



Application de gestion de devis  
pour une boutique de réparation d'appareil électronique

Projet réalisé dans le cadre de la présentation au  
**Titre Professionnel : Développeur Web et Web Mobile**

*présenté par*

**Aissam Lamjadab**  
AFPA - Roubaix

Mars 2024

# SOMMAIRE

<b>SOMMAIRE.....</b>	<b>1</b>
<b>&lt;h1&gt;Introduction&lt;/h1&gt;.....</b>	<b>3</b>
<b>&lt;h1&gt;Liste des compétences du référentiel couvertes par le projet&lt;/h1&gt;.....</b>	<b>4</b>
<h2>Développer la partie front-end d'une application web ou web mobile sécurisée</h2>.....	4
<h3>Installer et configurer son environnement de travail en fonction du projet web ou web mobile<h3>.....	4
<h3>Maquetter des interfaces utilisateur web ou web mobile</h3>.....	4
<h3>Réaliser des interfaces utilisateur statiques web ou web mobile</h3>.....	4
<h3>Développer la partie dynamique des interfaces utilisateur web ou web mobile</h3>.....	5
<h2>Développer la partie back-end d'une application web ou web mobile sécurisée</h2>.....	5
<h3>Mettre en place une base de données relationnelle</h3>.....	5
<h3>Développer des composants d'accès aux données SQL et NoSQL</h3>.....	5
<h3>Développer des composants métier côté serveur</h3>.....	6
<h3>Documenter le déploiement d'une application dynamique web et web mobile</h3>.....	6
<b>&lt;h1&gt;Le Contexte du projet&lt;/h1&gt;.....</b>	<b>6</b>
<h2>La présentation de l'entreprise</h2>.....	6
<h2>Le cahier des charges</h2>.....	7
1. Introduction.....	7
1.1 Besoins du Projet.....	7
2. Objectifs.....	7
2.1 Génération de Devis.....	7
2.2 Profil Client.....	7
2.3 Panel Administrateur.....	7
3. Fonctionnalités Requises.....	8
3.1 Génération de Devis.....	8
3.2 Profil Client.....	8
3.3 Panel Administrateur.....	8
4. Architecture et Technologie.....	8
4.1 Langages de Programmation.....	8
4.2 Framework.....	8
4.3 Base de Données.....	8
4.4 Autres Technologies.....	8
5. Design et Ergonomie.....	9
5.1 Interface Utilisateur.....	9
6. Sécurité.....	9
6.1 Certificat SSL.....	9
6.2 Gestion des Droits d'Accès.....	9
<h2>Les contraintes du projet, les livrables attendus</h2>.....	9
1. Les contraintes du projet.....	9
1.1 Contraintes techniques.....	9
1.2 Contraintes de délais.....	10
1.3 Contraintes de ressources.....	10
1.4 Contraintes fonctionnelles.....	10
2. Les livrables attendus.....	10

2.1 Description des livrables.....	11
2.2 Format et échéances.....	11
2.3 Critères d'évaluation.....	11
<b>&lt;h2&gt;L'environnement humain et technique, les objectifs de qualité&lt;/h2&gt;.....</b>	<b>11</b>
1. Environnement humain.....	11
1.1 Responsabilité et rôle.....	11
1.2 Organisation et communication.....	11
2. Environnement technique.....	12
2.1 Infrastructure.....	12
Hébergement et serveurs :.....	12
Base de données :.....	12
Langages de programmation et frameworks :.....	12
Outils de développement :.....	12
Environnement de test :.....	12
Sécurité :.....	12
Fiabilité :.....	13
<b>&lt;h1&gt;Réalisations personnelles&lt;/h1&gt;.....</b>	<b>14</b>
<h2>Le front-end</h2>.....	15
1. La base administrateur.....	16
<h2>Le back-end</h2>.....	40
<b>&lt;h1&gt;Conclusion&lt;/h1&gt;.....</b>	<b>59</b>

# <h1>Introduction</h1>

Mon aventure dans le développement web démarre comme beaucoup d'autres développeurs comme autodidacte. Le développement web m'a semblé être le plus accessible. Après 6 mois de formation sur différentes plateformes de cours en ligne, j'ai acquis de bonnes bases en HTML/CSS.

Afin de connaître les différentes possibilités pour devenir développeur web, je suis passé par la nurserie numérique qui fait découvrir brièvement les différents métiers du numérique. Ma feuille de route (roadmap) se précise pour atteindre mon but, je choisis de faire un TP DWWM pour parfaire mes compétences acquises et en développer de nouvelles. J'aimerai ensuite poursuivre sur un TP CDA. Enfin si j'en ai la possibilité faire un master ou diplôme d'ingénieur en informatique et en cybersécurité.

J'ai eu la chance de participer à toutes les étapes de la conception au développement, ou presque ,mise à part à la mise en production de l'application.

# <h1>Liste des compétences du référentiel couvertes par le projet</h1>

## <h2>Développer la partie front-end d'une application web ou web mobile sécurisée</h2>

### <h3>Installer et configurer son environnement de travail en fonction du projet web ou web mobile</h3>

Après la lecture du cahier des charges et de la demande du gérant de l'entreprise, nous avons mis en place une stack symfony en mode templating.

J'ai choisi d'utiliser Docker pour gérer la base de données MySQL, cette approche me permet d'initialiser les scripts SQL dès l'exécution du Docker Mysql.

Nous avons opté pour Git et GitHub pour gérer les versions de l'application de manière efficace. Avec deux branches, une "main" qui contient le code de production et une "dev" afin d'éviter les conflits sur la branche de production, tester les nouvelles features avant de les envoyer en production.

Nous avons mis en place un Trello pour les users stories à réaliser et un serveur discord pour communiquer de manière synchrone ou asynchrone.

### <h3>Maquetter des interfaces utilisateur web ou web mobile</h3>

A partir du cahier des charges, nous avons pu créer le backlog contenant l'ensemble des users stories, les documents de conception.

Nous avons découpé le projet en deux parties, une partie client et une partie administrateur.

La partie client a été modélisée par un collaborateur et la partie administrateur par moi-même.

J'ai utilisé figma pour modéliser l'ensemble de l'interface administrateur. J'ai gardé les mêmes composants sur toute l'interface et j'ai traduit les différentes fonctionnalités en maquette figma. J'ai gardé un "layout" classique de dashboard pour être utilisé facilement.

### <h3>Réaliser des interfaces utilisateur statiques web ou web mobile</h3>

L'application web est conçue pour être utilisée par tous types de publics, de ce fait elle se doit d'être accessible et simple à utiliser.

Chaque interface utilisateur est intégrée de manière responsive et utilisable sur n'importe quel format d'écran.

La partie administrateur est destinée à être utilisée par le gérant mais aussi d'éventuel employé ou stagiaire, elle est donc simple et rapide d'utilisation. Elle respecte les pratiques courantes dans l'intégration de dashboard administrateur.

