier et de corriger des failles potentielles.

Je prendrais notamment comme exemple ici le cas de Sequelize sur lesquels nous avons approfondie les failles de sécurité connue et référencée sur le site **CVE Details** : https://www.cvedetails.com/vulnerability-list/vendor_id-18114/product_id-46449/Sequelizejs-Sequelize.html de manière à s'assurer que l'accès à notre base de donnée ne soit pas compromis.

Nous avons dû à ce titre sécuriser toutes les entrées de fonctions query() qui présentent probablement un risque d'injection SQL.

```
const getAllChannelsByConnectedUser = async (req, res) => {
  // Participants.findAll({
  //   where: {},
  // });
  const { id } = req.decoded;
  // on s'assure que id est un nombre
  if (isNaN(id)) {
    const message = "L'identifiant de l'utilisateur doit être un nombre";
    return res.status(400).json({ message });
  }

  DB.sequelize
    .query()
    'SELECT ...'
```

Partie de code représentant un fonction de sécurité en amont de la requête à effectuer

6. Afin de s'assurer que la documentation technique liée aux technologies associées soit comprise, nous plançons les liens ou description dans un dossier Notion accessible à tous

Sent app | Documentation

Outils principaux <ul style="list-style-type: none">Agenda CommunMaquette Figma	Ressources <ul style="list-style-type: none">Bien gérer ses READMEUtilisation de Mermaid dans les MDDocumentation Expo GoReduxWebSocketCommit conventionWorkFlow Ux Process
Organisation <ul style="list-style-type: none">Cahier des chargeDiagrammes de GantWorkflow AgileModèle Meurise Conception ase de donnéeMerge RequestInstallation Back-EndInstallation Front-EndRoutesUseCase Github kanban	Réflexion <ul style="list-style-type: none">PopSQL (Client SQL)NativeWindStorybook (Frontend tools for UI development)Choix de la techno

ScreenShot de notre espace partagé sur expo

DOSSIER PROFESSIONNEL (DP)

✓ L'ensemble du processus d'activité suivante couvre les compétences professionnels attendue **Développer des composants d'accès aux données**

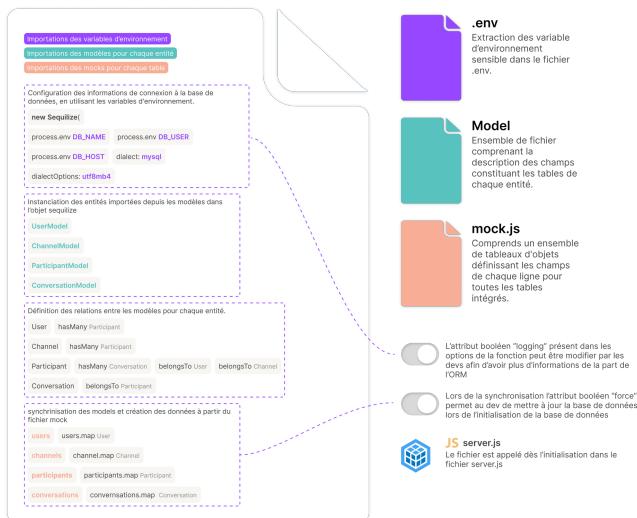
Développer la partie back-end d'une interface utilisateur web

Lors de la création de notre back-end, nous avons décidé de suivre les principes d'une API REST. Pour cela, nous avons procédé à une séparation claire entre la partie back-end et la partie front-end de notre application. Cette approche nous a permis de favoriser la cohérence dans notre architecture et de faciliter la répartition des tâches au sein de notre équipe de développement.

1. Nous nous sommes assuré que **bonnes pratiques de développement objet sont respectées** lors de l'usage de ces dernier

Nous avons opté pour le développement de notre back-end dans **l'environnement d'exécution Node.js**. Pour faciliter l'utilisation de Node.js, nous avons installé la librairie **Express.js** à l'aide de notre **gestionnaire de dépendances NPM** (Node Package Manager). Cela nous a permis de mettre en place rapidement une architecture et un environnement favorables à l'implémentation des principes REST en utilisant le **CLI (Command Line Interface) express-generator**.

Par la suite, à l'aide de **Sequelize**, nous avons créé des **modèles** de classe qui répondent aux **principes de la Programmation Orientée Objet (POO)**. Ces modèles de classe ont été utilisés pour définir les **entités** de notre application, en les représentant comme des **objets JavaScript**.



DOSSIER PROFESSIONNEL (DP)

Détail de l'instanciation des Modèles dans notre API

```
*****  
/* 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",  
});
```

Extrait d'une partie du code de config/db.js et usage des Modèles Sequelize

2. Nous avons développé des middlewares d'authentification assurant que **les composants serveur contribuent à la sécurité de l'application**.

Dans notre architecture, nous avons défini trois fonctions fondamentales pour l'utilisation du token JWT :

1. La fonction **CheckToken** : Cette fonction est responsable de la vérification de la validité du token. Elle s'assure que le token n'a pas été altéré et qu'il n'a pas expiré. Si le token est invalide, l'accès à la route est refusé.
2. La fonction **checkIsAdmin** : Cette fonction est réservée aux routes accessibles uniquement aux administrateurs. Elle vérifie si l'utilisateur qui fait la requête possède les autorisations nécessaires pour accéder à la route. Si l'utilisateur n'est pas un administrateur, l'accès est refusé.
3. La fonction **refreshToken** : Cette fonction permet de rafraîchir le token en fonction de la validité du refreshToken. Lorsque le token expire, le refreshToken peut être utilisé pour obtenir un nouveau token valide sans nécessiter une nouvelle connexion. Cela permet une expérience utilisateur fluide et sécurisée.

DOSSIER PROFESSIONNEL (DP)

En intégrant ces fonctions dans nos routes, nous assurons un contrôle d'accès sécurisé et granulaire, en autorisant uniquement les utilisateurs authentifiés et autorisés à accéder aux ressources appropriées. Cela garantit la confidentialité et l'intégrité des données, ainsi que la protection contre les accès non autorisés.

```
back > src > middlewares > tokenMiddleware.js > ...
...
1 import dotenv from "dotenv";
2 import jwt from "jsonwebtoken";
3 import DB from "../config/db.js";
4
5 const User = DB.User;
6
7 dotenv.config();
8
9 > const checkToken = (req, res, next) => { ...
41   };
42
43 > const checkIsAdmin = (req, res, next) => { ...
51   };
52
53 > const refrechToken = async (req, res, next) => { ...
99   };
100
101 export { checkToken, checkIsAdmin, refrechToken };
102
```

Extrait de code dans l'usage fonctionnel d'un middleware

```
*****
/* USER AUTHENTICATE ROUTES */
*****
router.post("/refreshToken", refrechToken);
router.get("/private", checkToken,
getAllAuthentified);
router.delete("/trash/me", checkToken,
deleteTrashOne);
router.get("/me", checkToken, getMe);
router.put("/me", checkToken, updateMe);
router.delete("/me", checkToken, deleteMe);
router.get("/:userId", checkToken, getOne);
```

Extrait de code sur l'usage des middlewares dans l'appel de nos routes

DOSSIER PROFESSIONNEL (DP)

3. Afin de garantir une bonne compréhension générale des fonctionnalités, nous avons veillé à ce que le **code source des composants soit documenté ou auto-documenté**.

Pour atteindre cet objectif, nous avons établi **des conventions** au préalable. Pour documenter les fonctions essentielles, nous avons utilisé **JSDoc**, qui est intégré dans **Visual Studio Code**.

```
/**  
 * Récupère la valeurs d'une clef dans l'AsyncStorage  
 * @param {string} item nom de clef recherché dans le storage  
 * @returns {object} retourne un l'object recherché sous forme clef/valeur  
 */  
const getItemStorage = async (item) => {  
  const data = await AsyncStorage.getItem(item);  
  return JSON.parse(data);  
};  
  
/**  
 * Supprime la clef/valeur d'un objet dans l'asyncStorage  
 * @param {string} item  
 * @returns {void}  
 */  
const removeItemStorage = async (item) => {  
  try {  
    await AsyncStorage.removeItem(item);  
    console.log("item removed");  
  } catch (e) {  
    console.log(e);  
  }  
};
```

Détail de fonction documenté avec jsdoc

4. Notre **démarche de recherche** sur l'usage d'Express **nous a permis de résoudre les divers problèmes techniques liés à son utilisation et de mettre en œuvre de nouvelles fonctionnalités**.

Comme nous avons pu voir précédemment, le bon usage de l'API REST dans un environnement Node.js avec Express.js vient principalement de leur documentation officielle très complète sur **Expressjs.com** :

<https://expressjs.com/en/guide/using-middleware.html>

Nous avons notamment pu nous appuyer sur des ressources open source pour comparer notre organisation à d'autres organisations existantes grâce à des ressources sur **Medium**.

<https://selvaganesh93.medium.com/how-node-js-middleware-works-d8e02a936113>

Notamment en comparant des exemple existant sur **Github** :

<https://github.com/thombergs/code-examples/tree/master/nodejs/middleware>

DOSSIER PROFESSIONNEL (DP)

5. Notre **veille sur les vulnérabilités** connues nous a notamment permis d'**identifier et corriger des failles potentielles venant d'express**.

En effet, lors de la mise en place de notre API REST, nous avons remarqué qu'Express préconisait l'utilisation d'une librairie supplémentaire pour assurer une couche de sécurité supplémentaire : la librairie **Helmet**.

À l'aide de notre gestionnaire de dépendances, nous avons donc installé la librairie Helmet :

```
npm install --save helmet
```

En intégrant Helmet à notre application Express, nous avons renforcé la sécurité de notre API en protégeant notre application contre certaines vulnérabilités et attaques courantes, telles que les attaques par injection de scripts, les attaques XSS (Cross-Site Scripting) et d'autres menaces potentielles liées aux en-têtes HTTP.

```
import helmet from "helmet";
import * as dotenv from "dotenv";
dotenv.config();
...
//couche de sécurité supplémentaire
app.use(helmet(
  {
    contentSecurityPolicy: {
      directives: {
        defaultSrc: ["'self'", process.env.CONTENTSECURITYSOURCE],
        scriptSrc: ["'self'", process.env.CONTENTSECURITYSOURCE],
        objectSrc: ["'none'"],
      },
      xssFilter: { setOnOldIE: true }
    },
  }
));
```

Extrait de code lié à l'usage du middleware helmet

6. À chaque usage d'une librairie, nous en informions l'intégralité de l'équipe et nous mettions les ressources nécessaires à leur compréhension à disposition dans **Notion** ou dans la colonne "Ressources" de notre **board Kanban** sur notre **github Project**.

