



SIMPLON.CO



OPUS

Plateforme de gestion des projets

Par Adam Khomsi

YouCode Safi

Deuxième année, Classe Robert Noyce

Développement Fullstack JavaScript

2021 - 2022

Table des matières

Résumé	2
Introduction	3
Problématique	3
Solution	3
Chapitre 1 : Description du projet	4
1.1. Qu'est-ce que la gestion des projets?	4
1.2. L'importance de la gestion des projets:	4
1.3. Fonctionnalités générales:	4
Chapitre 2 : Technologies utilisées	5
2.1. Conception logicielle:	5
2.2. Design UI et UX:	5
2.3. Développement logiciel:	5
Chapitre 3 : Conception UML	7
3.1. Diagramme de classes:	7
3.2. Diagramme de cas d'utilisation:	8
3.3. Diagramme de séquence:	9
Chapitre 4 : Maquette	10
Chapitre 5 : Développement	11
5.1. Introduction:	11
5.2. Backend:	11
5.2.1. Structure et architecture:	11
5.2.2. Variables d'environnement:	12
5.2.3. Commandes:	13
5.2.4. Création des entités et relations:	18
5.3. Frontend:	18
5.4. Captures d'écrans:	19
Conclusion	21
Remerciements	21
Webographie	22

Résumé

Ce rapport est pour un projet de fin d'études pour la deuxième année chez YouCode en Développement Fullstack JavaScript. Le projet consiste à mettre en place une solution pour gérer les tâches, bugs, incidents pouvant survenir dans tout type d'établissement ou d'organisation à l'aide d'un tableau Kanban.

L'introduction et le chapitre 1 couvriront des explications sur le concept, quels sont les problèmes et une brève étude des solutions existantes, et la solution proposée avec ce projet.

Le chapitre 2 traite les technologies et les outils utilisés pour développer ce projet, de la conception logicielle au design de l'interface utilisateur et enfin au développement proprement dit de l'application.

A partir du chapitre 3, chaque étape du processus sera abordée plus en détail en commençant par les diagrammes UML, puis le design de l'interface utilisateur au chapitre 4, et enfin le processus de développement avec le chapitre 5.

Introduction

La gestion de projet est l'ensemble des activités visant à organiser le bon déroulement d'un projet et à en atteindre les objectifs. Elle consiste à appliquer les méthodes, techniques, et outils de gestion spécifiques aux différentes étapes du projet, de l'évaluation de l'opportunité jusqu'à l'achèvement du projet.

Il s'agit d'une plateforme conçue pour une gestion des tâches, incidents (bugs, problèmes) d'un projet

La plateforme offrira des espaces par équipe, gérable par un chef de projet qui aura la responsabilité tout au long du processus.

Problématique

Il existe de nombreuses solutions, mais pour certaines personnes, elles peuvent être trop compliquées ou remplies avec d'autres fonctionnalités inutiles, et elles s'appuient également sur des logiciels et des services d'autres sociétés.

Solution

Concevoir une application ou une plateforme simple d'utilisation avec seulement les fonctionnalités nécessaires et qui ne dépend pas de systèmes externes.

Chapitre 1 : Description du projet

1.1. Qu'est-ce que la gestion des projets?

De manière générale, la gestion des projets est un processus qui permet la planification des activités nécessaire au bon déroulement de ces derniers, pour garantir et atteindre un niveau de qualité souhaité, gérer et maîtriser les risques et problèmes tout au long du processus.

1.2. L'importance de la gestion des projets:

Sans organisation, la gestion des ressources, la planification et le contrôle des risques s'avèrent difficiles. Il est donc impératif de bien planifier les tâches et de les répartir, d'avoir une référence accessible à tout moment.

1.3. Fonctionnalités générales:

La plateforme offrira des tableaux pour chaque équipe gérable par un chef de projet. Les tableaux auront différents types (tâches, bugs) et seront constitués de différentes sections où les membres de l'équipe pourront contrôler leur tâches attribuées sous forme de tickets.

Chapitre 2 : Technologies utilisées

2.1. Conception logicielle:

UML (Unified Modeling Language): Langage de modélisation à usage général qui visualise la conception d'un système à travers des diagrammes.

2.2. Design UI et UX:

Adobe XD: Outil de prototypage utilisé pour concevoir l'UI et l'UX de l'application (Web et Mobile).

2.3. Développement logiciel:

TypeScript: est un langage de programmation développé et maintenu par Microsoft. Il s'agit d'un sur-ensemble syntaxique strict de JavaScript et ajoute un typage statique facultatif au langage. Il est conçu pour le développement de grandes applications et se transpose en JavaScript. Il permet de traiter la majorité des erreurs avant qu'elles n'apparaissent et vérifie l'intégralité du code à la compilation au lieu de l'exécution, ce qui permet d'éviter de nombreux bugs avant leurs apparitions.

Visual Studio Code: Environnement de développement intégré (IDE) à l'aide de nombreuses extensions telles que:

- **Draw.io:** Intégration de Draw.io dans VS Code pour dessiner des diagrammes UML.
- **Thunder Client:** Extension client HTTP (alternative à Postman), pour tester les APIs.

Backend (Node.js) :

- **Express:** est un framework d'application Web backend pour Node.js, publié en tant que logiciel gratuit et open-source sous la licence MIT. Il est conçu pour créer des applications Web et des APIs.
- **MongoDB:** MongoDB est un programme de base de données multiplateforme orienté document disponible en open source. Classé comme programme de base de données NoSQL, MongoDB utilise des documents de type JSON avec des schémas facultatifs. BSON est le format binaire JSON utilisé par MongoDB pour le stockage et le transfert de données

- **Mongoose:** est une bibliothèque de programmation orientée objet JavaScript qui crée une connexion entre MongoDB et le framework d'application Web Express. fournit une solution simple basée sur un schéma pour modéliser vos données d'application. Il comprend le casting de type intégré, la validation, la création de requêtes, les hooks et autres.
- **JWT:** JSON Web Token est une norme Internet proposée pour créer des données avec une signature facultative et/ou un cryptage optionnel dont la charge utile contient JSON qui affirme un certain nombre de revendications. Les jetons sont signés à l'aide d'un secret privé ou d'une clé publique/privée.