En parallèle à la conception du modèle conceptuel de données, nous avons intégré les différentes parties en HTML/CSS/JS sans Twig, afin de ne pas perdre du temps avec les controllers dans symfony.

### <h3>Développer la partie dynamique des interfaces utilisateur web ou web mobile</h3>

Ayant opté pour Twig, le moteur de template de symfony, la partie dynamique est gérée par une seule partie.

Twig nous a permis de créer un modèle réutilisable pour nos deux parties du projet, et d'intégrer facilement les données provenant des controllers symfony.

La gestion des différents événements du DOM, l'animation et les effets visuels sont réalisés par Javascript.

En combinant les fonctionnalités de Twig avec l'interactivité apportée par Javascript, nous avons essayé de créer la meilleure expérience utilisateur.

## <h2>Développer la partie back-end d'une application web ou web mobile sécurisée</h2>

### <h3>Mettre en place une base de données relationnelle</h3>

En collaboration avec le gérant, nous avons fixé les différentes fonctionnalités à implémenter dans l'application à partir du cahier des charges.

Nous avons utilisé Merise qui est une méthode de conception de systèmes d'information. Les concepts de Merise nous ont aidé à structurer et organiser les différentes composantes de notre projet. Nous respectons également les formes normales les plus courantes (1NF, 2NF, 3NF).

En utilisant JMerise comme logiciel de modélisation, nous avons itéré sur le modèle conceptuel de données, enfin générer le modèle logique de données, le dictionnaire de données et le script SQL associé.

Le gérant possède un hébergement mutualisé avec PHP installé et Mysql.

### <h3>Développer des composants d'accès aux données SQL et NoSQL</h3>

Pour accéder aux données présentes dans la base de données, nous avons utilisé une bibliothèque intégrée avec Symfony Doctrine.

Il permet de représenter des données sous forme d'objets PHP.

### <h3>Développer des composants métier côté serveur</h3>

Le gérant souhaite pouvoir intervenir sur le code et donc nous avons dû utiliser symfony qui est un framework PHP qu'il maîtrise.

Symfony utilise par défaut le design pattern MVC qui convient pour notre projet.

Les controllers sont reliés à une route sous formes d'annotations qui peuvent faire appel à différentes action issues de DAO (repository).

En termes de sécurité, nous implémentons différentes parties afin de prévenir les risques connus:

- Prévention d'injection SQL
- Protection contre les attaques CSRF
- Prévention des attaques XSS
- Droit d'accès différents selon le rôle
- Système d'authentification avec hachage

### <h3>Documenter le déploiement d'une application dynamique web et web mobile</h3>

L'hébergement mutualisé limite la personnalisation du serveur qui reçoit l'application.

Le processus de déploiement pourrait être automatisé via GitHub Actions, qui permet de configurer un workflows pour déclencher le déploiement sur l'hébergement chaque fois qu'un nouveau commit est poussé sur la branche de production.

J'ai créé un schéma SQL qui facilite le déploiement de la base de données.

On devra se connecter au serveur via SSH et copier le dossier du projet à l'intérieur, en amont le projet devra être préparé pour la mise en production.

## <h1>Le Contexte du projet</h1>

### <h2>La présentation de l'entreprise</h2>

Mon stage s'est déroulé au sein de DEVFIX SASU, une entreprise de services spécialisée dans la réparation d'ordinateurs et d'appareils personnels et domestiques. DEVFIX réalise des réparations sur une gamme variée d'appareils électroniques, notamment les smartphones, ordinateurs, tablettes et consoles de jeu. En plus de ses services de réparation, l'entreprise propose la vente de produits dérivés de ces appareils.

Fondée le 22 août 2023, DEVFIX est domiciliée au 203 rue Pierre de Roubaix à Roubaix. Elle est dirigée par un gérant, et assistée d'un alternant chargé de la tenue de la boutique, de l'accueil des clients et de la gestion des réparations, sous la supervision du chef d'entreprise.

Dans le cadre de ses ambitions de croissance, le gérant envisage d'étendre la zone de chalandise de l'entreprise, d'abord dans la région Nord puis à travers toute la France. Cette vision a donné naissance au projet de rendre la boutique accessible en ligne, afin de

permettre aux clients de recevoir rapidement des devis, indépendamment de leur localisation géographique.

À l'heure actuelle, la clientèle de DEVFIX est principalement composée de résidents locaux, avec un flux supplémentaire de clients potentiels grâce à son rôle de point relais pour La Poste.

En vue de la future application, DEVFIX a déjà mis en place un hébergement mutualisé et utilise un site Shopify pour la vente de produits dérivés. L'objectif principal de l'application est d'attirer de nouveaux clients, ce qui contribuera à l'augmentation du chiffre d'affaires et à la pérennité de l'entreprise.

Face à un marché en perpétuelle évolution, l'adaptabilité et la croissance sont essentielles pour assurer la pérennité et le succès d'une entreprise comme DEVFIX.

## <h2>Le cahier des charges</h2>

### 1. Introduction

#### 1.1 Besoins du Projet

L'application web DEVFIX de réparation téléphonique vise à permettre aux clients de générer rapidement des devis en sélectionnant leur appareil, la marque, le modèle et la panne, tout en offrant une gestion de profil et un accès à l'historique des devis.

### 2. Objectifs

#### 2.1 Génération de Devis

Le site doit permettre aux utilisateurs de créer des devis en sélectionnant leur appareil, la marque, le modèle et la panne.

#### 2.2 Profil Client

Chaque client doit avoir un profil avec un minimum d'informations telles que le nom, prénom, numéro de téléphone, et adresse e-mail.

#### 2.3 Panel Administrateur

Le panel administrateur doit fournir une vue d'ensemble de l'historique des devis, avec la possibilité d'ajouter de nouvelles marques et modèles. Il doit également gérer la gestion de stock.

### 3. Fonctionnalités Requises

#### 3.1 Génération de Devis

- Choix de la marque
- Choix du modèle
- Choix de la panne
- Calcul automatique du devis.
- Possibilité de visualiser et de modifier le devis pour l'administrateur.

#### 3.2 Profil Client

- Création de profil avec nom, prénom, numéro de téléphone, et adresse e-mail.
- Accès au profil pour consulter l'historique des devis.

#### 3.3 Panel Administrateur

- Vue d'ensemble des Devis
- Historique des devis avec date, heure, référence, et informations client.
- Possibilité de recherche et de filtres pour faciliter la gestion.
- Gestion de Stock
- Liste des pièces détachées avec référence, libellé et quantité en stock.
- Possibilité d'ajouter, modifier ou supprimer des éléments du stock.
- Notifications automatiques en cas de niveau de stock critique.