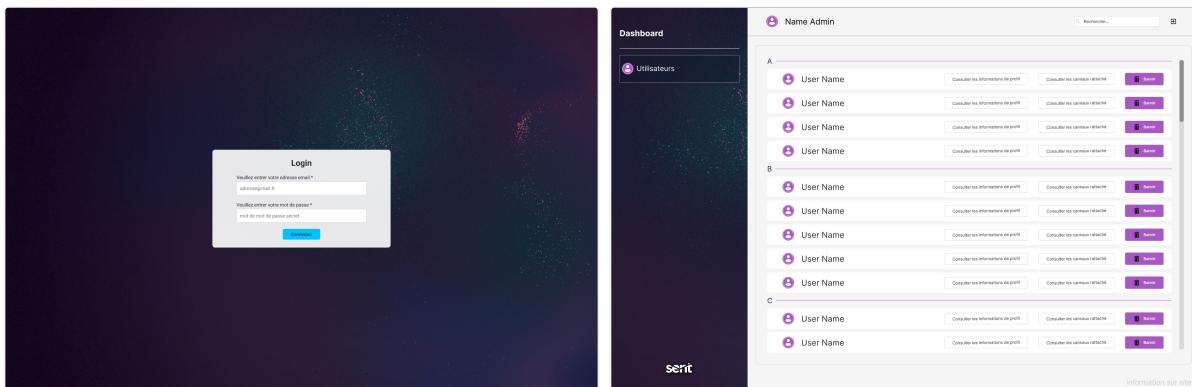
DOSSIER PROFESSIONNEL (DP)

- ✓ L'ensemble du processus d'activité suivante couvre les compétences professionnels attendue **Développer la partie back-end d'une interface utilisateur web**

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

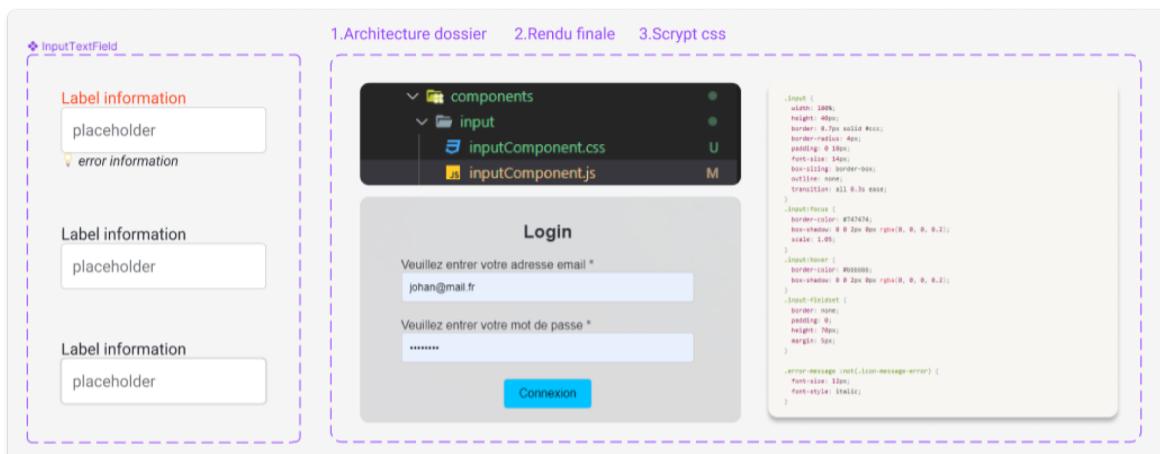
Lors de la réalisation de cette application, nous avons dû concevoir un panel admin pour permettre aux administrateurs d'avoir un contrôle sur leurs applications. Pour ce faire, nous avons méticuleusement suivi les recommandations liées à la sécurité et aux bonnes pratiques du web.

1. Afin de rendre **l'interface** adaptée à son usage, nous avons créé une **maquette** pour développer le front-end en **conformité** avec celle-ci.



Screenshot de deux frames venant de figma

Pour ce faire, j'ai utilisé la librairie React.js afin de développer des composants selon une approche modulaire. Chaque composant reprend le code d'esthétisme de la maquette.



DOSSIER PROFESSIONNEL (DP)

Dans la figure de gauche, la conception du composant dans Figma. Dans la figure de droite, le résultat lors du développement.

J'ai profité du fait que React précompile le CSS et j'ai divisé les feuilles de style en cascade liées à chaque composant. Cela a rendu la lisibilité des développeurs plus accessible, car ils peuvent facilement localiser et gérer le style spécifique à chaque composant.

2. Nous avons tenu à respecter **les bonnes pratiques de développement** à travers une sémantique adaptée ainsi que **les règles d'accessibilité**.

```
<fieldset className="input-fieldset" style={this.state.style}>
  <label
    className="input-label"
    style={{ color: this.state.error ? "red" : "#282c34" }}
    htmlFor={this.state.id}
  >
    {this.state.label}
    {this.state.required ? "*" : null}
  </label>
  <input
    type={this.state.type}
    placeholder={this.state.placeholder}
    name={this.state.name}
    required={this.state.required}
    disabled={this.state.disabled}
    onChange={this.state.onChange}
    className={this.state.className}
    id={this.state.id}
    onBlur={this.state.onBlur}
    onFocus={this.state.onFocus}
  />
  {this.state.error !== "" && (
    <span className="error-message">
      <span
        className="icon-message-error"
        role="img"
        aria-label="icon information"
        style={{ display: this.state.error ? "initial" : "none" }}
      >
        
      </span>
      {this.state.errorMessage}
    </span>
  )}
</fieldset>
```

Extrait de code JSX du composant *inputComponent.js*

Nous pouvons constater dans l'exemple suivant que la structure sémantique de l'input respecte les règles de recommandation du World Wide Web Consortium (W3C),

DOSSIER PROFESSIONNEL (DP)

notamment en ce qui concerne l'usage destiné à la lecture des malvoyants.

Nous nous sommes employés à **définir un ensemble de conventions** afin de nous assurer de la compréhension de tous concernant le travail d'autrui. À cet égard, nous avons favorisé la **documentation** du texte sur les parties sensibles, utilisé des noms de variable explicites et **utilisé JSDoc** sur les fonctions pour faciliter cette approche.

```
const service = axios.create({
  baseURL: process.env.BASE_URL_API ,
  timeout: 5000,
  headers: {
    "Content-Type": "application/json",
    // WARNING : cors "*" seulement pour le dev
    "Access-Control-Allow-Origin": "*",
    //méthodes autorisées par l'api
    "Access-Control-Allow-Methods": "GET, POST, PATCH, PUT, DELETE, OPTIONS",
    //headers autorisés par l'api
    "Access-Control-Allow-Headers": "Origin, Content-Type, X-Auth-Token",
  },
});

...
/** 
 * Add a response interceptor
 * @param response
 * @returns {Promise<*>}
 * @private
 * @description
 * Intercepte la réponse de l'api et retourne la réponse
 * Ajoute le token dans le header de la requête
 */
service.interceptors.response.use((response) => {
  if (response.status === 200 && response.data.data.token !== undefined) {
    // alors on ajoute le token dans le header
    service.defaults.headers.common[
      "Authorization"
    ] = `Bearer ${response.data.data.token}`;
  }
  return response;
});|
```

Extrait de code documenté sur des parties sensibles

3. Nous avons mis en place des **tests end-to-end** afin de nous assurer de la cohérence avec le comportement attendu lors de la navigation.

DOSSIER PROFESSIONNEL (DP)

Pour ce faire, nous avons mis en place Cypress dans notre projet. Grâce à son interface facile d'accès, nous avons pu rapidement prendre le contrôle des comportements indésirables. Cypress nous offre une suite complète d'outils de test automatisés qui nous permet de simuler les actions de l'utilisateur sur l'interface et de vérifier le comportement de l'application en temps réel.

```
describe("Module de Connexion", () => {
  it("Connexion réussie avec des identifiants valides", () => {
    cy.visit("http://localhost:3001/"); //page d'accueil de l'application

    // Vérifiez que le formulaire de connexion s'affiche
    cy.contains("Login");

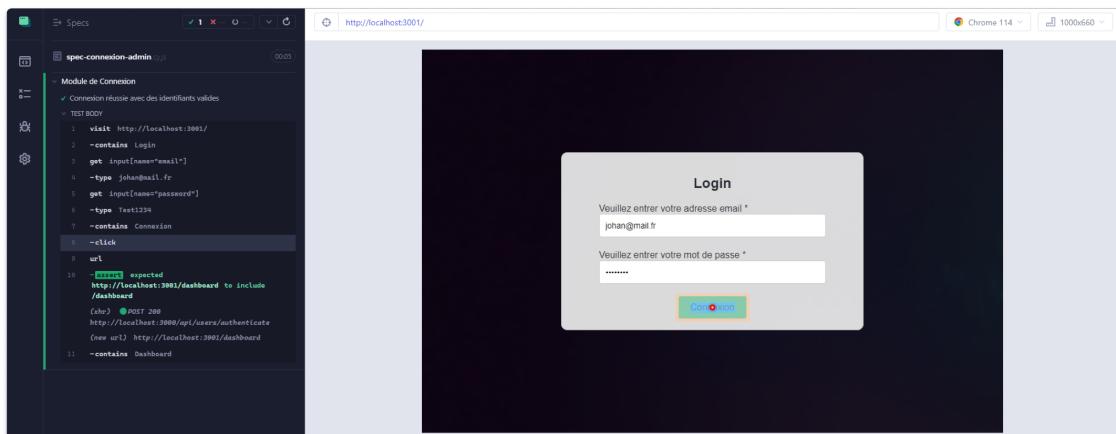
    // Remplissage du formulaire de connexion
    cy.get('input[name="email"]').type("johan@mail.fr");
    cy.get('input[name="password"]').type("Test1234");

    // INteraction avec le bouton de connexion
    cy.contains("Connexion").click();

    // l'utilisateur est redirigé vers le dashboard après la connexion réussie
    cy.url().should("include", "/dashboard");

    // l'utilisateur est connecté en vérifiant
    // qu'un élément spécifique s'affiche après la connexion.
    cy.contains("Dashboard"); //but
  });
});|
```

Exemple de test n2n passé avec cypress sur le module de connexion



Screenshot de l'interface Cypress avec le test n2n réalisé

DOSSIER PROFESSIONNEL (DP)

4. Afin de nous assurer **de la sécurité** du site, nous avons suivi un ensemble de normes et de standards en la matière.