Frontend:

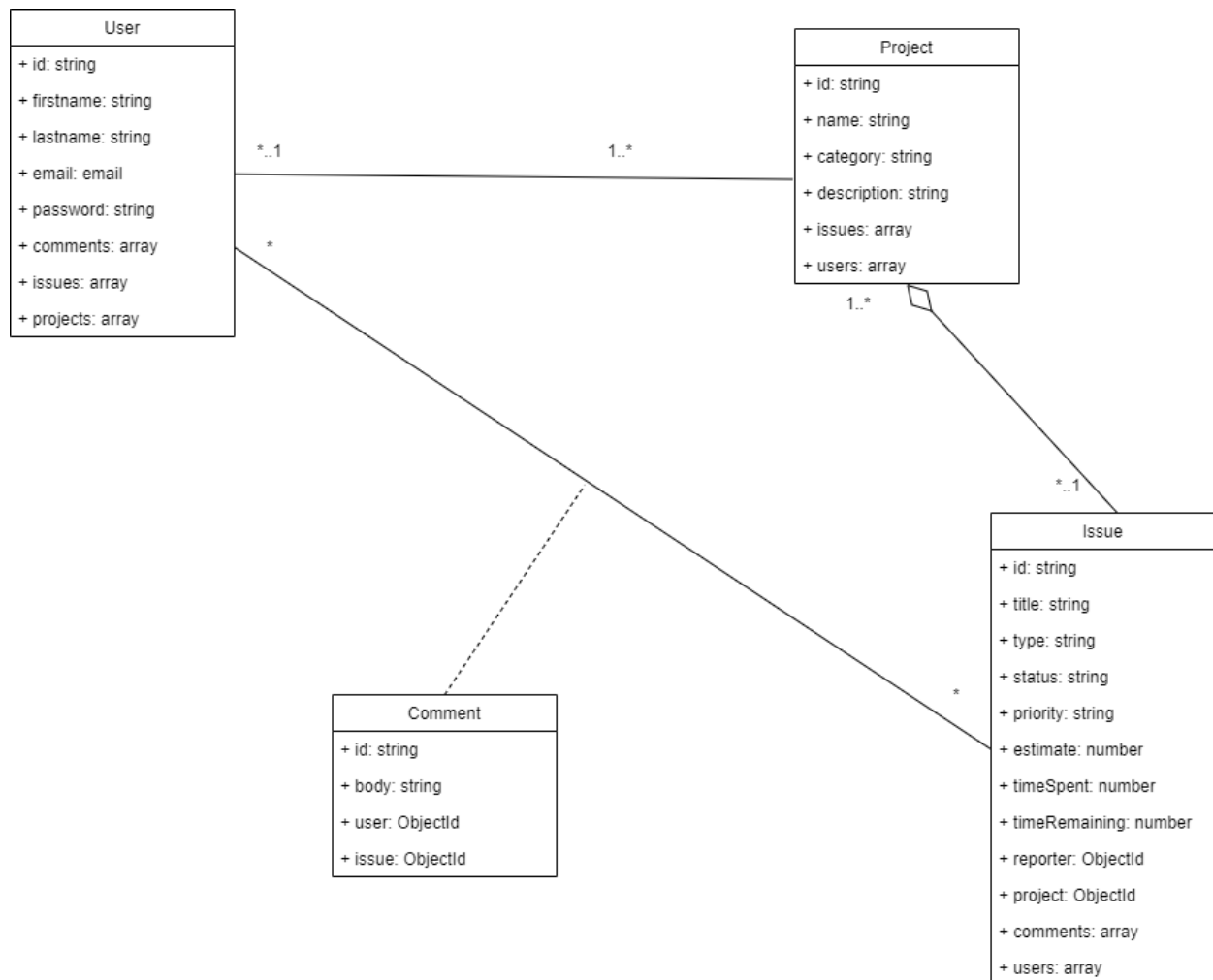
- **React:** est une bibliothèque JavaScript open-source permettant de créer des interfaces utilisateur basées sur des composants d'interface utilisateur. Il est maintenu par Meta (auparavant Facebook) et une communauté de développeurs individuels et d'entreprises.
- **Vite:** est un “build tool” qui vise à fournir une expérience de développement plus rapide et plus légère (comparé à Webpack) pour les projets Web modernes.
- **TailwindCSS:** Un framework CSS ‘utility first’ rempli de classes qui peuvent être composées pour créer n'importe quelle conception, directement dans votre balisage. Il est facile à personnaliser, s'adapte à n'importe quelle conception et la taille des feuilles de styles est optimale car il peut filtrer toutes les classes non utilisées.
- **Redux Toolkit** Redux est une bibliothèque JavaScript open-source permettant de gérer et de centraliser l'état des applications.
- **RTK Query:** Inclus dans **Redux Toolkit**, permet la synchronisation des données performante et puissante pour React. Récupération, mise en cache et mise à jour des données dans vos applications React et React Native sans toucher à aucun "état global".

Chapitre 3 : Conception UML

Le langage de modélisation unifié n'est pas un langage de programmation. C'est un langage visuel courant dans le monde complexe du développement de logiciels. Il se compose de différents types de diagrammes qui décrivent les limites, la structure et le comportement du système et des objets qu'il contient.

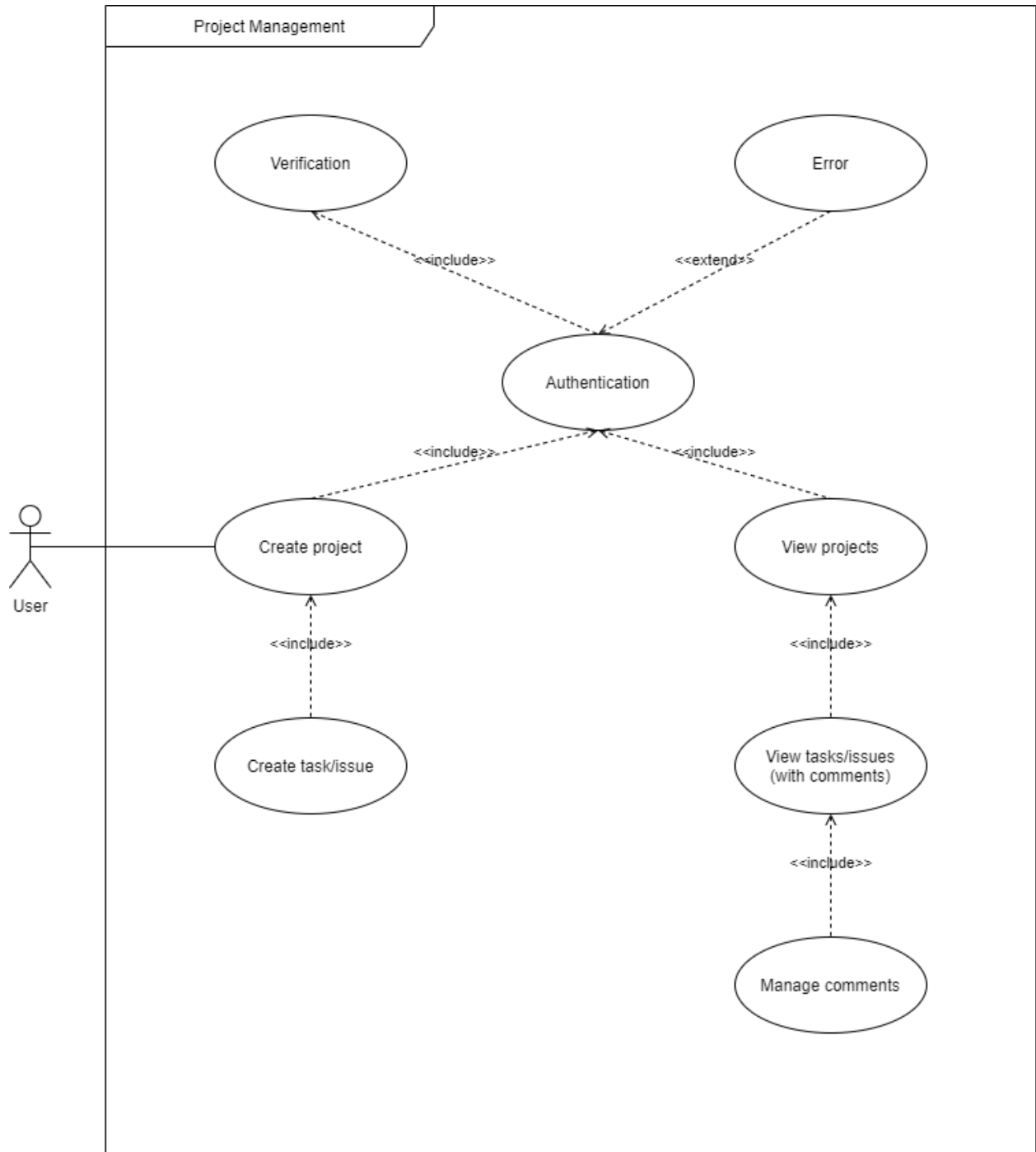
3.1. Diagramme de classes:

Les diagrammes de classes sont le fondement principal de toute solution orientée objet, qui décrit la structure du système en termes d'objets, d'attributs, d'associations et d'opérations.