Exemple de données stockées :

Référence: XYZ123

Libellé: Batterie iPhone 12

Stock Actuel: 50

### 4. Architecture et Technologie

#### 4.1 Langages de Programmation

- Utilisation de **PHP** pour le développement côté serveur.
- Utilisation de **HTML**, **JavaScript**, et **CSS** pour le développement côté client.

#### 4.2 Framework

Mise en place du framework **Symfony** pour le développement du site web.

#### 4.3 Base de Données

Utilisation d'un système de gestion de base de données **MySQL** compatible avec Symfony et **Doctrine**.

#### 4.4 Autres Technologies

Utilisation de technologies complémentaires telles que **Twig** pour les templates HTML.

## 5. Design et Ergonomie

### 5.1 Interface Utilisateur

- Design intuitif pour une expérience utilisateur optimale.
- Utilisation de couleurs et de polices cohérentes avec l'identité visuelle de l'entreprise.

## 6. Sécurité

### 6.1 Certificat SSL

- Mise en place d'un certificat SSL pour assurer la sécurité des données client.
- Captcha pour le formulaire de contact

### 6.2 Gestion des Droits d'Accès

Mise en place de mécanismes pour garantir l'accès sécurisé aux informations sensibles.

## <h2>Les contraintes du projet, les livrables attendus</h2>

### 1. Les contraintes du projet

#### 1.1 Contraintes techniques

Le langage PHP avec comme framework Symfony sont imposés dans le cahier des charges. De plus, l'hébergement sur un serveur mutualisé limite les choix possibles de notre stack technique.

D'un côté, cela nous a épargné l'enquête et le choix d'une stack technique, d'un autre côté, j'aurais aimé choisir une stack après la comparaison des différentes possibilités.

PHP est un langage à typage faible et interprété, ce qui signifie que les erreurs de typage ne sont détectées qu'à l'exécution. Cela multiplie les possibilités d'erreurs ou de bugs. De plus, le temps de code est plus long, il faut jongler entre le navigateur et le code pour déboguer l'application, il faut installer une extension PHP, Xdebug pour pouvoir suivre l'évolution des variables et mettre des points d'arrêts dans le code, avec l'extension vscode PHP debug.

Étant mon premier projet Symfony, il a fallu appréhender ce framework complet comprenant de nombreuses fonctionnalités.

Configurer mon docker m'a apporté son lot de configuration additionnelle pour que le container s'exécute prêt à être utilisé.

L'utilisation du css nativement augmente les difficultés en termes de maintenabilité, j'aurais préféré l'utilisation de Sass ou Tailwind css pour faciliter les modifications.

Modification de la configuration à plusieurs reprises dû à une mauvaise gestion des push sur le repository github, il aurait peut être préférable de créer une branche par développeur ainsi garder en interne les changements de configuration personnalisé. En effet, en utilisant un docker, j'ai dû adapter ma configuration.

## 1.2 Contraintes de délais

On a dû démarrer le projet à partir du cahier des charges sans aucune base.

Réaliser le modèle de donnée, créer le backlog avec la liste des users stories, créer le projet symfony, configurer le projet, , le script SQL, générer les entities, développer les pages en front-end, les controllers, les repositories, les templates et l'ensemble de ses tâches devaient être réalisé en 1 mois.

L'application coté client est terminée, il reste des parties administrateurs à finaliser, notamment créer les différentes routes dans les contrôleurs ainsi qu'ajouter si nécessaire des méthodes dans les repositories.

Nous avons réalisé en parallèle le modèle de données et le maquettage sans faire de wireframe pour la partie administrateur, en effet celle-ci reste très classique comme interface administrateur.

De plus, nous avons développé la partie front-end à part en HTML/CSS/JS sans intégrer Twig dans le processus de développement.

La création de template Twig s'est faite au moment de la création de l'ensemble de la route responsable de délivrer les données.

## 1.3 Contraintes de ressources

Aucun budget n'était nécessaire, chacun possède son propre matériel.

Nous sommes tous sur vscode pour développer l'application.

L'équipe de développement était relativement du même niveau avec une partie pouvant aider l'autre partie sans difficulté.

Si besoin, nous pouvions faire appel à l'aide d'un développeur-formateur expérimenté lors d'éléments bloqués pour notre progression dans le projet.

## 1.4 Contraintes fonctionnelles

La gestion de la bonne redirection selon le rôle devait bien être gérer, le panel administrateur ne doit surtout pas être accessible au public.

Le choix de l'équipe s'est porté sur une authentification sur la page d'accueil de l'application web avec une redirection selon le rôle du l'utilisateur connecté, néanmoins j'avais suggéré de créer une page distincte pour l'administrateur et d'en limiter l'accès en définissant des adresses ip d'accès prédéfinies.

# 2. Les livrables attendus

## 2.1 Description des livrables

Le code source du projet sur GitHub

## 2.2 Format et échéances

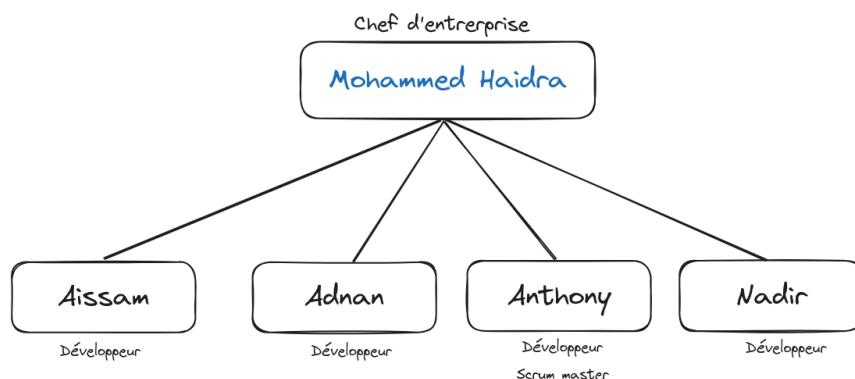
L'application devait être fonctionnelle et accessible depuis notre repository GitHub.

## 2.3 Critères d'évaluation

- Le seul critère qui pourrait être retenu, serait de livrer une application fonctionnel et simple d'utilisation

## <h2>L'environnement humain et technique, les objectifs de qualité</h2>

### 1. Environnement humain



#### 1.1 Responsabilité et rôle

- Adnan responsable de la partie client
- Aissam responsable de la partie administrateur
- Anthony scrum master de l'équipe assurant le bon déroulé du projet
- Nadir était notre Happiness Officer

#### 1.2 Organisation et communication

La première réunion se fit en présentiel dans un espace de coworking (pleine image) pour réaliser le modèle conceptuel de données.

A chaque début de semaine, il y avait une réunion organisée qui permettait de connaître l'avancement des users stories, la définition des objectifs de la semaine.