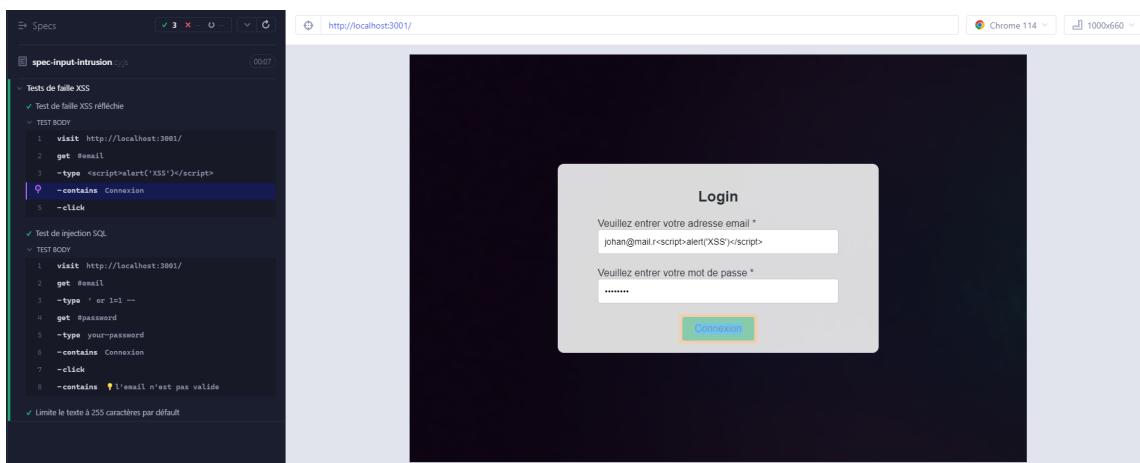
Nous avons sécurisé les redirections en utilisant la librairie React Router DOM. Cela garantit que seules les routes déclarées et satisfaisant certaines conditions peuvent être accessibles.

```
//on bloque l'accès au dashboard si l'utilisateur n'est pas connecté
function App() {
  const { auth, role } = React.useContext(AuthContext);

  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route
          path="/dashboard"
          exact={true}
          element={auth && role ? <Dashboard /> : <Navigate to="/" />}
        />
        <Route path="*" element={<Navigate to="/" />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Extrait de code dans le fichier App.js expliquant la structure du router

J'ai renforcé la sécurité de mes tests en effectuant des simulations d'injections lors de mes tests de niveau 2 à travers l'input de connexion. Ces tests visent à détecter et prévenir des vulnérabilités telles que les attaques XSS (Cross-Site Scripting) et les injections SQL.



DOSSIER PROFESSIONNEL (DP)

J'ai développé une bibliothèque de regex personnalisée pour renforcer la sécurité lors de l'insertion de données dans les entrées de mon application. Cette bibliothèque utilise des expressions régulières spécifiques pour valider et filtrer les données saisies par les utilisateurs, permettant ainsi de prévenir les attaques malveillantes.

```
/**  
 * Verifie si la valeur est un caractère spécial  
 * @param {*} value - valeur à tester  
 * @returns {boolean} Retourne true si la valeur est un caractère spécial  
 * @example  
 * isSpecialcaractereEntry("a") => false  
 * isSpecialcaractereEntry("1") => false  
 * isSpecialcaractereEntry("@") => true  
 */  
function isSpecialcaractereEntry(value) {  
    const regex = /^[^a-zA-Z0-9]/g;  
    return regex.test(value);  
}  
  
/**  
 * Verifie si la valeur est un nombre  
 * @param {*} value - valeur à tester  
 * @returns {boolean} Retourne true si la valeur est un nombre  
 * @example  
 * isNumberEntry("1") => true  
 * isNumberEntry("a") => false  
 */  
function isNumberEntry(value) {  
    const regex = /[0-9]/g;  
    return regex.test(value);  
}  
  
/**  
 * Verifie si la valeur est une lettre  
 * @param {*} value - valeur à tester  
 * @returns {boolean} Retourne true si la valeur est une lettre  
 * @example  
 * isLetterEntry("a") => true  
 * isLetterEntry("1") => false  
 * isLetterEntry("A") => true  
 */  
function isLetterEntry(value) {  
    const regex = /[a-zA-Z]/g;  
    return regex.test(value);  
}  
...|
```

Extrait des fonction disponibles dans rateRegex.js

DOSSIER PROFESSIONNEL (DP)

5. La démarche de recherche m'a permis de résoudre le problème technique liées au Routeur.

En effet, React étant une librairie Javascript, sa structure est orientée CSR (client Side Rendering). Il est donc important de construire un routeur qui permet la mise à jour et le contrôle des entrées dans L'URL. A l'aide de la communauté github et Medium et en prenant soin de m'approcher d'articles parlant de bonne pratique. J'ai pu identifier l'avantage de la librairie React Router DOM.

J'ai pu ainsi mettre en place un routeur permettant une certaine sécurité sur la navigation.

J'ai déplacé linstanciation de mon contexte au niveau de lindex afin quil puisse être utilisable dans le fichier app.js.

Une fois ayant accès à létat du contexte, j'ai pu conditionner lappel des routes selon létat de lutilisateur.

```
//on bloque l'accès au dashboard si l'utilisateur n'est pas connecté
function App() {
  const { auth, role } = React.useContext(AuthContext);

  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route
          path="/dashboard"
          exact={true}
          element={auth && role ? <Dashboard /> : <Navigate to="/" />}
        />
        <Route path="*" element={<Navigate to="/" />} />
      </Routes>
    </BrowserRouter>
  );
}
```

6. Durant ma veille sur les vulnérabilités connues, j'ai identifié des failles potentielles sur notre panel Admin.

En effet, en lisant un manuel de bonne pratique de sécurité, celui-ci parlé des failles CSRF (Cross-Site Request Forgery), vulnérabilité de sécurité qui permet à un attaquant de forcer un utilisateur authentifié à effectuer des actions non voulues sur un site web sans son consentement. Cela se produit lorsque le site web ne vérifie pas lorigine des requêtes, permettant à lattaquant de soumettre des requêtes légitimes à lutilisateur

DOSSIER PROFESSIONNEL (DP)

sans qu'il le sache.

A l'inverse d'un token d'authentification qui s'assure de la validité de la session. Le X-Token-CSRF est une simple preuve de la validité de l'origine de la requête. Il est conseillé de l'appliquer dans les cas sensibles sur des requêtes modifiant l'état de notre application tel que les POST, UPDATE et DELETE.

Cela a impliqué des modifications dans notre back-end. Nous avons dû créer de nouveaux middlewares. Et s'assurer que le retour des réponses comporte le Token en **ReadOnly**.

```
JavaScript ▾
Copier Légende ...
```

```
/*
 * Crée un token X-CSRF-Token
 * @return {Promise<*>}
 * Crée un token X-CSRF-Token
 */
const createXCSRFToken = async (req, res, next) => {
    const token = crypto.randomBytes(64).toString("hex");
    res.cookie("XSRF-TOKEN", token);
    res.locals.csrfToken = token;
    next();
};

/*
 * Vérifie le token X-CSRF-Token
 * @return {Promise<*>}
 * Si le token est valide, on passe au middleware suivant
 * Sinon on retourne une erreur 401
 */
const verifyXCSRFToken = async (req, res, next) => {
    const csrfToken = req.cookies["XSRF-TOKEN"];
    const xsrfToken = req.headers["x-xsrf-token"];
    if (csrfToken !== xsrfToken) {
        const message = `Token CSRF invalide...`;
        return res.status(401).json({ message });
    }
    next();
};
```

Partie de code du middlewares servant à gérer le X-token-CSRF partie back-end

Coté Front : A la réception du Token on le passe dans le header. Il se trouve désormais dans les Cookie en http readOnly et dans l'entête HTTP

DOSSIER PROFESSIONNEL (DP)

```
/**  
 * Function permettant de set le X-CSRF-Token et le mettre dans le header  
 * @returns {Promise<*>}  
 * @private  
 * @description  
 * Récupère le X-CSRF-Token  
 * @example  
 * const token = await getXCSRFToken();  
 * console.log(token);  
 * // retourne le token  
 */  
const setCSRFToken = async () => {  
    const response = await service.get("/csrf-token");  
    const token = response.data.data;  
    service.defaults.headers.common["X-CSRF-Token"] = token;  
    return token;  
};
```

 L'ensemble du processus d'activité suivante couvre les compétences professionnels attendue **Développer la partie front-end d'une interface utilisateur web**

2. Précisez les moyens utilisés :

- La réalisation de la maquette a été réalisé avec le logiciel en ligne **Figma**
- La réalisation des tests unitaires ont été réalisé avec **jest**
- La réalisation des test d'API fonctionnelles ont été réalisé avec **Thunder Client**
- Les composants d'accès à la base de données ont été réalisés à l'aide de **Sequelize**
- La veille technologique présenté sur l'usage des fonctionnalité du JWT a été effectuée sur :
 - JWT (JSON Web Tokens) : <https://jwt.io/>
 - auth0 : <https://auth0.com/>
 - Github repository (expo-jwt): <https://github.com/blake-simpson/expo-jwt>
- Les veilles technologiques sur la sécurité effectuées sur la partie sécurité ont été pratiquées sur les sites anglophones suivants :
 - OWASP (Open Web Application Security Project) : <https://owasp.org/>
 - CVE (Common Vulnerabilities and Exposures) : <https://cve.mitre.org/>
- La réalisation d'une documentation communautaire a été effectuée sur **Notion & Github Project**
- La réalisation de notre API REST a été réalisé à l'aide de l'environnement d'exécution **Node.js** et la librairie **express js**

DOSSIER PROFESSIONNEL (DP)

- La gestion des failles de sécurité a été améliorée avec **Helmet**
La veille technologie sur la mise en place de l'environnement de notre API REST a été pratiquée sur :
 - Expressjs.com : <https://expressjs.com/en/guide/using-middleware.html>
 - Divers articles medium comme <https://selvaganesh93.medium.com/how-node-js-middleware-works-d8e02a936113>
 - A travers des exemple comme <https://github.com/thombergs/code-examples/tree/master/nodejs/middleware>
- La réalisation de notre panel ADMIN, interface WEB a été réalisé avec la librairie **REACT**
- Les test n2n ont été réalisé à l'aide de **Cypress**
- La structuration du router sur le front-End du panel Admin a été réalisé avec la librairie **React-Router**
 - La documentation du router sur le front-End du panel Admin a été pratiqué sur :
 - Medium: <https://algfry12.medium.com/react-router-best-practices-9c564388f4d3>
 - Documentation officielle : <https://reactrouter.com/en/main>
- La documentation sur la veille de sécurité a été faite sur : [file.html \(cni.es\)](#)

3. Avec qui avez-vous travaillé ?

Nous avons travaillé en groupe de 4 avec Boris TIKHOMIROFF, Cyril Porez et Ahmed Magassouba, tous trois étudiants au sein de l'école La Plateforme.

4. Contexte

Nom de l'entreprise, organisme ou association	► <i>Ecole la Plateforme</i>
Chantier, atelier, service	► <i>Réalisation d'une application mobile</i>
Période d'exercice	► Du : <i>04/01/2023</i> au : <i>07/02/2024</i>

5. Informations complémentaires (facultatif)

Tenant pour référence mon projet de fin d'année, des éléments facultatifs concernant l'activité ainsi présentés sont disponibles dans l'article suivant:

<https://www.notion.so/Sent-Dossier-Professionnel-389ea4e931ef46eda6e724ae5d120b14?pvs=4>

DOSSIER PROFESSIONNEL (DP)

Exemple n° 2 - Sokoban

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

Développer une interface utilisateur de type desktop

Au cours de notre formation académique, une partie du cursus était destiné à la compréhension de langage destiné au développement logiciel. Dans le cadre d'un exercice pratique, nous avons dû développer un jeu répondant aux règles du Sokoban.



Pour améliorer les conditions de développement du jeu, j'ai choisi d'utiliser la bibliothèque Pygame en Python. Le comportement du setup est simple : une fois que Pygame est instancié, une boucle while est mise en place pour exécuter en continu le jeu. Pour sortir de cette boucle, j'ai défini des règles spécifiques qui, lorsqu'elles sont satisfaites, mettent la boucle en état "True" et permettent ainsi au jeu d'être quitté de manière contrôlée.