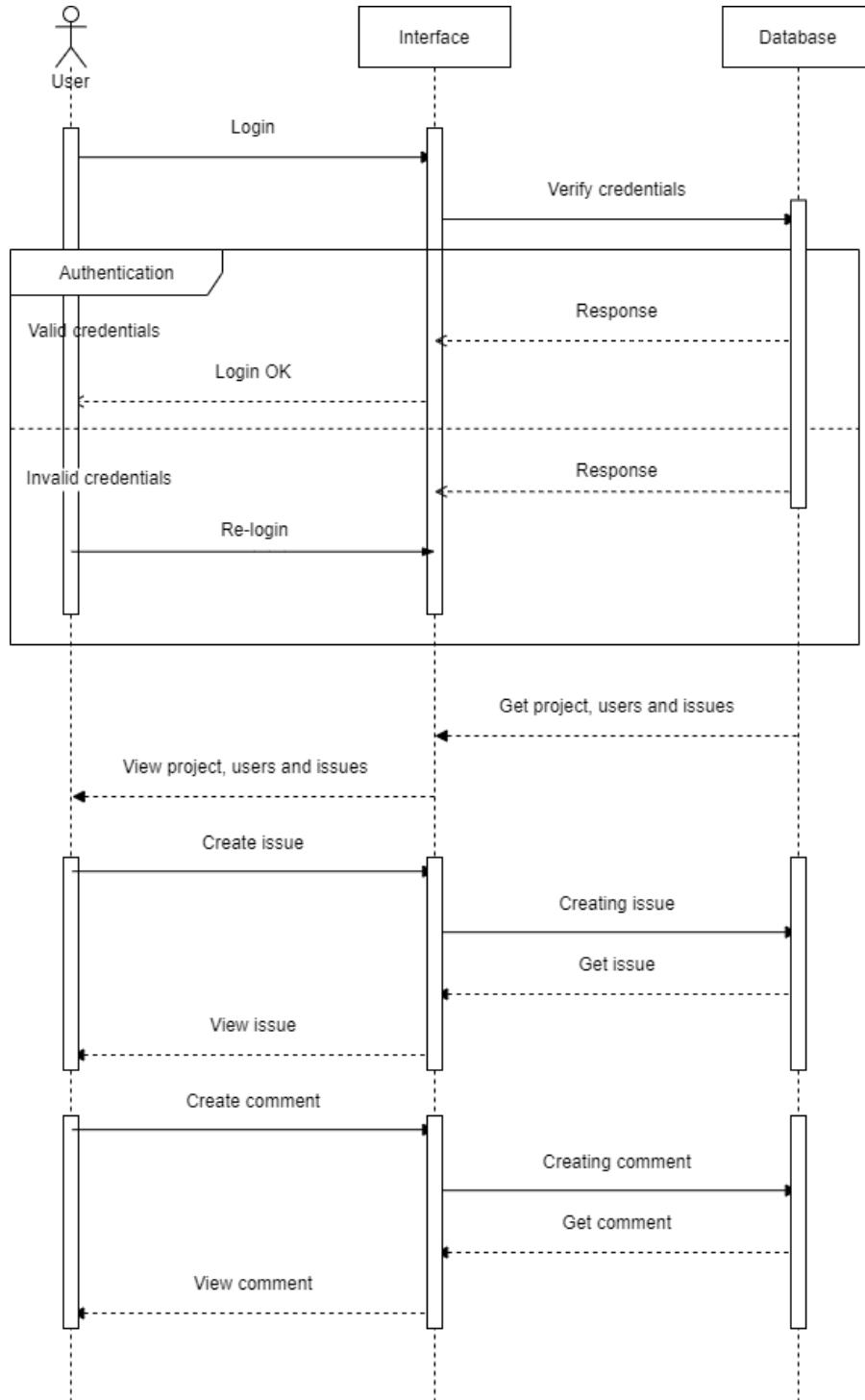
3.2. Diagramme de cas d'utilisation:

Les diagrammes de cas d'utilisation représentent la fonctionnalité d'un système, qui décrit la fonctionnalité du système du point de vue d'un acteur.



3.3. Diagramme de séquence:

Les diagrammes de séquence montrent comment les objets interagissent les uns avec les autres et l'ordre d'occurrence. Ils représentent des interactions pour un scénario particulier.