Nous communiquons quotidiennement sur notre serveur discord sur l'avancement des users stories ou résolutions de problèmes. Il y avait un salon dédié pour communiquer de manière vocale ou pour partager nos écrans.

Le gérant pouvait suivre nos conversations, en outre suivre l'avancement du développement de l'application. Il pouvait ainsi estimer l'implication des différents développeurs.

L'ensemble de la liste des tâches, users stories étaient regroupés grâce à Trello.

## 2. Environnement technique

### 2.1 Infrastructure

Hébergement et serveurs :

Devfix possède un hébergement mutualisé chez IONOS, accessible via le protocole SFTP + SSH

Base de données :

Nous utilisons une base de données Mysql hébergée sur le même serveur que l'application.

Langages de programmation et frameworks :

- PHP avec Symfony
- HTML/CSS/JS
- Twig pour le moteur de template

Outils de développement :

- Vscode
- Docker
- Xdebug
- web-profiler-bundle

Environnement de test :

Les tests ont été effectués manuellement par chaque développeur, néanmoins le manque de temps nous a poussé à laisser de côté cette partie.

- PHPUnit est installé, il suffit de créer un fichier dans le dossier tests pour créer un test

Sécurité :

- Mot de passe hashé grâce à l'algorithme Argon2i
- Mot de passe solide avec plus de 8 caractères, incluant au moins un chiffre et un caractère spécial
- Accès restreint par rôle
- Utilisation de doctrine et du DQL pour les requêtes en database
- Contrôle de saisie avec condition de types et de valeurs

- Typage des variables en PHP, possibilité de basculer PHP en mode strict depuis Symfony 7 avec " declare(strict\_types=1); "
- Système d'authentification pour accéder aux informations personnels
- Activation de l'option qui protège contre les attaques CSRF<sup>1</sup>, symfony génère des jetons CSRF uniques pour chaque formulaire et qui ensuite permette de vérifier que chaque requête issue d'un formulaire est légitime et non forgée

J'ai vu qu'il était possible de réalisé un scan des vulnérabilités sur  
[https://owasp.org/www-community/Vulnerability\\_Scanning\\_Tools](https://owasp.org/www-community/Vulnerability_Scanning_Tools)

Fiabilité :

Le projet étant hébergé sur un serveur, en cas d'erreur ou de bug, il n'est pas aisément d'intervenir depuis le terminal même si cela reste possible.

Nous n'avons pas réfléchi à une stratégie de sauvegarde et de récupération des données. Je sais qu'il existe des moyens de sauvegarde au sein de MySQL en saisissant dans le shell différentes commandes ou directement dans Phpmyadmin.

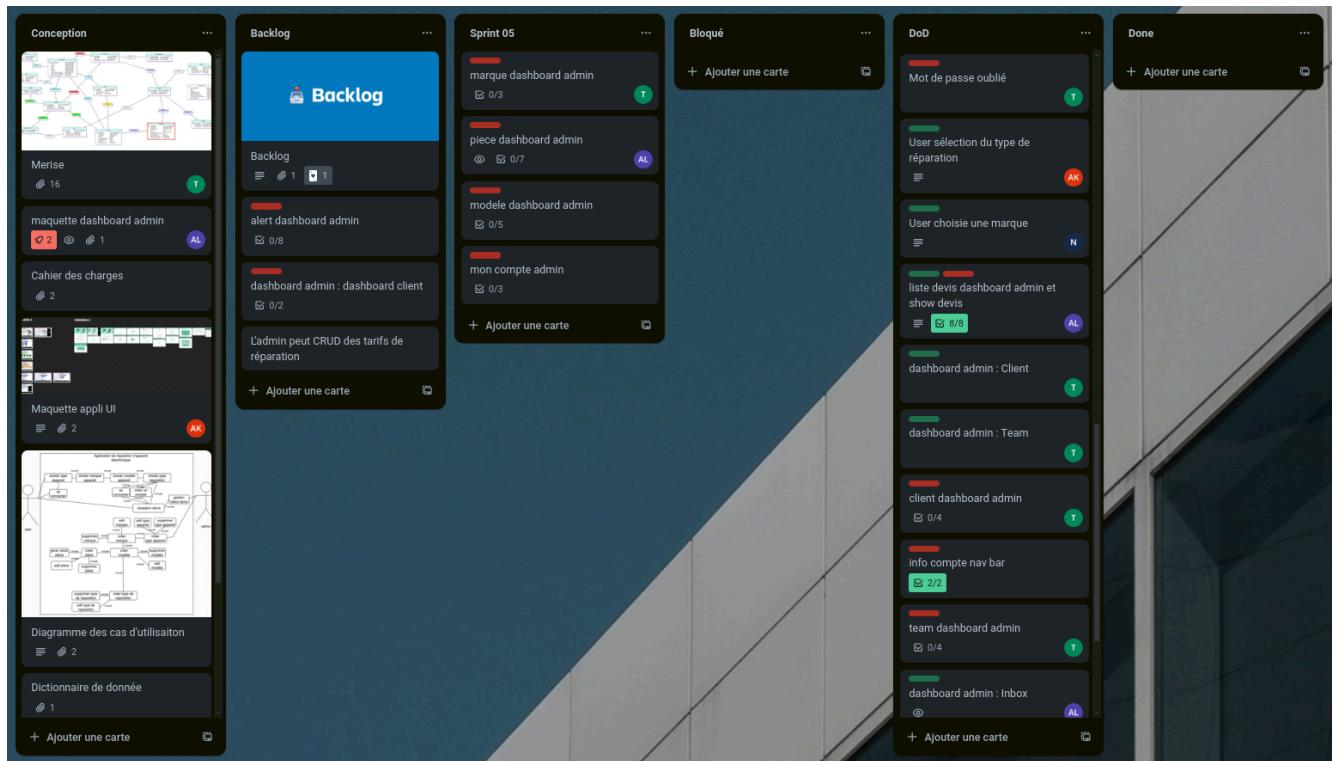
Autrement appelé un Dump de la database.

---

<sup>1</sup> Falsification de requête inter-sites, une vulnérabilité de sécurité où un attaquant peut tromper un utilisateur pour qu'il effectue des actions non voulues sur un site web dans lequel il est authentifié.