```
# pygame setup
pygame.init()
screen = pygame.display.set_mode((1280, 720))
clock = pygame.time.Clock()
running = True

while running:
    ...

    pygame.quit() |
```

Exemple de setup donné par la documentation pygame

Pour la création des niveaux , j'ai pris comme référence les niveaux créés en [1999 par David Skinner pour le jeu Sasquatch](#). Chaque niveau est récupéré sous forme de chaîne de caractères. J'ai ensuite intégré ces chaînes de caractère dans un glossaire sous forme de tableau comportant des tulles sous format ("lvl_name","lvl_string"). Chaque tulles comporte donc un niveau du jeu et chaque caractère du lvl_string sont des cases servant à générer la

DOSSIER PROFESSIONNEL (DP)

map.

Les caractères sont: # : mur, \$: caisse , . (point) : emplacement, @ : joueur, * : caisse sur emplacement, + : joueur sur emplacement, (espace) : case vide

présentation du glossaire de level dans l'application Sokoban

GlossaryLevels sera utilisée pour fournir les données nécessaires pour créer un objet Level. À l'exception du Contrôle, qui est géré dans une classe séparée, toutes les autres classes interagissent entre elles. Cela nous permet de minimiser les interactions au sein de notre boucle while, rendant le code plus modulaire et lisible.

Dans notre boucle while, nous avons trois éléments principaux :

- **Le levels** : Le levels est un tableau d'object Level, cette classe met à jour l'état global du jeu en utilisant les données fournies par levels. Elle gère la logique du jeu, les collisions, les interactions entre les objets, etc.
 - **Le screen** : il est mis à jour en fonction de l'état de la matrice renvoyée par Level, ce qui permet de rafraîchir l'affichage graphique en fonction des changements dans le niveau du jeu.
 - **Le inputHandler** : cette classe récupère les entrées clavier déclenchées par l'utilisateur, ce qui permet de gérer les commandes et les interactions du joueur avec le jeu.

DOSSIER PROFESSIONNEL (DP)

Création de classes et la séparation des tâches.

En découpant mon code en classes, chaque classe est responsable d'une partie spécifique du jeu, ce qui rend le code plus organisé et facilement maintenable.

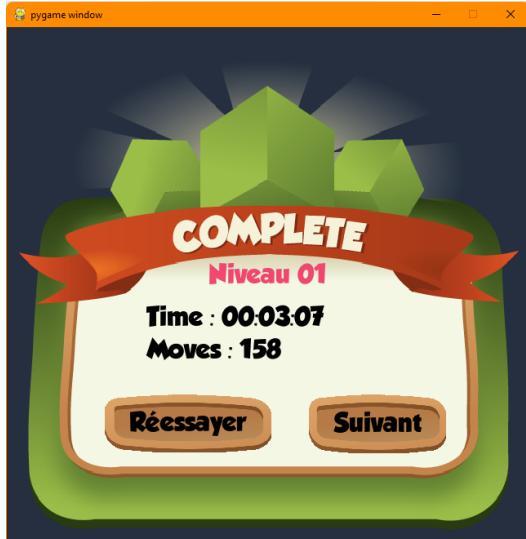
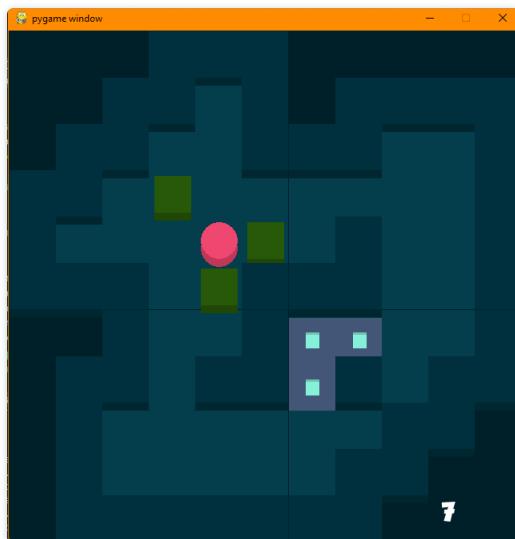
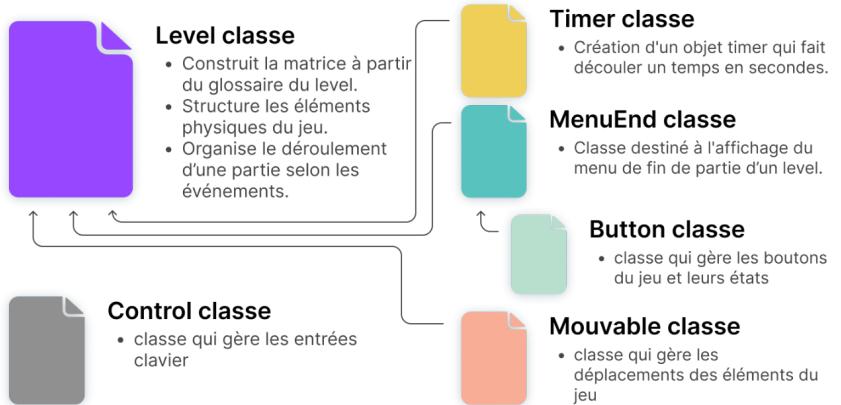
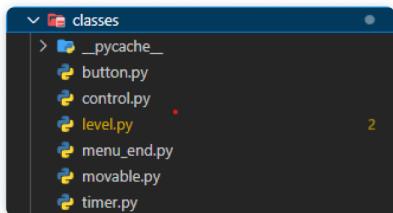


Figure 1 : Lancement d'une partie
Figure 2 : Menu de fin de partie affiché

Une fois les différentes classes mises en place, je me suis intéressé au comportement global de la boucle while.

DOSSIER PROFESSIONNEL (DP)

```
...
levels = []
for level in GlossaryLevels:
    levels.append(Level(level[0], level[1]))
...

while True: # Boucle infinie
    # Handle timer
    # Si le timer n'est pas démarré et que le niveau n'est pas gagné
    if(levels[current].timer_as_started() == False
        and levels[current].check_win() == False):
        # On instancie le menu de fin au début du niveau
        levels[current].set_menuEnd(screen)
        levels[current].set_Timer(screen) # On positionne le timer en bas à droite
        levels[current].start_Timer() # On démarre le timer
    for event in pygame.event.get(): # Parcours de la liste des événements reçus
        if event.type == pygame.QUIT: # Si un de ces événements est de type QUIT
            exit() # On arrête la boucle
    # Handle input
    for key, movement in axes.items(): # Pour chaque touche de direction
        if io.get_key(key): # Si la touche est enfoncée
            levels[current].movement_logic(movement) # On déplace le personnage
    # Handle reset
    if io.get_key(pygame.K_r): # Si la touche R est enfoncée
        levels[current].reset() # On réinitialise le niveau
    # Check win
    if levels[current].check_win() : # Si le niveau est gagné
        if current + 1 > len(levels) - 1: # Et que c'était le dernier niveau
            exit() # On quitte le jeu
        levels[current].stop_Timer() # On arrête le timer
        # On met à jour le menu de fin avant de le draw
        levels[current].update_score_menuEnd()
        levels[current].draw_menuEnd() # On affiche le menu de fin
        #on verifie si le bouton next du menu à été cliquer
        if levels[current].menuEnd.button_next.is_clicked() == True:
            current += 1
        if levels[current].menuEnd.button_restart.is_clicked() == True:
            levels[current].reset_hard()

    else :
        screen.fill((0, 32, 41)) # On remplit l'écran de couleur
        levels[current].render(screen) # On affiche le niveau