Chapitre 4 : Maquette

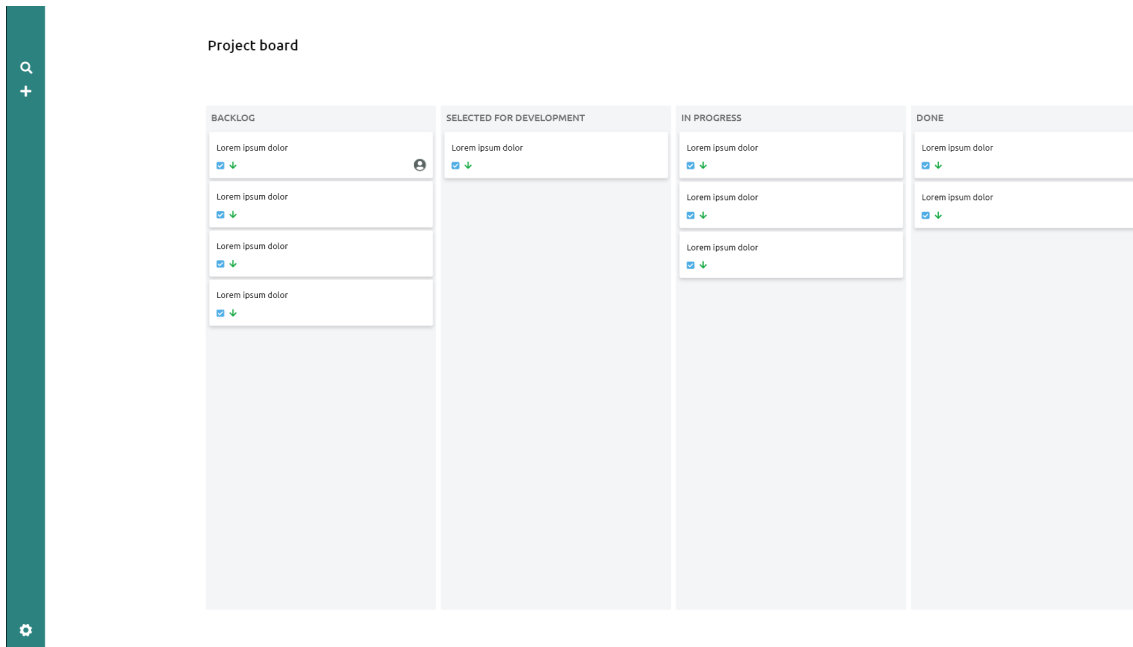


Figure 1 - Tableau Kanban

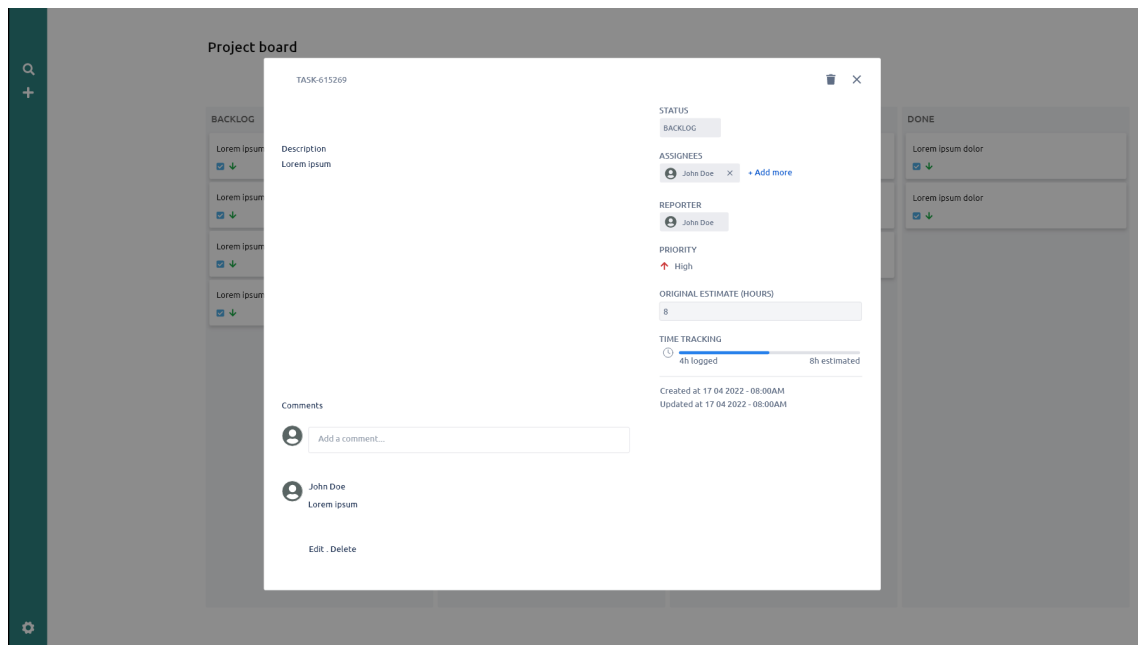


Figure 2 - Détails d'une tâche

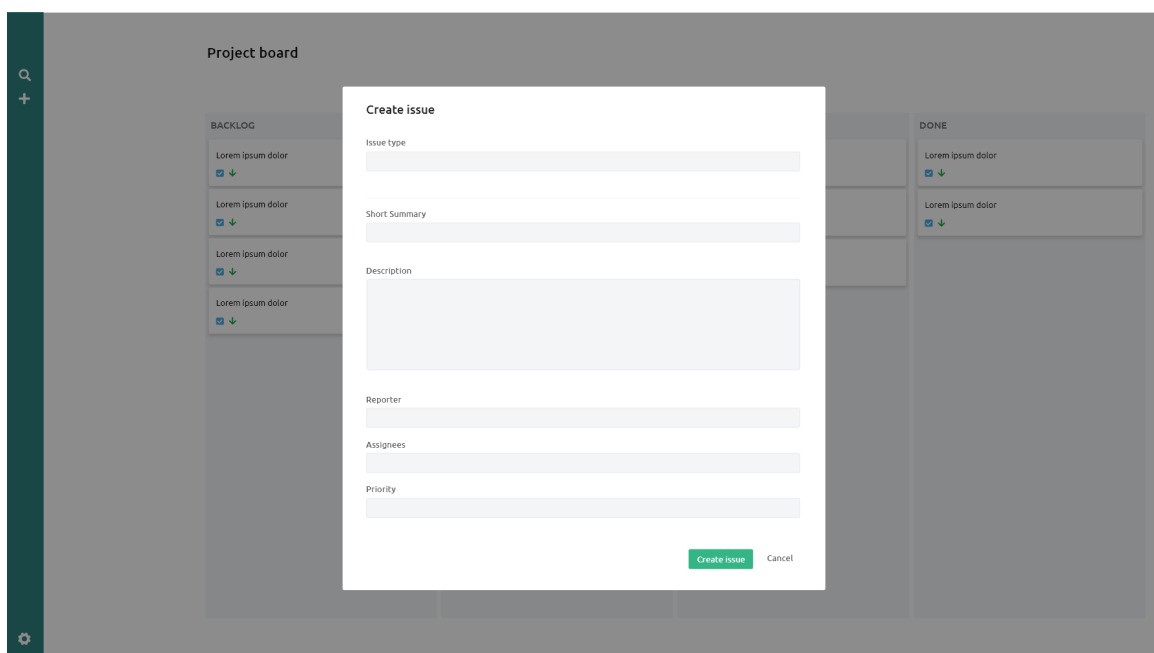


Figure 3 - Création d'une tâche

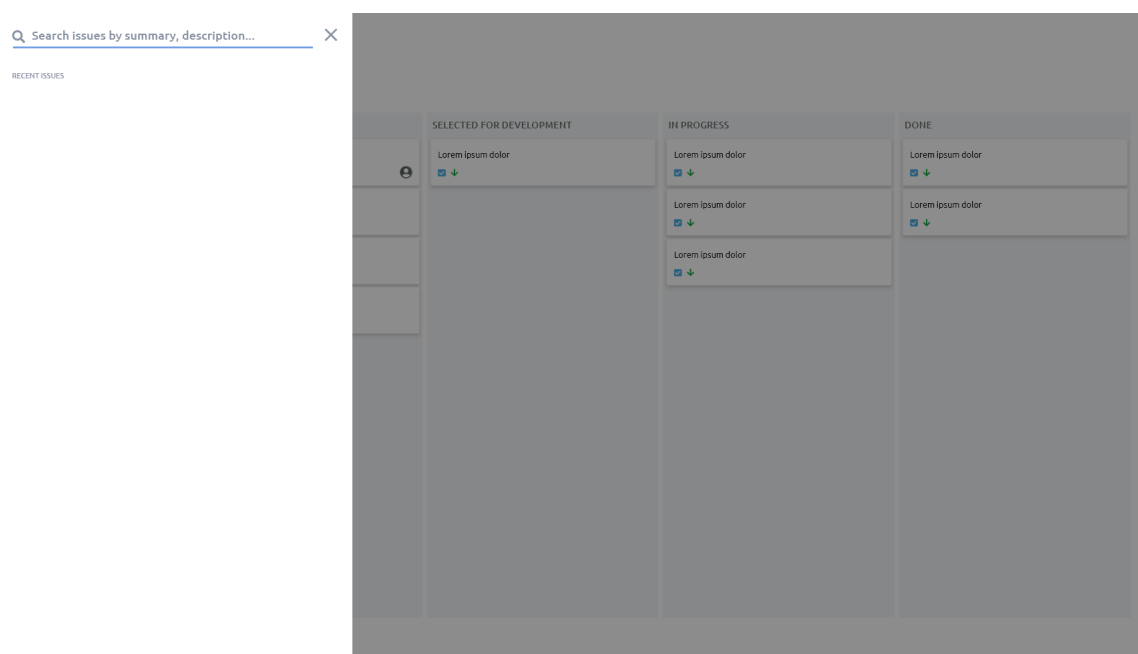


Figure 4 - Recherche des tâches

Chapitre 5 : Développement

5.1. Introduction:

Tout en travaillant sur les briefs assignés au cours de ma formation, j'ai réussi à développer ma propre structure ([REST](#) ou [GraphQL](#)) en backend, inspirée de l'architecture MVC. Cela me permet de commencer n'importe quel projet sans perdre de temps à créer la structure à zéro à chaque fois et configurer l'authentification et le CRUD basique. Cela m'a permis de passer rapidement au développement et de commencer à créer l'application et ses fonctionnalités.

5.2. Backend:

5.2.1. Structure et architecture:

La structure du projet est basée sur l'architecture MVC et suit ses principes de base. Au lieu d'avoir la logique des entités réparties dans des dossiers spécifiques (dossier des modèles contenant tous les modèles, dossier des contrôleurs contenant tous les contrôleurs, etc.).



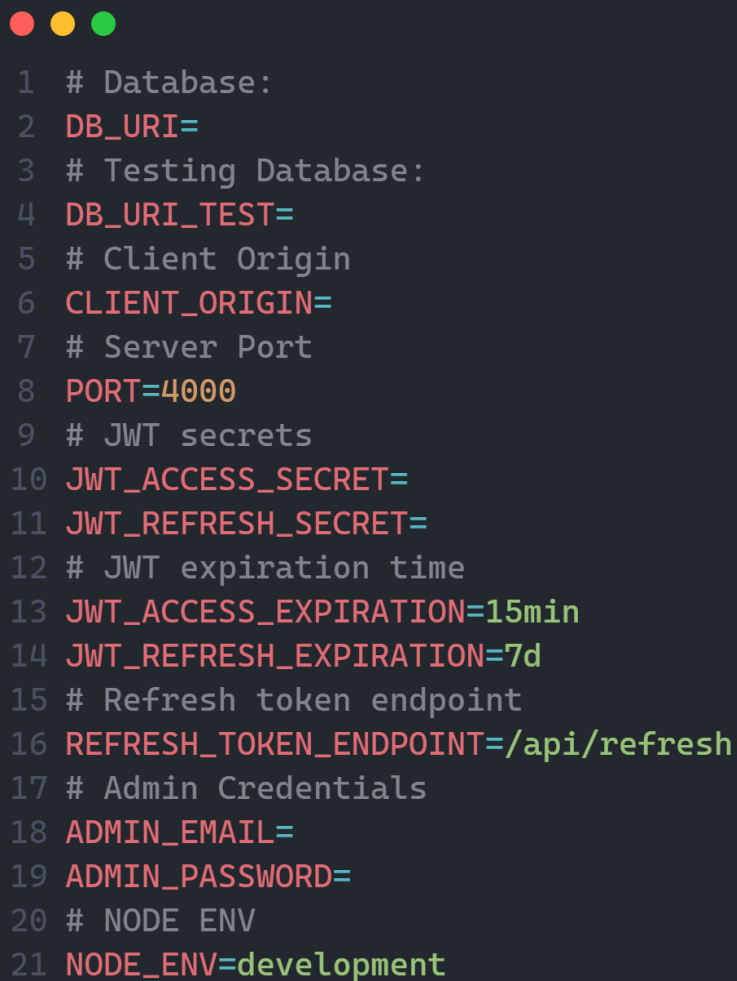
Figure 1 - Structure du projet: backend/api

Avec cette structure (figure à gauche), il est plus facile de maintenir et d'évoluer avec plusieurs entités (vous aurez rarement à basculer entre les dossiers afin de gérer une entité).

Chaque entité a son propre dossier (figure à droite, exemple de l'entité **User**) contenant toute sa logique de base isolée des autres entités.