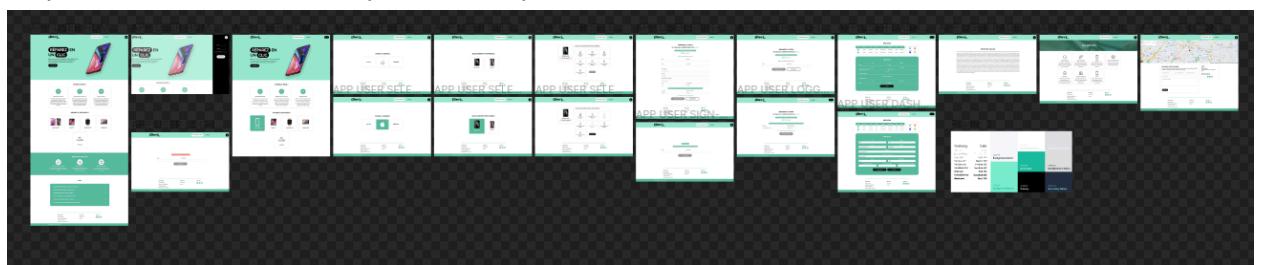
# Réalisations personnelles

Nous avons créée et répartit les users stories sur Trello, voici une aperçu du Tableau à ce jour :



Nous avons mis l'accent sur la partie visible pour le client, voici l'ensemble des maquettes de la partie client que je ne présenterais pas dans ce rapport.

Maquette vue d'ensemble de la partie visible pour le client



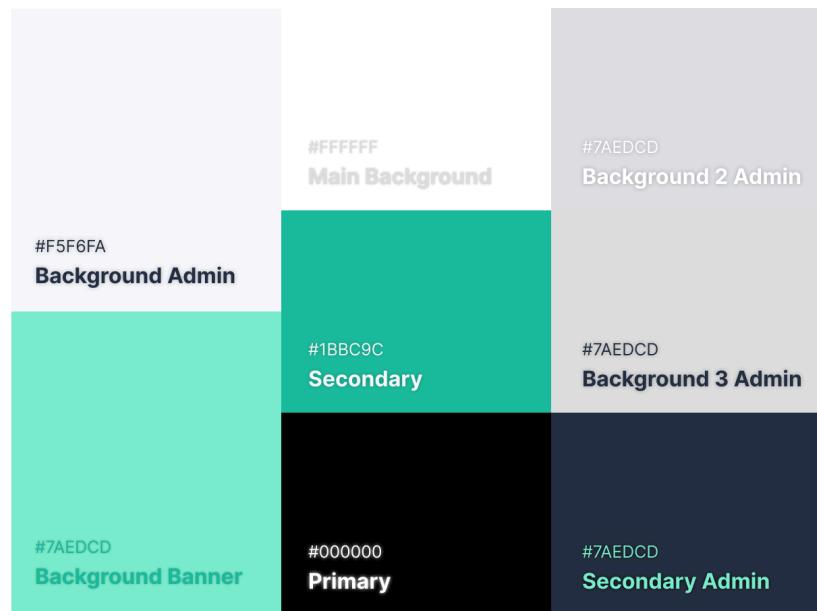
La partie administrateur est quant à elle visuellement développée en HTML/CSS/JS, mais il manque la création des routes correspondantes, il y a deux fonctionnalités finalisées dans le panel administrateur que je vais vous présenter dans la partie back-end.

Je n'ai pas réalisé de wireframe pour la partie administrateur, j'ai réalisé directement le mockup de l'application entière à partir des différentes fonctionnalités à implémenter.

L'application Devfix suit les conseils en matière de design, en limitant le nombre de couleur utilisé, définissant pas plus de deux polices.

*Charte graphique de l'application*

Raleway	Lato
Thin 100	Thin 100
ExtraLight 200	ExtraLight 200
Light 300	Light 300
Regular 400	Regular 400
Medium 500	Medium 500
SemiBold 600	SemiBold 600
<b>Bold 700</b>	<b>Bold 700</b>
<b>ExtraBold 800</b>	<b>ExtraBold 800</b>
<b>Black 900</b>	<b>Black 900</b>



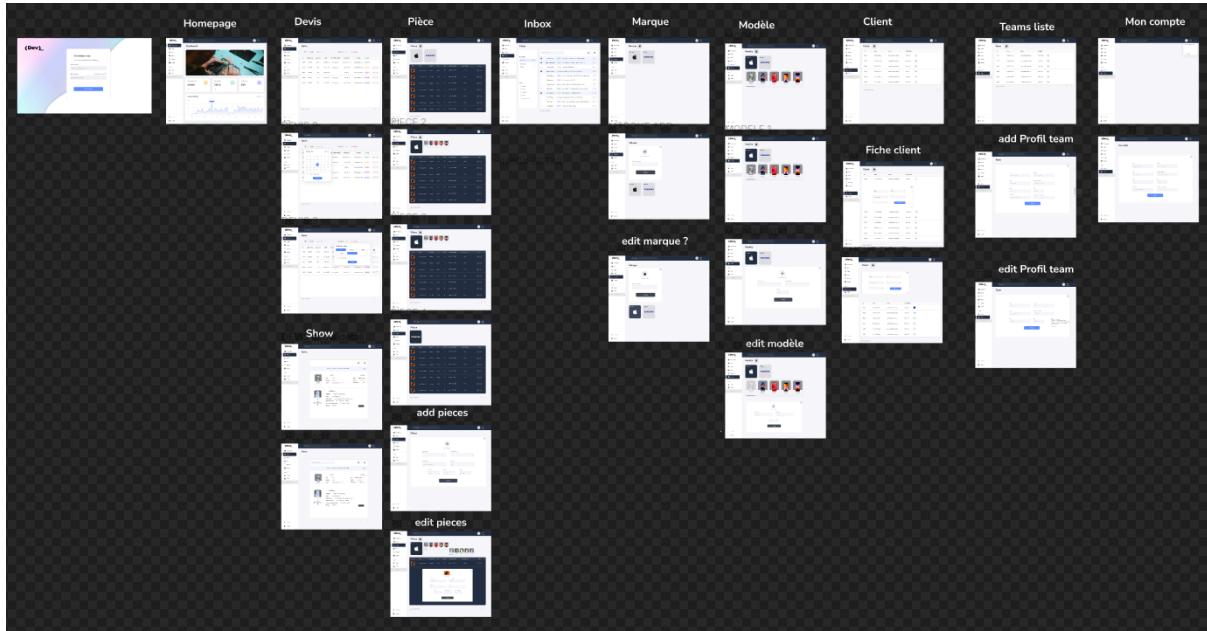
## <h2>Le front-end</h2>

Dans un premier temps, je présenterai l'aspect visuel de l'application et dans un deuxième temps des extraits de code.

J'ai réalisé la maquette de l'application côté administrateur en utilisant Figma. Un outil formidable pour la création de maquette.

J'ai gardé les mêmes composants sur toute la partie administrateur, afin de faciliter le développement front-end de l'application.

Maquette Vue d'ensemble de l'espace administrateur



Le header et la barre de navigation ont été insérés dans le template de base Twig, il a suffit de l'implémenter qu'une seule fois dans le code pour la retrouver sur chaque page, c'est l'une des fonctionnalités qu'offre Twig qui accélère considérablement le processus de développement.