    pygame.display.flip() # On met à jour l'affichage
    clock.tick(60) # On limite le nombre d'images par seconde à 60
```

Partie du code dans la boucle while

DOSSIER PROFESSIONNEL (DP)

L'ensemble du processus d'activité suivante couvre les compétences professionnels attendue **Développer une interface utilisateur de type desktop**

2. Précisez les moyens utilisés :

- La réalisation de l'application a été réalisé le langage de programmation **Python**
- La réalisation des élément graphique a été réalisé avec le logiciel en ligne **Figma/ illustrator**
- La réalisation de l'application a été réalisé avec la bibliothèque **Pygame**
- La documentation liée à la réalisation des divers problématique **Pygame.org** :
<https://www.pygame.org/docs/>

3. Avec qui avez-vous travaillé ?

Ce travail a été réalisé seul

4. Contexte

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

Chantier, atelier, service ▶ *Réalisation d'un jeu vidéo à l'aide de Python : RunTrack Python*

Période d'exercice ▶ Du : 19/04/2023 au : 24/04/2023

5. Informations complémentaires (*facultatif*)

Activité-type

pe

2

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

Exemple n° 1 ▶ Cliquez ici pour entrer l'intitulé de l'exemple

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

DOSSIER PROFESSIONNEL (DP)

Avant propos

Cette Activité couvre les compétences professionnels suivante (certain axes d'analyse sont couvert par les présentations précédente):

- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
- Concevoir une application
- Construire une application organisée en couches
- Développer une application mobile
- Préparer et exécuter les plans de tests d'une application
- Préparer et exécuter le déploiement d'une application

Nous avons vu dans la première partie quelques cas pratiques lors de la création de Sent. À présent, nous allons détailler plus en profondeur l'aspect organisationnel et les phases de conception qui ont eu lieu en amont des Sprints.

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

1. **Le suivi des activités ou des tâches du projet a été mis en œuvre en selon notre démarche de travail** inspiré d'une alliance entre les principes SCRUM et les méthodes Crystal Clear.
 - a. **Le rôle du Scrum Master et la rotation au sein de l'équipe**
Chaque membre de l'équipe a l'opportunité de prendre en charge ce rôle, ce qui favorise une meilleure compréhension de la méthodologie Scrum
 - b. **Organisation interne pour favoriser la collaboration et l'efficacité**
Afin de maximiser la collaboration et l'efficacité au sein de l'équipe, nous avons décidé de fragmenter les équipes de développement de manière à ce que certains membres puissent se concentrer sur la recherche de solutions tandis que d'autres continuent à avancer sur les sprints
 - c. **Application de Daily Meetings**
En combinant les Daily Meetings pour une communication fluide et la communication osmotique pour surmonter les défis de la fatigue et de la compréhension, nous renforçons notre capacité à travailler de manière collaborative et à maintenir une

DOSSIER PROFESSIONNEL (DP)

compréhension globale du projet, même dans des conditions exigeantes.

d.La Cérémonie du Backlog pour ajuster les priorités

Chaque début de sprint, nous organisons la cérémonie du Backlog, une réunion où nous passons en revue les tâches à accomplir et redéfinissons les priorités en fonction de notre progression.

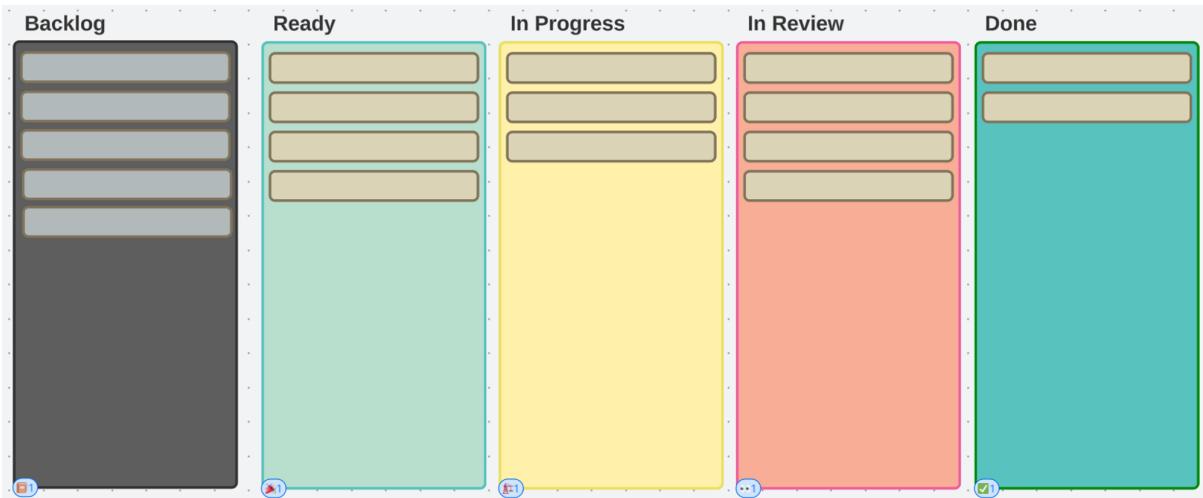
C.La rétrospective hebdomadaire pour tirer les leçons

En fin de semaine, nous nous réunissons pour la rétrospective hebdomadaire. Cette séance nous permet de faire le point sur ce qui a été réalisé.

d.Des sprints d'une semaine pour une progression itérative

e.Les User Stories : Une approche centrée sur les besoins des utilisateurs

Cette démarche ainsi présentée s'illustre à travers l'usage de l'**outil de projet** développé par **github**. Qui nous permet de manière collaborative de gérer notre projet sous la forme de **Kaban**. Chaque ticket ainsi créé peut être une **issue associé à un branche**. Nous permettant d'avoir une approche simple de l'avancée du projet à travers les **milestones**.



Structure de notre kaban

Les différents type de label utilisé dans notre méthodes :

DOSSIER PROFESSIONNEL (DP)

i Tutors

Le Labels tutors définit une issue comme étant un parentes a un ensemble de features à apportées. Il est souvent liées a un milestones.

i documentation

Le label documentations défini une issue destinées seulement offrir un ensemble de données étant utile aux développeurs. Elle rassemble des éléments centraux permettant a ces derniers d'y apposé des grande modification ou des changement majeurs devant être consulté par les autres.

i question

Le label question est dédier au discussions de groupes sur un sujet précis. Les sujets peuvent êtres, l'utilisations d'une bibliothèque, l'implémentation d'une nouvelle fonctionnalité ou encore la révision d'étapes organisée. Elle servent a concilié la volonté de chacun sur une vision commune de manière proactive.

i Back-End

Issues en relations avec le dossier back

i Front-End

Issue en relation avec le dossier front

i Feature

L'issue est associé a une amélioration

i Bug

Lier a un fix

i Duplicate

L'issue est un doublon et a été délaissé au profit d'une autre.

Lors de l'utilisation de duplicate on lie la nouvelle issue a celle-ci

- # [n°issue]

i Help Wanted

L'issue est associé a une amélioration L'issue est en attente d'informations supplémentaire

Evaluation des priorités

Lors de la mise de l'issue en "Ready", il est essentiel de comprendre l'importance de

DOSSIER PROFESSIONNEL (DP)

chaque niveau de priorité pour une meilleure planification et exécution du travail. Voici une explication de l'importance de chaque niveau :

Urgent

Les issues marquées comme urgentes nécessitent une attention immédiate et doivent être traitées en priorité absolue. Elles peuvent inclure des problèmes critiques ou des blocages majeurs qui doivent être résolus rapidement pour éviter tout impact négatif sur le projet ou les utilisateurs.

High

Les issues avec une priorité élevée ont un impact significatif sur le projet et doivent être traitées avec une attention particulière. Elles peuvent inclure des fonctionnalités essentielles, des améliorations importantes ou des corrections de bugs qui doivent être résolus rapidement pour maintenir la qualité et la progression du projet.

Medium

Les issues de priorité moyenne ont une importance modérée et doivent être traitées dans un délai raisonnable. Elles peuvent inclure des fonctionnalités ou des améliorations qui ne sont pas critiques mais qui apportent une valeur ajoutée au projet.

Low

Les issues de priorité basse ont une importance moindre et peuvent être traitées avec une priorité plus faible. Elles peuvent inclure des demandes de fonctionnalités mineures, des améliorations cosmétiques ou des tâches de maintenance qui peuvent être réalisées ultérieurement sans impacter de manière significative le projet.

Estimation de la vitesse

A la récupération d'une issue nous évaluons la vitesse de la tâche. ↓

DOSSIER PROFESSIONNEL (DP)

XLARGE

L'issue est complexe et risque de prendre tout le temps imparti du sprint pour être résolue, ou nécessite un temps supplémentaire pour sa résolution.

Large

L'issue est de taille importante mais réalisable dans les délais du sprint.

Medium

L'issue est de taille moyenne et ne présente pas de difficultés majeures, pouvant être réalisée dans un temps raisonnable.

Small

L'issue est de petite taille et peut être terminée en moins de 24 heures.

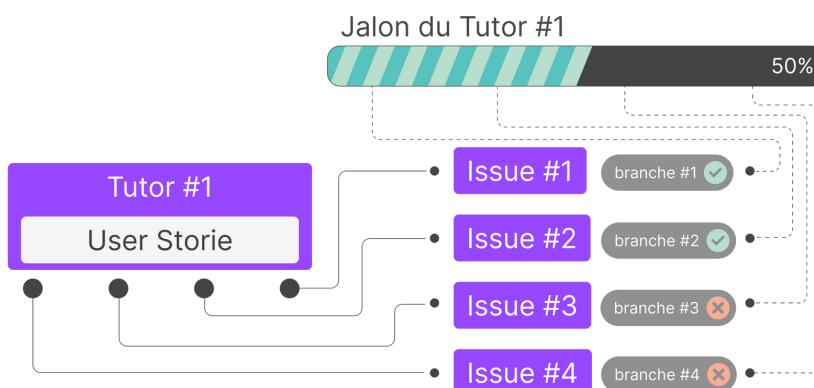
Tiny

L'issue est de taille très petite et peut être rapidement résolue.

Mise en place du Backlog

Dans un premier temps, nous avons segmenté les différentes problématiques métier en User Stories, permettant ainsi de définir le travail à accomplir du point de vue de l'utilisateur.

Nous avons utilisé des tickets appelés "Tutor" qui comprenaient une description détaillée de l'User Story correspondante. Cela permettait aux collaborateurs de consulter ces tickets afin de maintenir une vision globale de l'application.

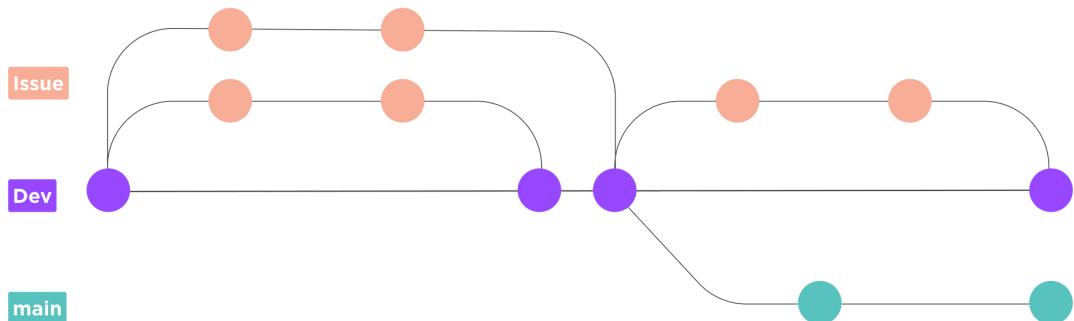


Détail du fonctionnement des jalons

DOSSIER PROFESSIONNEL (DP)

Gitflow en Trunk-based Development

Afin de maximiser le temps de développement, nous avons adopté un workflow basé sur les principes du "Trunk-Based Development" (TBD). Ce workflow simplifie l'accès aux branches en utilisant une branche commune sur laquelle nous créons des branches spécifiques pour chaque issue.



2. Nous avons défini l'environnement de développement

Nous avons opté pour un environnement de développement basé sur **Windows**, en utilisant **Visual Studio Code** comme **IDE** principal. Pour gérer notre base de données **SQL**, nous avons choisi d'utiliser **WAMP**, un ensemble de logiciels comprenant **Apache**, **MySQL** et **PHP**. Pour interagir avec la base de données, nous avons utilisé **phpMyAdmin**, une interface conviviale qui nous a permis d'exécuter des requêtes et de gérer les tables.

Une autre approche à consisté de **conteneuriser notre environnement de travail avec Docker** afin de s'assurer de sa cohérence.

```
FROM node:16

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied
# where available (npm@5+)
COPY package*.json .