5.2.2. Variables d'environnement:

On commence par configurer nos variables d'environnement dans le fichier .env, où nous avons nos informations de base de données, et quelques constantes définies pour l'authentification de l'utilisateur.



```
1 # Database:
2 DB_URI=
3 # Testing Database:
4 DB_URI_TEST=
5 # Client Origin
6 CLIENT_ORIGIN=
7 # Server Port
8 PORT=4000
9 # JWT secrets
10 JWT_ACCESS_SECRET=
11 JWT_REFRESH_SECRET=
12 # JWT expiration time
13 JWT_ACCESS_EXPIRATION=15min
14 JWT_REFRESH_EXPIRATION=7d
15 # Refresh token endpoint
16 REFRESH_TOKEN_ENDPOINT=/api/refresh
17 # Admin Credentials
18 ADMIN_EMAIL=
19 ADMIN_PASSWORD=
20 # NODE ENV
21 NODE_ENV=development
```

Figure 2 - Variables d'environnements

5.2.3. Commandes:

Il y a plusieurs commandes qui permettent par exemple de lancer le serveur de développement ou production (après avoir compilé TypeScript en JavaScript), de formater tous les fichiers, lancer les tests etc.. La commande (dépendant du package manager utilisé) **npm run/yarn/pnpm entity** permet de générer une entité avec des propriétés prédéfinies à partir de type choisis et du nom de l'entité:

```
? What entity template would you like to generate? (Use arrow keys)
> default
  user

? What entity template would you like to generate? default
? Entity name: post
```

Figure 3 - Création d'une entité "Post" comme exemple, avec la commande: "yarn entity"

En choisissant le type par défaut, et après avoir entré le nom de l'entité, elle sera générée automatiquement à partir d'une template et mise dans le dossier **src/entities**.