Lors de mon stage, j'ai pu développer les pages ou composants suivantes :

- Le header et la barre de navigation faisant partie de la base template Twig
- Les pages Devis
- Les pages Pièces
- La pages Inbox
- Les pages marque

Les pages devis et la page Homepage sont fonctionnelles. Il seront donc détaillés côté back-end également.

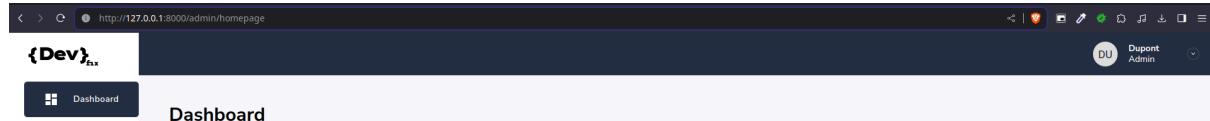
## 1. La base administrateur

Le **header** est composé d'un avatar avec le nom de l'utilisateur connecté et son rôle soit administrateur ou employé pour cette partie qui est accessible qu'à ses deux rôles. Avec un dropdown permettant d'accéder à son espace compte, de créer un compte administrateur/employé ou de se déconnecter

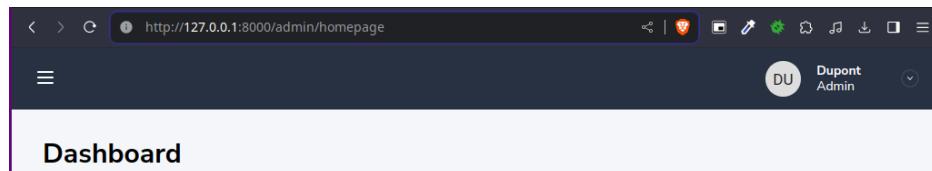
Maquette du header administrateur



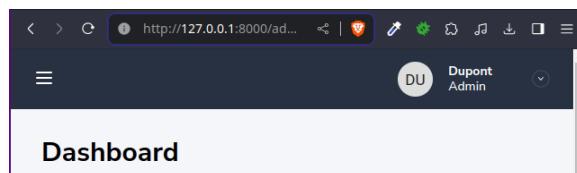
### *Vue du header : desktop*



### *Vue du header : Tablet*



### *Vue du header : mobile*



J'ai développé la partie responsive en utilisant des "media query", et en transformant la barre de navigation en menu burger, ainsi permettre de libérer de la place pour le contenu de la page.

Présentation du code du head, il permet d'adapter des blocs dans chaque template.

### *Code html du head*

```
base_admin.html.twig templates/base_admin.html.twig
You, yesterday | 2 authors (You and others)
5  <!DOCTYPE html>
4  <html dir="ltr" lang="fr">
3  	<head>
2  	<meta charset="UTF-8">
1  	<meta name="viewport" content="width=device-width, initial-scale=1">
6  	<meta http-equiv="content-type" content="text/html; charset=utf-8"> You, yesterday via PR #1 • fix: dashboard client
1  	<!-- Document Title -->
2  	<meta name="keywords" content="DevFix">
3  	<meta name="description" content="DevFix dashboard administrateur">
4  	<title>
5  	{%
6  	block title %}
7  	{%
8  	endblock %}
9  	</title>
10 	<!-- Favicon -->
11 	<link
12 	rel="shortcut icon" href="{{ asset('user/images/favicon.png') }}"/>
13 	<!-- font -->
14 	<link rel="preconnect" href="https://fonts.googleapis.com">
15 	<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
16 	<link href="https://fonts.googleapis.com/css2?family=Nunito+Sans:ital,opsz,wght@0,6..12,200;0,6..12,300;0,6..12,400;0,
17 	6..12,500;0,6..12,600;0,6..12,700;0,6..12,800;0,6..12,900;0,6..12,1000;1,6..12,400&display=swap" rel="stylesheet">
18
19 	{%
20 	block stylesheets %}
21 	<link rel="stylesheet" href="{{ asset('admin/css/main.css') }}" type="text/css">
22 	<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
23 	{%
24 	endblock %}
25
26 	{%
27 	block javascripts %}
28 	<script src="{{ asset('admin/js/main.js') }}" defer></script>
29 	{%
30 	endblock %}
31
32 	</head>
```

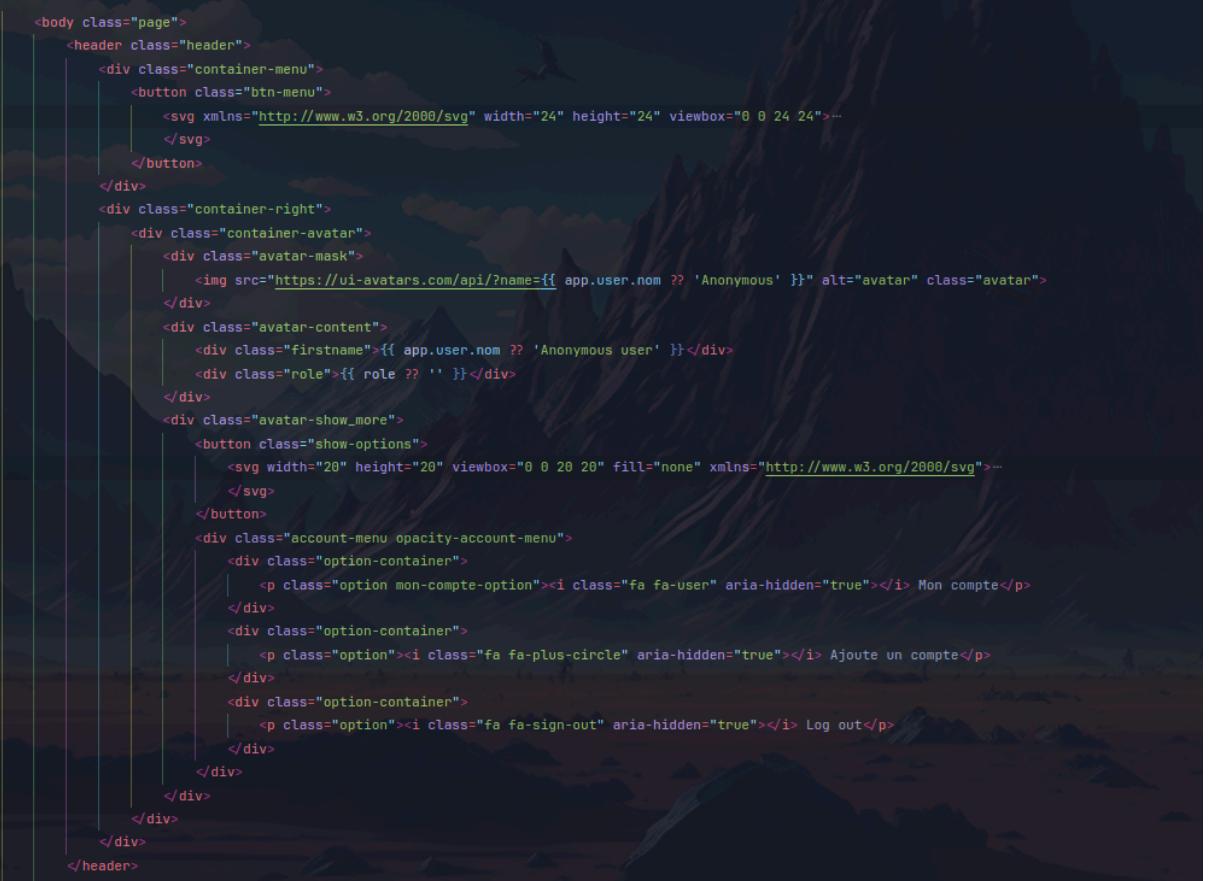
Cette partie permet à chaque développeur d'adapter selon ses besoins le titre de la page qui s'affiche dans l'onglet du navigateur.