RUN npm install
# If you are building your code for production
# RUN npm ci --omit=dev

# Bundle app source
COPY .

EXPOSE 3000
CMD [ "node", "server.js" ]
```

DOSSIER PROFESSIONNEL (DP)

```
Plain Text ▾
version: "3"
services:
  db:
    image: mysql:latest
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_PASSWORD}
      MYSQL_USER: "${DB_USER}"
      MYSQL_PASSWORD: ${DB_PASSWORD}
    ports:
      - "3306:3306"

  app:
    build: .
    environment:
      DB_HOST: db
      DB_NAME: ${DB_NAME}
      DB_USER: ${DB_USER}
      DB_PASSWORD: ${DB_PASSWORD}
    ports:
      - "${APP_PORT}:3000"
    depends_on:
      - db
```

 Copier Légende ...

DOSSIER PROFESSIONNEL (DP)

3. Nous avons choisi des outils collaboratifs nous permettant un meilleur organisation

❖ **Git** | comme Système de contrôle de version distribué

- Intégration rapide du projet en locale & Intégrité des données
- Mise à jour simplifié du répertoire commun
- Facilité de travail sur des fonctionnalités ou des correctifs de bugs isolés
- Visibilité rapide de l'arborescence de travail

⌚ **GitHub** → Project | comme interface de gestion du Projet

- Établissement de délais et de jalons
- Partage du code sur répertoire commun
- Peer Review lors des merge
- intrication des issues vs branches
- interface de projet sous forme de Kaban
- détail des issues en MarkDown

❖  **Google Drive** | espace de stockage collaboratif

- stockage des assets
- transfert d'informations annexes

📅 **Google Agenda** | Organisation de notre calendrier

- Prise de rendez-vous
- Daily Meeting programmés

⌚  **lucidChart** | conception de Diagramme UML & Meurise

- Réalisation de Diagramme d'activité lors de procédé complexes
- Réalisation du MCD/MLD

▣ **Notion** | comme application collaborative de prise de note

- facilitation de la hiérarchisation de l'information
- prise de note collaborative
- documentation liés au technologies associées

❖  **Figma** | interface collaborative de maquette

- Mise en place du Design Système
- Création du Wireframe
- Création de la maquette

DOSSIER PROFESSIONNEL (DP)

4. Nous avons favorisé **la communication écrite, en français** afin de rédiger les specs et les user stories de façon adaptée

The screenshot shows a Jira issue page for a user story titled "US - (CHAT) Option de discussion #18". The status is "Closed" and it was opened by "johan-bouguermouh" on Jan 8. The user story details the requirement for a user to be able to access channel options. The acceptance criteria list the steps for a user to manage their participation in a channel. A branch named "Creation infoChannel Screen #86" is associated with the user story, containing the same acceptance criteria.

Assignees: johan-bouguermouh
Labels: tutor
Milestone: MVP - Front-End
Status: Done
Linked pull requests: No linked pull requests
Repository: app-mobile-chat
Priority: High
Size: X-Large

Actions: Open in new tab, Copy link, Copy link in project, Archive, Delete from project

Issue affectées : Creation infoChannel Screen #86

Acceptance criteria:

- L'utilisateur est authentifié et participant d'un channel.
- L'utilisateur peut accéder aux options du channel.
- L'utilisateur peut quitter la discussion et ne plus recevoir de notifications.
- Le channel ne figure plus dans la liste des channels de l'utilisateur lorsqu'il quitte la discussion.
- Si l'utilisateur est à l'origine de la création du channel, il peut modifier son nom.
- Si l'utilisateur est à l'origine de la création du channel, il peut inviter de nouveaux amis.
- Si l'utilisateur est à l'origine de la création du channel, il peut supprimer définitivement la discussion.

ScreenShot d'une user stories et des branche associé

Concevoir une application

1. Nous avons **défini les cas d'utilisation afin de couvrir l'ensemble des exigences utilisateur** exprimées dans le **cahier des charges**

En nous basant sur les enseignements de Mike Cohn et sa méthodologie présentée dans "[User Stories Applied: For Agile Software Development](#)", nous avons adopté une approche de fragmentation des éléments du projet en catégories "Must-have",

DOSSIER PROFESSIONNEL (DP)

"Should-have" et "Nice-to-have". Cela nous permet de prioriser les fonctionnalités essentielles, importantes et facultatives, facilitant ainsi la prise de décisions et la planification du développement.

Must-have (Doit avoir) :

- 1 - Authentification : Permettre aux utilisateurs de créer un compte et de se connecter à l'application.
- 2 - Profil utilisateur : Permettre aux utilisateurs de créer et de gérer leur profil, y compris la modification des informations personnelles.
- 3 - Messagerie instantanée : Mettre en place un système de chat en temps réel permettant aux utilisateurs d'envoyer et de recevoir des messages.
- 4 - Annuaire des utilisateurs : Fournir une fonctionnalité de recherche ou d'affichage des utilisateurs enregistrés dans l'application.

Should-have (Devrait avoir) :

- 5 - Canal public : Permettre aux utilisateurs d'accéder à un canal public commun pour envoyer des messages visibles par tous.
- 6 - Gestion des utilisateurs : Implémenter un panneau d'administration permettant de gérer les utilisateurs, y compris la création, la modification et la suppression de comptes.
- 7 - Donner la possibilité à l'utilisateur de quitter le channel
- 8 - Gestion des droits d'accès : Permettre à l'utilisateur créateur d'un canal de discussion de définir les droits d'accès et les autorisations pour les autres utilisateurs.

Nice-to-have (Souhaitable) :

- 9 - Gestion des messages : Permettre à l'administrateur ou au modérateur de gérer les messages, y compris la suppression des messages inappropriés ou indésirables.

2. Nous avons par la suite identifier les besoins de sécurité de l'application selon les

DOSSIER PROFESSIONNEL (DP)

vulnérabilités fréquentes :

- Injection SQL
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Mauvaise gestion des sessions
- Exposition de données sensibles

Solutions envisagées :

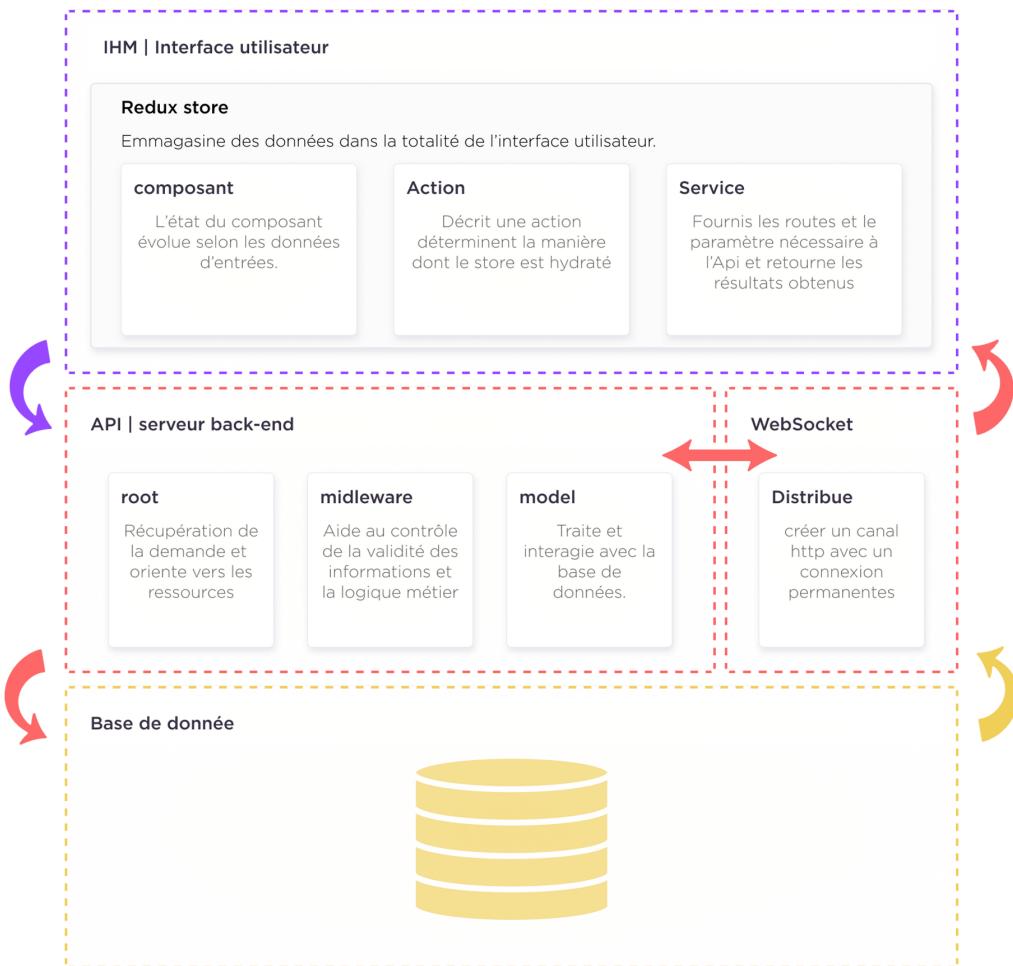
- Utilisation de requêtes préparées et d'échappement pour éviter l'injection SQL avec Sequelize
- Validation et échappement des données d'entrée pour prévenir les attaques XSS avec Helmet
- Utilisation de jetons CSRF pour protéger contre les attaques CSRF
- Utilisation de mécanismes de session sécurisés et de cookies sécurisés avec les options HttpOnly et Secure avec le JWT
- Cryptage des données sensibles en transit et au repos, et mise en place de contrôles d'accès stricts avec les hachage des mot de passes utilisateurs

3. Nous nous sommes assuré que l'architecture de notre application soit conforme au bonne pratiques

Pour ce faire, nous avons opté pour une séparation claire entre l'interface utilisateur et notre serveur. Cette approche nous permet de distribuer efficacement les ressources nécessaires à la fois pour notre application mobile et pour notre application web du panneau d'administration.