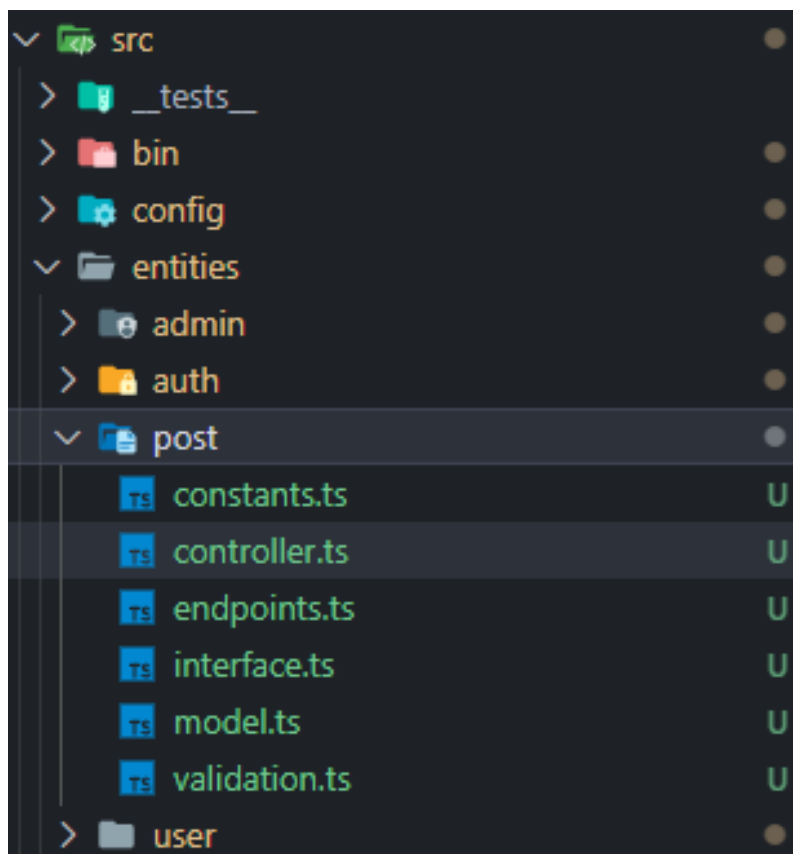


Figure 4 - Exemple de la structure d'une entité

```

1 export enum SuccessMessages {
2   POST_CREATED = 'Post created successfully.',
3   POST_UPDATED = 'Post updated successfully.',
4   POST_DELETED = 'Post deleted successfully.',
5 }
6
7 export enum ErrorMessages {
8   POSTS_NOT_FOUND = 'No posts found.',
9   POST_NOT_FOUND = 'Post was not found.',
10 }

```

Figure 5 - Exemple des constantes d'une entité

```

1 import type { Request, Response, NextFunction } from 'express';
2 import * as controller from '@services/crud.service';
3
4 import { catchErrors } from '@helpers/catchErrors';
5 import { PostModel } from '../model';
6 import { createPostSchema, updatePostSchema } from './validation';
7 import { SuccessMessages, ErrorMessages } from './constants';
8
9 export const create = catchErrors(async (req: Request, res: Response, next: NextFunction) => {
10   controller.create(req, res, next, createPostSchema, PostModel, SuccessMessages.POST_CREATED);
11 });
12
13 export const getAll = catchErrors(async (_req: Request, res: Response, next: NextFunction) => {
14   controller.getAll(_req, res, next, PostModel, ErrorMessages.POSTS_NOT_FOUND);
15 });
16
17 export const getById = catchErrors(async (req: Request, res: Response, next: NextFunction) => {
18   controller.getByField(req, res, next, PostModel, ErrorMessages.POST_NOT_FOUND);
19 });
20
21 export const update = catchErrors(async (req: Request, res: Response, next: NextFunction) => {
22   controller.update(
23     req,
24     res,
25     next,
26     updatePostSchema,
27     PostModel,
28     SuccessMessages.POST_UPDATED,
29     ErrorMessages.POST_NOT_FOUND,
30   );
31 });
32
33 export const remove = catchErrors(async (req: Request, res: Response, next: NextFunction) => {
34   controller.remove(req, res, next, PostModel, SuccessMessages.POST_DELETED, ErrorMessages.POST_NOT_FOUND);
35 });
36

```

Figure 6 - Exemple de contrôleur d'une entité avec les méthodes réutilisable du contrôleur principal

Le contrôleur contient les méthodes du CRUD basique, en utilisant les méthode d'un contrôleur de base qui contient toute la logique, les valeurs sont passées automatiquement à partir du nom de l'entité.



```
1 import { Router } from 'express';
2 import { is } from '@middlewares/isAuth';
3 import * as post from './controller';
4
5 const endpoints = Router();
6
7 endpoints.post('/', is.Auth, post.create);
8 endpoints.get('/', is.Auth, post.getAll);
9 endpoints.get('/:id', is.Auth, post.getById);
10 endpoints.patch('/:id', is.Auth, post.update);
11 endpoints.delete('/:id', is.Auth, post.remove);
12
13 export default endpoints;
```

Figure 7 - Exemple des routes prédéfinies d'une entité



```
1 import { Router } from 'express';
2
3 import authEndpoints from '@entities/auth/endpoints';
4 import adminEndpoints from '@entities/admin/endpoints';
5 import userEndpoints from '@entities/user/endpoints';
6 // Imported Post endpoints
7 import postEndpoints from '@entities/post/endpoints';
8
9 const router = Router();
10
11 router.use('/', authEndpoints);
12 router.use('/admins', adminEndpoints);
13 router.use('/users', userEndpoints);
14 // Posts route definition
15 router.use('/posts', postEndpoints);
16
17 export default router;
```

Figure 8 - Importation des routes de la nouvelle entité créer

Il reste juste à remplir l'interface, le modèle et la validation avec les champs voulu et leurs propriétés:

```
1 export interface PostEntity {}
```

```
1 import { Schema, model } from 'mongoose';  
2  
3 import { PostEntity } from './interface';  
4  
5 const PostSchema = new Schema<PostEntity>({}, { timestamps: true });  
6  
7 export const PostModel = model<PostEntity>('Post', PostSchema);  
8
```

```
1 import Joi from 'joi';  
2  
3 export const createPostSchema = Joi.object({});  
4  
5 export const updatePostSchema = Joi.object({});  
6
```

Figure 8 - Exemple de l'interface, modèle et validation à remplir avec les champs requis

5.2.4. Création des entités et relations:

Les entités **Admin** et **User** existent déjà, donc il suffit de créer les entités restantes (**Project**, **Issue**, **Comment**), et de remplir les champs requis et configurer les relations par rapport notre diagramme de classe à l'aide des middlewares et hooks de Mongoose.

Après avoir fini la configuration des relations, ayant les méthodes CRUD de base déjà prédéfinies. Il suffit d'ajouter les méthodes nécessaires et leur routes (affectation des tâches etc..).

5.3. Frontend:

J'ai choisi de travailler avec React seulement parce que la plateforme est conçue pour être utilisée en interne donc pas besoin de SEO.