J'importe le favicon de l'application grâce à la fonction asset qui permet de récupérer les différents fichiers stockés dans le dossier "assets" en précisant le chemin voulu. L'ensemble des éléments externes au code PHP sont généralement stockés à cet endroit, on y a mis les fichiers de style css, les images et le javascript.

Ensuite j'utilise une police issue de google-font, cela garantit qu'elle sera disponible sur l'ensemble de l'application côté administrateur.

J'inclue un bloc pour les fichiers css avec la fonction asset et un autre pour le javascript. De cette manière, chaque développeur pourra ajouter ses fichiers css ou javascript en incluant ou non les fichiers utilisés dans la base administrateur.

*Code html du header*



```
<body class="page">
  <header class="header">
    <div class="container-menu">
      <button class="btn-menu">
        <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">...
```

```
      </svg>
    </button>
  </div>
  <div class="container-right">
    <div class="container-avatar">
      <div class="avatar-mask">
        
      </div>
      <div class="avatar-content">
        <div class="firstname">{{ app.user.nom ?? 'Anonymous user' }}</div>
        <div class="role">{{ role ?? '' }}</div>
      </div>
    <div class="avatar-show_more">
      <button class="show-options">
        <svg width="20" height="20" viewBox="0 0 20 20" fill="none" xmlns="http://www.w3.org/2000/svg">...
```

```
        </svg>
      </button>
      <div class="account-menu opacity-account-menu">
        <div class="option-container">
          <p class="option mon-compte-option"><i class="fa fa-user" aria-hidden="true"></i> Mon compte</p>
        </div>
        <div class="option-container">
          <p class="option"><i class="fa fa-plus-circle" aria-hidden="true"></i> Ajoute un compte</p>
        </div>
        <div class="option-container">
          <p class="option"><i class="fa fa-sign-out" aria-hidden="true"></i> Log out</p>
        </div>
      </div>
    </div>
  </div>
</header>
```

J'inclue le nom et le rôle de l'utilisateur connecté, l'utilisateur non connecté ne devrait pas pouvoir accéder à cet espace administrateur, en effet dans le code ci-dessus j'ai utilisé un opérateur de fusion null pour fournir une valeur par défaut pour le cas où la variable serait null.

Néanmoins, cette opérateur n'est plus nécessaire en environnement de production, en effet lors du développement l'authentification n'était pas encore finalisé, ce qui n'est plus le cas.

Voici quelques extraits du code css de la base administrateur.

## Code css main

```
main.css assets/admin/css/main.css/.container-menu
You, 5 days ago | 2 authors (You and others)
30 :root {
29   background: #F5F6FA;
28   line-height: 1;
27   font-family: "Nunito Sans";
26   font-size: 14px;
25   --sidebar-width: 15rem;
24   --background: #9c88ff;
23   --navbar-width: 256px;
22   --navbar-width-min: 80px;
21   --color-primary: #fff;
20   --color-secondary: #273142;
19 }
18
17 /* html5resetcss */
16 *, ::before, ::after {box-sizing: border-box; margin: 0; padding: 0;}
15
14 button {
13   cursor: pointer;
12 }
11 .header {
10   display: flex;
9   align-items: center;
8   justify-content: space-between;
7   padding-top: 1rem;
6   padding-bottom: 1rem;
5   background: #273142;
4 }
3
2 .container-menu {
1   display: none;
31 } You, 2 weeks ago via PR #1 · feat: add base admin twig template in /templates ...
1
2 .container-right {
3   display: flex;
4   align-items: center;
5   width: 100%;
6   padding-left: 4.875rem;
7   padding-right: 1.875rem;
8   justify-content: flex-end;
9 }
10
11 .container-avatar {
12   display: flex;
13   align-items: center;
14 }
```

Je définit des variables css que je peux utiliser dans toutes les css enfants de ce fichier, il doit donc rester en premier dans le bloc stylesheet.

De plus, les éléments réutilisables sont stylisés en priorité dans ce fichier.

### Code css main

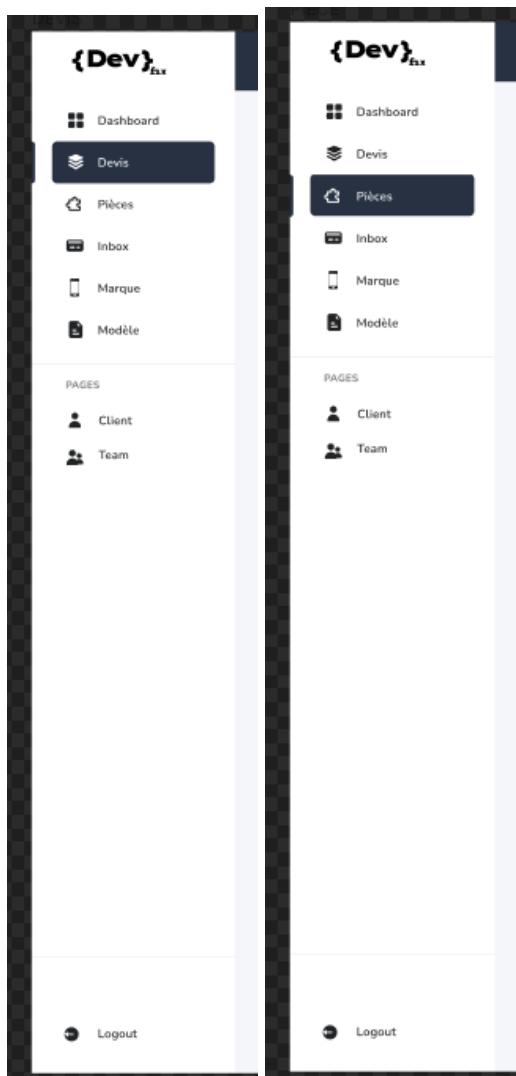
```
6 .avatar-mask {
5   border-radius: 50%;
4   overflow: hidden;
3   display: flex;
2   align-items: center;
1   justify-content: center;
0   margin-right: 1.25rem;
9 }
8 .avatar {
7   width: 2.75rem;
6 }
5
4 .avatar-content {
3   color: #fff;
2   font-size: 14px;
1   font-weight: 700;
0   margin-right: 3.125rem;
9 }
8
7 .firstname {
6   margin-bottom: 0.2rem;
5 }
4
3 .role {
2   font-weight: 500;
1 }
0
9 .show-options {
8   display: flex;
7   align-items: center;
6   justify-content: center;
5   background: none;
4   border: none;
3   cursor: pointer;
2   transition: .4s ease-out;
1 }
0
9 .show-options:hover svg {
8   stroke: #3a3a3ae5;
7   fill: #CFCFCF
6 }
```