Pour garantir un développement harmonieux, nous avons choisi de construire notre serveur en suivant les principes de l'API REST. Cela nous permet d'adopter des conventions avantageuses pour la gestion des requêtes et des réponses, facilitant ainsi l'interopérabilité de notre système.

DOSSIER PROFESSIONNEL (DP)



Représentation de notre architecture de l'application mobile

Nous avons notamment organisé notre API de manière à respecter la répartition des tâches selon un architecture multicouche.



DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

- La documentation liés aux issue a était faite avec **Github action** dans le language **Markdown**
- Les diagramme ont était réalisé avec **Mermaid js**
- La documentation général a était stocké sur **Notion**
- Les différents assets à partagés ont été stocker sur **google Drive**
- L'organisation des cérémonies a été réalisé à l'aide de **Google Agenda**
- La transmission des aspect techniques de la maquettes ont été transmis à travers **Figma**
- La veille technique liées à la création de la conteneurisation avec Docker a été faite sur les sites suivant :
 - bezdoker : <https://www.bezkoder.com/docker-compose-nodejs-mysql/>
 - nodejs.org : <https://nodejs.org/en/docs/guides/nodejs-docker-webapp>

3. Avec qui avez-vous travaillé ?

Nous avons travaillé en groupe de 4 avec Boris TIKHOMIROFF, Cyril Porez et Ahmed Magassouba, tous trois étudiants au sein de l'école La Plateforme.

4. Contexte

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

Chantier, atelier, service ▶ *Réalisation d'une application mobile*

Période d'exercice ▶ Du : *04/01/2021* au : *07/02/2021*

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Concevoir et développer la persistance des données

Activité-type 3 en intégrant les recommandations de sécurité

Exemple n° 1 - Sent Application Mobile

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

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

1- Le schéma entité-association des données couvre les règles de gestion sur les données

Présentation de nos entités clefs :

La conception de base de données est cruciale pour un système d'information performant, fiable et sécurisé. Elle commence par l'identification précise des entités, objets essentiels qui stockent et organisent les données. Dans notre application chat, nous avons identifié trois entités clés : User, Channel et Conversation, chacune jouant un rôle spécifique. En structurant la base de données de manière logique et cohérente, nous garantissons un fonctionnement optimal.

User

Cette entité et la représentation de notre utilisateurs.

Chaque utilisateur est une entité distincte avec des attributs tels que le nom, l'adresse e-mail et le numéro de téléphone. Les utilisateurs sont au cœur de notre application, car ils sont la raison pour laquelle les conversations ont lieu. Ils peuvent converser avec une ou plusieurs personnes présentes dans l'application.

Channel

Cette entité représente une collectivité d'utilisateurs regroupé sous un même nom.

Un canal permet de circonscrire une communauté spécifique au sein de l'application. Les canaux facilitent la communication et l'interaction entre les utilisateurs partageant des intérêts communs.

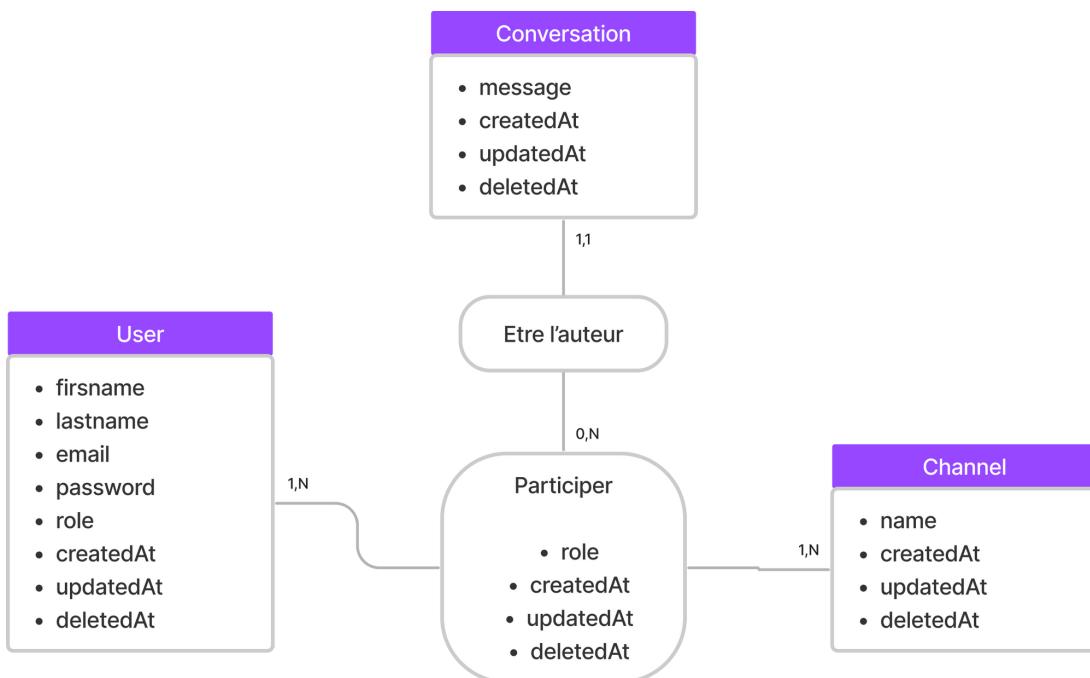
Conversation

Cette entité représente la transmission d'un message écrit.

Elle permet de conserver un message pouvant être lu par les utilisateurs d'un même canal. Les conversations sont essentielles pour permettre aux utilisateurs de communiquer entre eux

DOSSIER PROFESSIONNEL (DP)

L'analyse des entités révèle des interactions clés : les utilisateurs participent à des canaux et s'engagent dans des conversations. Cette relation entre les entités est cruciale pour une base de données cohérente et efficace. Les canaux regroupent des utilisateurs partageant des intérêts communs, tandis que les conversations facilitent les échanges. Une approche relationnelle permet d'organiser les données de manière optimale. Le Modèle Conceptuel de Base de Données (MCD) permettra de définir ces interactions et les attributs nécessaires pour caractériser les entités de manière optimale.

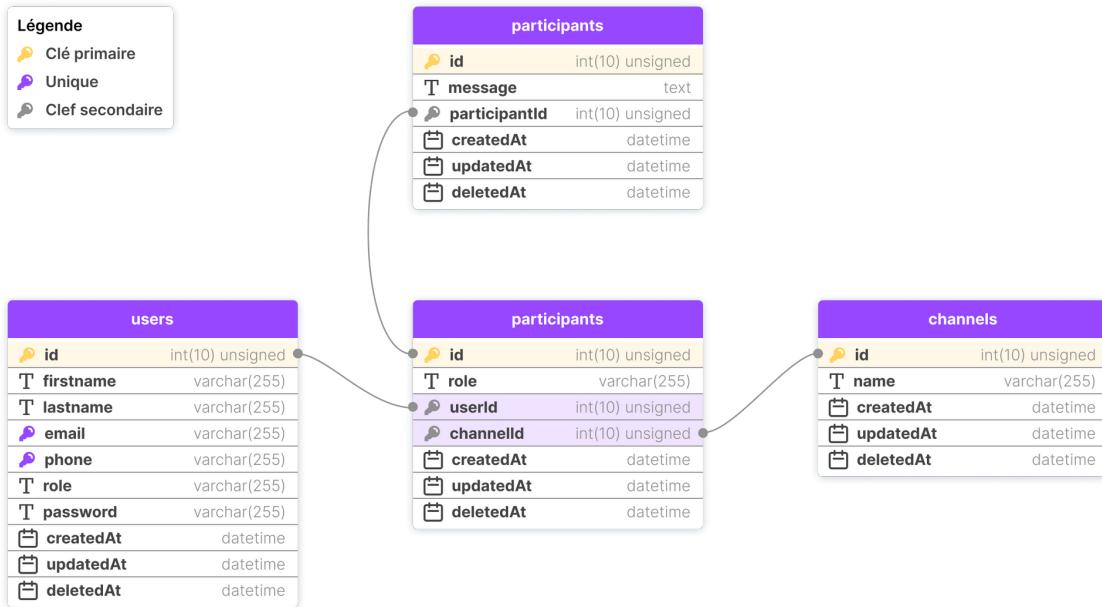


Il est important de noter que la table "Participer" agit comme une table de jonction qui enregistre les relations entre les utilisateurs, les canaux et les conversations. Elle permet de gérer la relation many-to-many entre les utilisateurs et les canaux, ainsi que la relation one-to-many entre les participants et les conversations.

MLD | Modèle logique de base de donnée

Nous définissons les tables, les clés primaire et secondaire et les relations entre les entités pour structurer notre base de données. Ce chapitre traduit les concepts du MCD en une structure concrète de tables et de relations pour optimiser notre système d'information.

DOSSIER PROFESSIONNEL (DP)



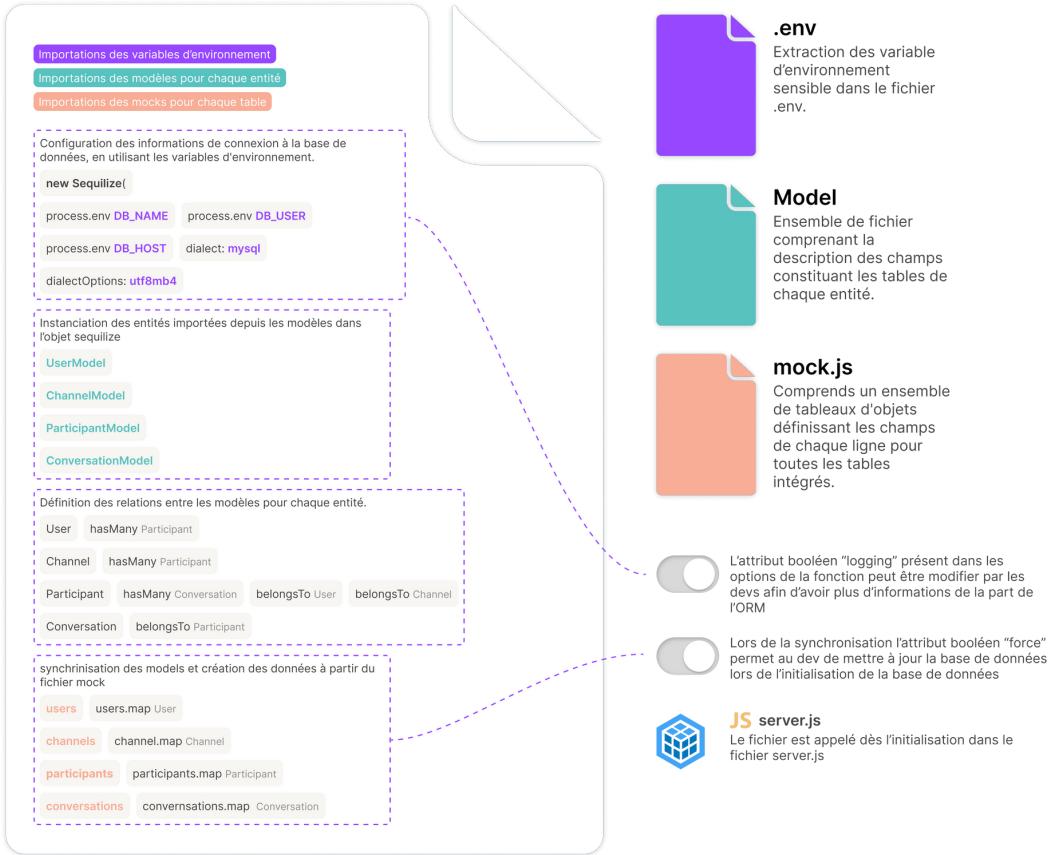
Notre choix de normaliser la relation entre les tables "Channel", "User" et "Participant" en respectant les principes de la [3NF](#) nous permet de réduire les redondances, les anomalies et d'optimiser les performances de notre système d'information.

2. Une fois la conception de base donnée terminé nous avons mis le cas en pratiques avec la création des Modèle pour chaque entité

La mise en place de l'ORM (Object-Relational Mapping) et des mocks a été cruciale pour notre backend. Nous avons utilisé Sequelize comme ORM pour interagir avec notre base de données MySQL. Une fois la connexion établie, nous avons mis en place les relations entre les modèles en utilisant les associations de Sequelize favorisant ainsi l'intégrité et la cohérence des données.

L'utilisation de Sequelize présente plusieurs avantages significatifs. Tout d'abord, il offre une couche d'abstraction qui permet une meilleure visibilité et une facilité de compréhension du code grâce à des noms de fonctions explicites. De plus, Sequelize facilite la gestion des transactions, ce qui permet d'effectuer des opérations complexes de manière simplifiée et sécurisée. En outre, Sequelize fournit des fonctionnalités supplémentaires pour renforcer la sécurité des données, notamment la validation des entrées et la prévention des attaques par injection SQL.

DOSSIER PROFESSIONNEL (DP)

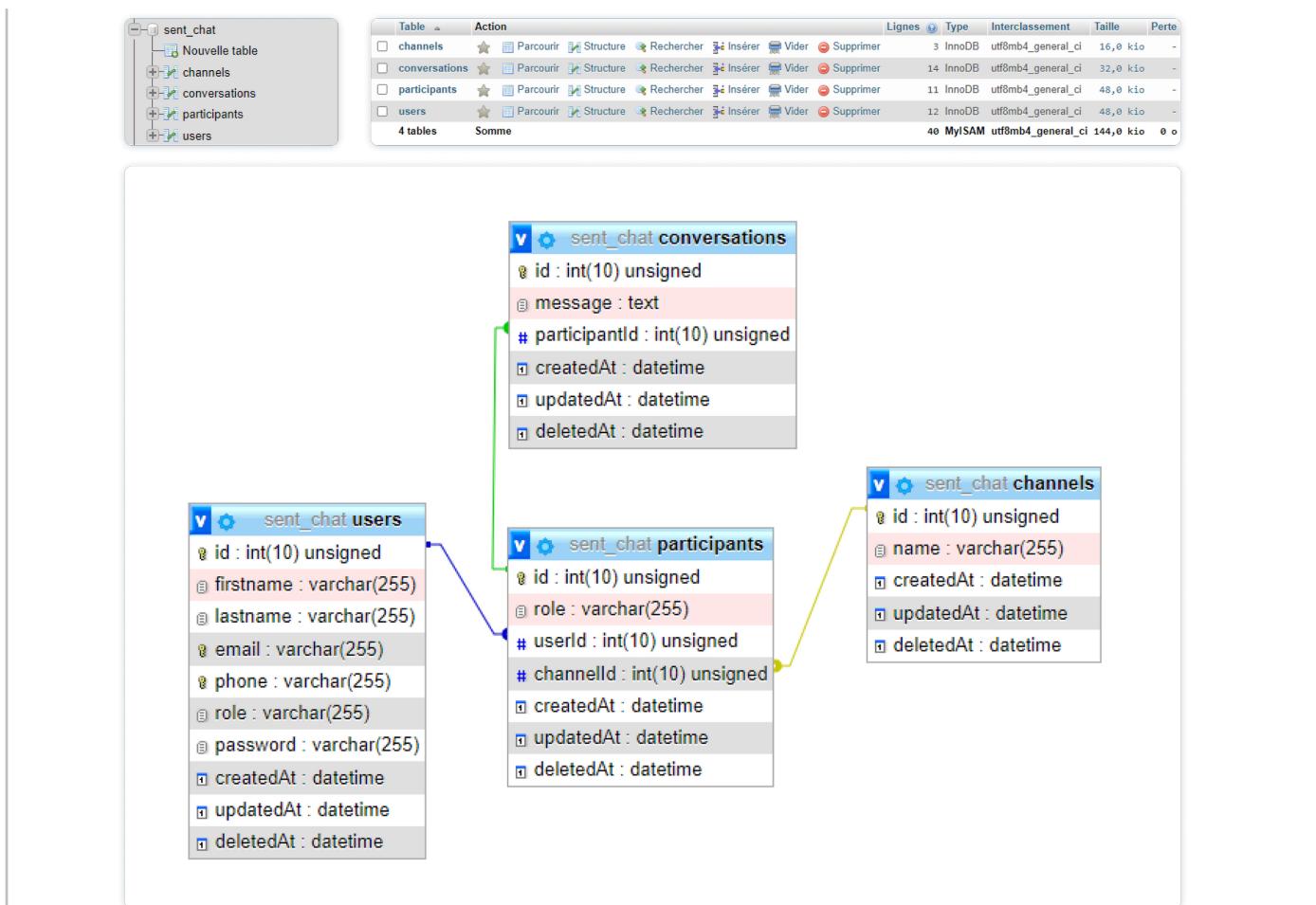


Une fois les mocks, les modèles et l'ORM mis en place, nous sommes prêts à effectuer nos tests de conformité. Après avoir structuré notre appel dans le fichier server.js et créé une base de données locale nommée "sent_chat", lorsque nous lançons la commande `npm start` dans le dossier "back", nous devrions voir apparaître dans la console :