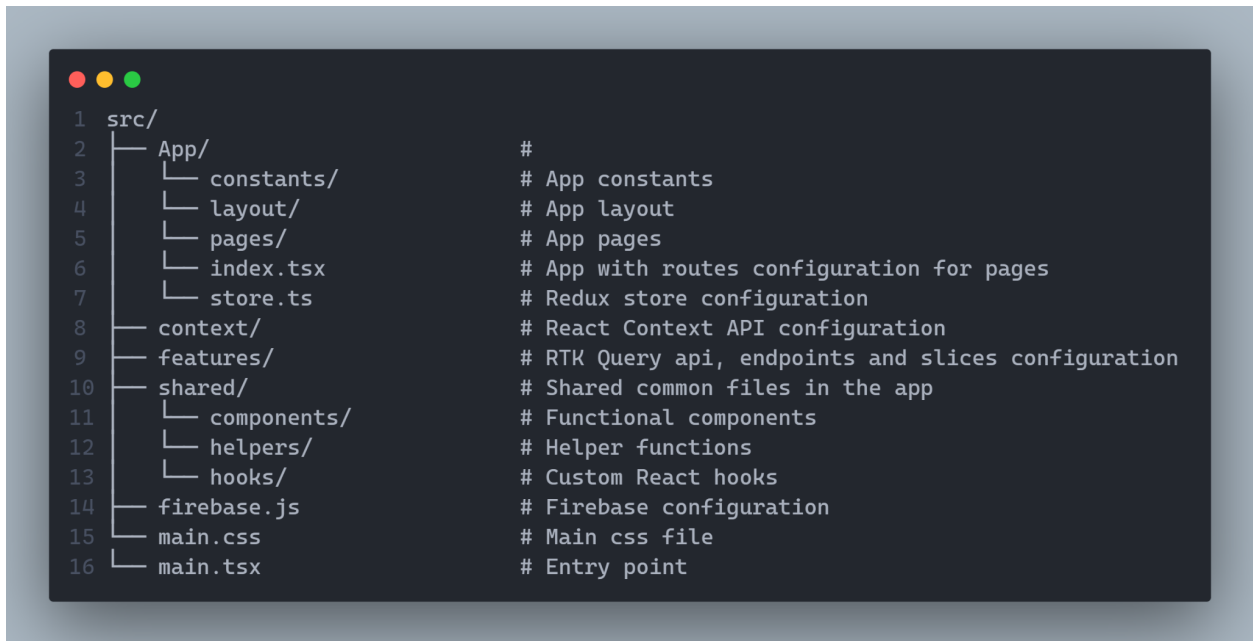
Utilisant React Context API pour traiter les données du projet en local pour une meilleure expérience utilisateur.

Redux Toolkit est utilisé pour gérer l'authentification dans l'application, car on reçoit un token d'authentification avec une période courte, qui est destiné à être enregistré dans la mémoire au lieu du LocalStorage pour un maximum de sécurité. Le problème avec cette méthode est que suite à un rafraichissement de la page, on perd le token enregistré. C'est là que vient le rôle du "refresh token" qui est envoyé en tant que cookie HTTP only. A chaque rafraichissement de page ou quand le token d'authentification s'expire, une requête est envoyée automatiquement pour obtenir un nouveau token et continuer l'utilisation de l'application sans aucune coupure.

Avec RTK Query, il est possible de configurer les routes de l'API avec des hooks qui gèrent l'état des données du serveur et nous permettent de manipuler les données, l'interface et l'expérience utilisateur.

Le tableau Kanban a été géré avec la librairie "react-beautiful-dnd" d'Atlassian, développeurs des solutions Jira et Trello.

La structure de la partie client a été conçue avec une vue maintenable.