```
Connection has been established successfully...
Synchronizing models with database...
Server is running on port 3000 : http://localhost:3000
Database is connected...
```

Après une brève vérification de notre serveur, nous constatons le succès de l'installation. Le serveur est en cours d'exécution, ce qui signifie que notre application est accessible et fonctionnelle. Nous pouvons maintenant procéder à des tests plus approfondis pour confirmer le bon fonctionnement de toutes les fonctionnalités.

DOSSIER PROFESSIONNEL (DP)



3. Une fois sequilize instancié nous avons pue nous attardé sur le développement des **composant pour la manipulation de la base de donnée**

Pour des requêtes plus complexes, Sequelize offre la possibilité d'exprimer directement notre requête en SQL. Cependant, il est important de noter que cette approche peut présenter des vulnérabilités potentielles. En effet, la CVE Détails répertorie [12 problèmes de sécurité connus liés à cette bibliothèque](#) (tous corrigés à ce jour). Par conséquent, il est essentiel de faire preuve de vigilance lors de l'utilisation de cette fonctionnalité et de prendre les mesures nécessaires pour sécuriser correctement les requêtes SQL. Notamment en s'assurant de bien échapper les caractères spéciaux afin d'évit  les injections SQL.

DOSSIER PROFESSIONNEL (DP)

```
SELECT
    co.createdAt AS sendAt,
    c.id AS channelId,
    c.name AS channelName,
    CONCAT(u.firstname, ' ', u.lastname, ' : ', co.message) AS LastMessage
FROM conversations AS co
INNER JOIN participants AS p1 ON p1.id = co.participantId
INNER JOIN users AS u ON u.id = p1.userId
INNER JOIN channels AS c ON c.id = p1.channelId
WHERE co.createdAt IN (
    SELECT MAX(co2.createdAt)
    FROM conversations AS co2
    INNER JOIN participants AS p2 ON p2.id = co2.participantId
    WHERE p2.channelId IN (
        (SELECT pA.channelId
        FROM participants AS pA
        JOIN channels AS ca ON ca.id = pA.channelId
        JOIN users AS uA ON uA.id = pA.userId
        WHERE pA.userId = ${id}
        AND pA.deletedAt IS NULL )
        GROUP BY p2.channelId)
    ORDER BY co.createdAt DESC)
```

Requête Principal

Récupère les informations des conversations, des participants, des utilisateurs et des channels associés, en filtrant les conversations en fonction des valeurs rentrées par la SubRequest A.

SubRequest A

Extrait la date de création maximale parmi les conversations liées aux participants ayant les channelId présents dans le retour de la SubRequest B suivante. Ces dates maximales groupé par channels seront ensuite utilisées dans la requête principale pour filtrer les conversations en fonction de cette valeur.

SubRequest B

Extrait les channelId des participants associés à un utilisateur spécifique qui n'ont pas été supprimés. Ces channelId seront ensuite utilisés dans la subrequest A pour filtrer les conversations en fonction de ces valeurs

Exemple de requête sql faite dans le model participant

Nous pouvons observer que l'utilisation de requêtes personnalisées nous permet de plus de flexibilité dans l'exécution des requêtes. Cependant, cette approche présente des risques potentiels en termes de sécurité. Bien que cela améliore la lisibilité, il peut y avoir des problèmes de logique. Dans cet exemple, les messages sont sélectionnés en fonction de la valeur createdAt (DATETIME), ce qui peut poser des problèmes dans des cas très rares de retour en arrière. Il serait donc préférable d'identifier les ressources par leur ID. Cependant, ce problème reste mineur

5. Nous avons notamment veillé à ce que l'intégrité et la confidentialité des données soit maintenues

Afin de garantir l'intégrité des données des utilisateurs, nous avons pris des mesures pour ne pas transmettre les informations sensibles, telles que l'adresse e-mail et le numéro de téléphone. De plus, nous nous assurons que le mot de passe de l'utilisateur ne soit jamais renvoyé en dehors du contrôleur responsable de sa vérification. Dans notre solution, tous les mots de passe stockés en base de données sont hashés et salés en utilisant la librairie bCrypt. Nous appliquons également des règles strictes de validation, à la fois côté front-end et côté back-end, en utilisant des expressions régulières pour garantir que les données respectent nos exigences de sécurité. En prévision d'une version destinée à être mise en ligne, nous inclurons un lien vers les

DOSSIER PROFESSIONNEL (DP)

conditions d'utilisation, conformément aux exigences de la loi RGPD en vigueur. De plus, nous ajouterons une fonctionnalité qui obligera les utilisateurs à donner leur consentement explicite avant de pouvoir s'inscrire. Cela permettra aux utilisateurs d'accéder facilement aux informations concernant la protection de leurs données personnelles et leurs droits en matière de confidentialité.

```
if (!passwordRegex.test(body.password)) {
  const message =
    "Le mot de passe doit contenir au moins 8 caractères,
    une majuscule, une minuscule et un chiffre";
  return res.status(400).json({ message });
} else {
  const salt = bcrypt.genSaltSync(
    parseInt(process.env.BCRYPT_SALT_ROUNDS)
);
```

De la même manière que l'exemple donnée précédemment nous nous sommes assuré de constamment **contrôler les entrées et validé dans nos composants serveur** avant de les envoyer en base de données.

2. Précisez les moyens utilisés :

- Nous avons utilisé **Sequelize** pour l'initialisation de la base de donnée
- En local nous avons utilisé **phpMyAdmin** pour une visualisation claire de la base de donnée
- Les schémas de conception de la base de donnée et du modèle logique de donnée ont été réalisé préalablement avec **Draw.io**
- Nous avons utilisé **bCrypt** pour saler et hasher nos informations sensibles
- Nous avons utilisé **Crypto** pour une meilleure approche du cryptage

3. Avec qui avez-vous travaillé ?

4. Contexte

Nom de l'entreprise, organisme ou association ▶

Cliquez ici pour taper du texte.

Chantier, atelier, service ▶

Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL (DP)

Période d'exercice ▶ Du : [Cliquez ici](#) au : [Cliquez ici](#)

5. Informations complémentaires (facultatif)

Activité-type 4 Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ▶ sent Application Mobile

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

2. Précisez les moyens utilisés :

- ▶ Nous avons utilisé **Sequelize** pour l'initialisation de la base de donnée
- ▶ En local nous avons utilisé **phpMyAdmin** pour un visualisation claire de la base de donnée
- ▶ Les schémas de conception de la base de donnée et du modèle logique de donnée ont été réalisé préalablement avec **Draw.io**
- ▶ Nous avons utilisé **bCrypt** pour saler et hasher nos informations sensible
- ▶ Nous avons utilisé **Crypto** pour une meilleur approche du cryptage

3. Avec qui avez-vous travaillé ?

4. Contexte

Nom de l'entreprise, organisme ou association ▶ [Cliquez ici pour taper du texte.](#)

Chantier, atelier, service ▶ [Cliquez ici pour taper du texte.](#)

Période d'exercice ▶ Du : [Cliquez ici](#) au : [Cliquez ici](#)

DOSSIER PROFESSIONNEL^(DP)

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné(e) Johan Bouguermouh , 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 21/07/2023

pour faire valoir ce que de droit.



Signature :

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)