5.4. Captures d'écrans:

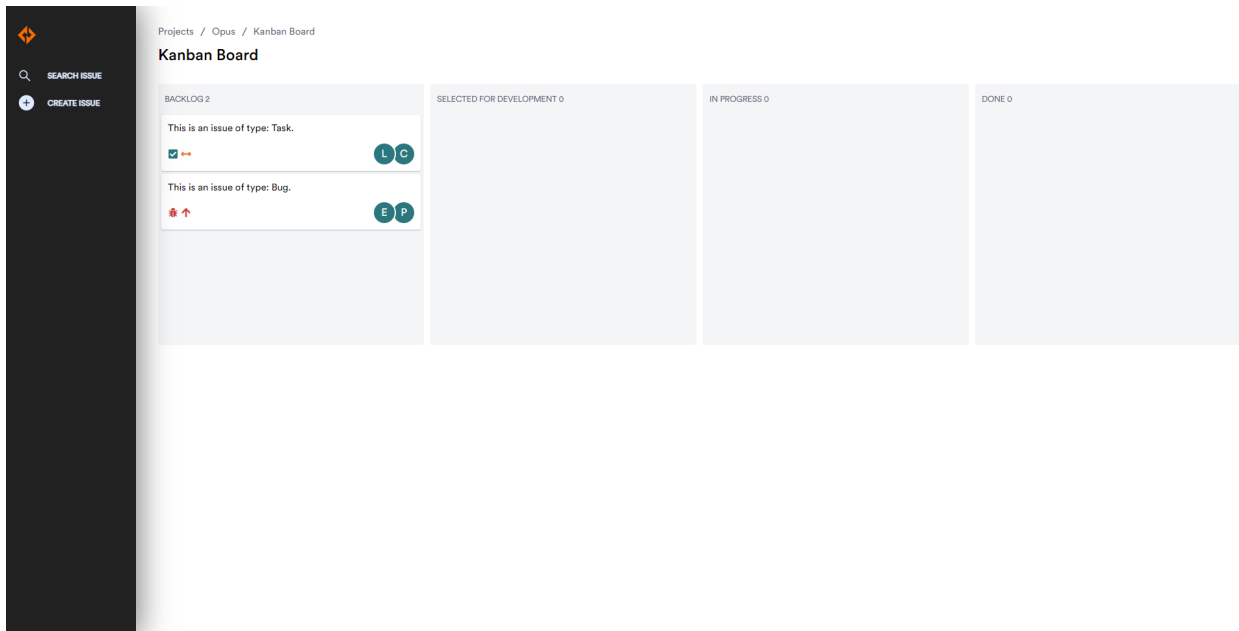


Figure 1 - Tableau Kanban

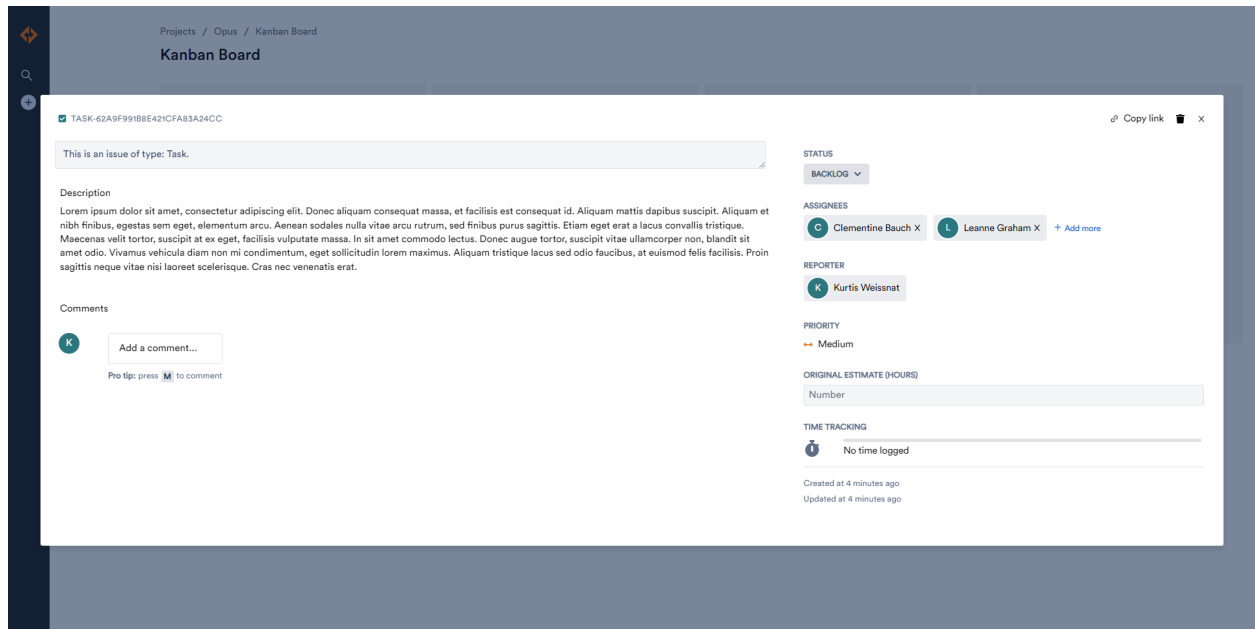


Figure 2 - Détails d'une tâche

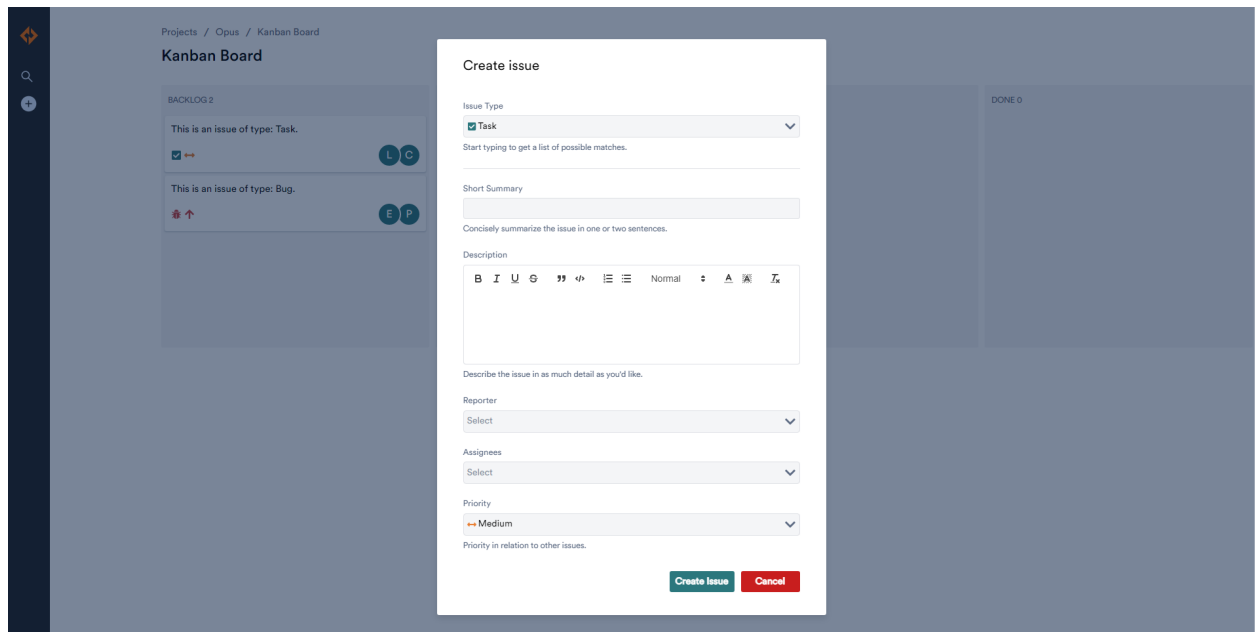


Figure 3 - Création d'une tâche

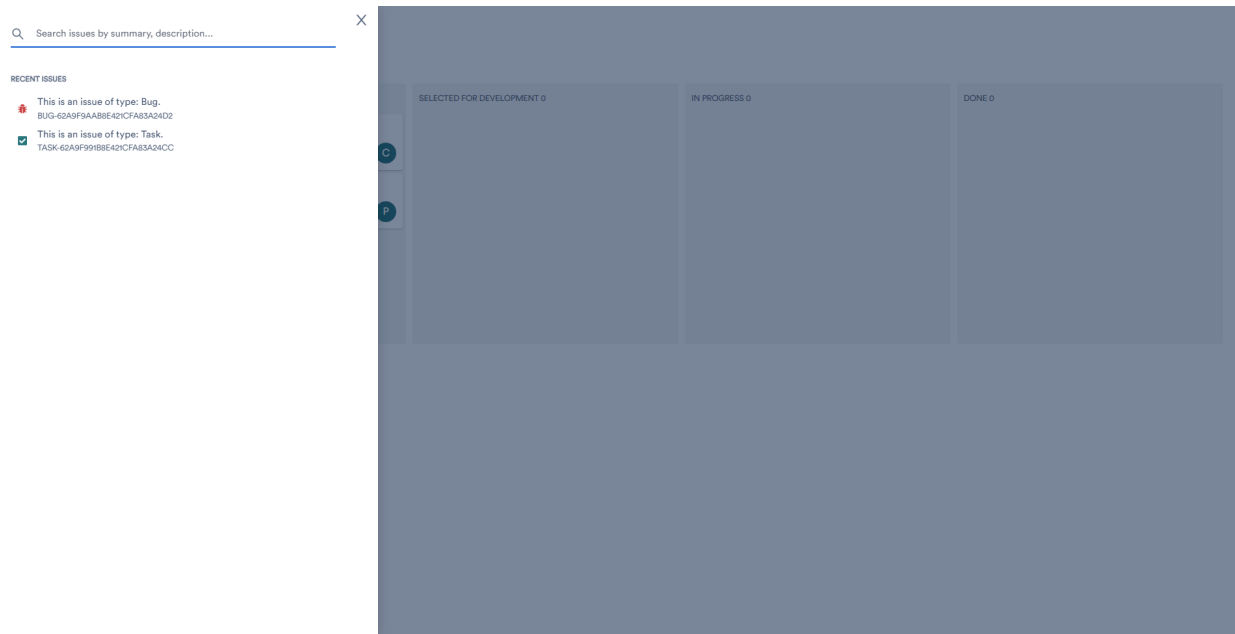


Figure 4 - Recherche des tâches

Conclusion

L'objectif de ce projet était de fournir un moyen plus simple de gérer les projets, qui est conçu pour être utilisé en interne et ne pas dépendre d'une solution externe.

Ce projet est l'aboutissement de tout ce que j'ai appris durant ma formation et mon parcours chez YouCode après beaucoup de travail acharné et de multiples projets réalisés.

Je ne me serais jamais vu arriver seul à ce point. Même s'il s'agit d'un système difficile, le processus d'apprentissage ici chez YouCode nous a donné des bases solides en tant que développeur web fullstack en si peu de temps, sans oublier toutes les compétences générales que nous avons apprises et qui sont très importantes.

Remerciements

Je tiens à exprimer toute ma reconnaissance à YouCode de m'avoir offert cette opportunité.

J'adresse mes sincères remerciements à Monsieur Moulay Youssef SBAI durant cette année, à Monsieur Youness ECHCHADI durant l'année précédente, à Madame Siham OULD HNINI et Monsieur Abdelaziz de m'avoir encadré, orienté, aidé et conseillé.

Je remercie aussi l'administration et le staff, intervenants, camarades et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mon parcours.

Webographie

Documentations:

<https://www.typescriptlang.org/docs/>

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

<https://expressjs.com/en/api.html>

<https://www.mongodb.com/docs/>

<https://mongoosejs.com/docs/api.html>

<https://reactjs.org/docs/getting-started.html>

<https://tailwindcss.com/docs/installation>

Forums:

<https://stackoverflow.com/>

Tutoriels:

<https://www.youtube.com/>

<https://medium.com/>