J'utilise les rem comme unités de mesures qui est relative à la taille de la police de l'élément racine. Elle permet également de s'adapter à la taille de la police du navigateur, ce qui améliore l'accessibilité et la lisibilité de l'utilisateur.

J'utilise également beaucoup les flexbox dans mon code css, cela permet d'aligner facilement les différents éléments ainsi que rendre responsive en ajoutant flex-wrap par exemple ou en changeant le flex-direction dans un media query

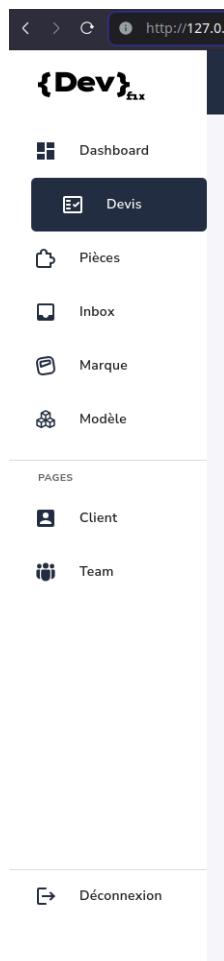
**La barre de navigation** (sidebar) est composée d'une liste de liens pour naviguer dans l'espace administrateur entre les différentes pages, la page courant est identifiée grâce à une couleur de fond différente des autres.

### Maquette sidebar

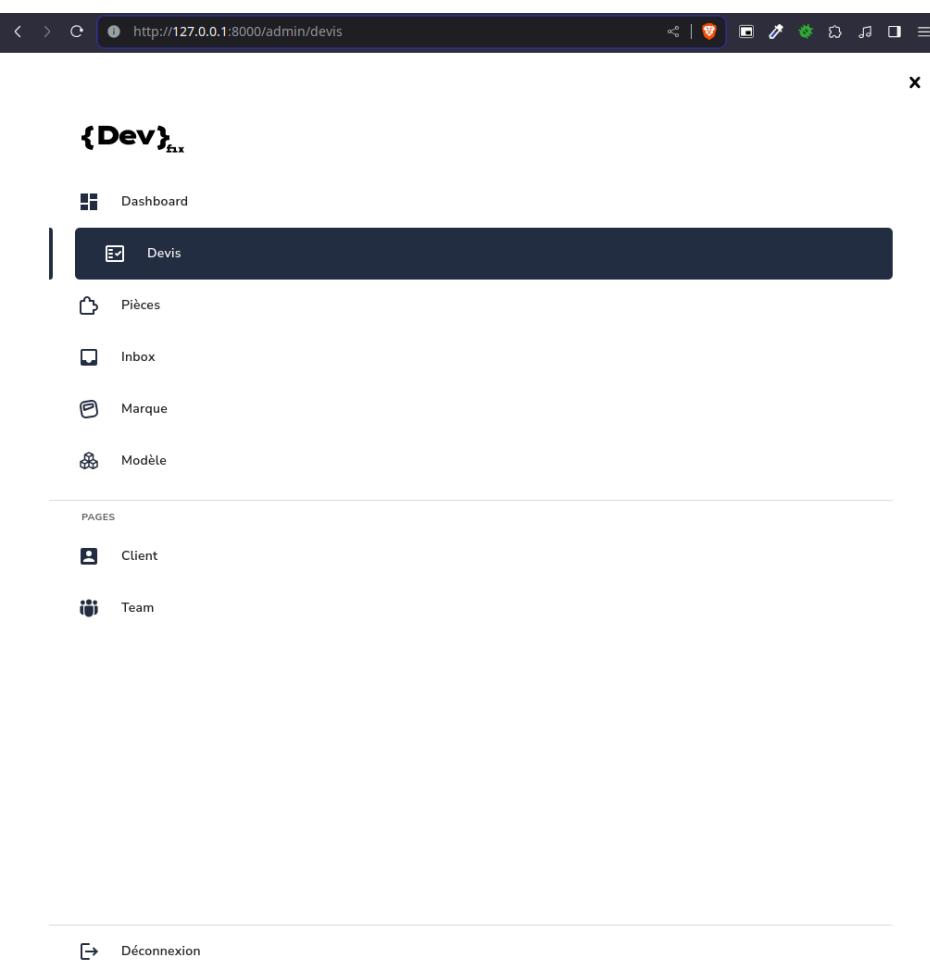


Cette barre est masquée lorsque l'écran devient trop petit pour contenir le contenu de la page. Celle-ci prend toute la page lors du clique sur le menu burger.

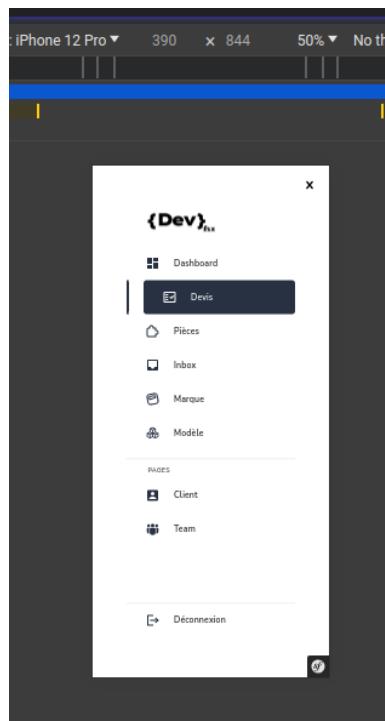
Vue sidebar desktop



Tablette



Mobile



J'utilise les media query pour masquer la sidebar, ensuite j'ajoute un écouteur d'événement au menu burger, lors du clique j'ajoute une class à l'élément sidebar qui change le display, la largeur, le padding et le z-index.

C'était la manière la plus simple d'implémenter un menu burger dans l'application sans de changement majeur sur le code de la sidebar.

En effet, le panel administrateur n'était pas destiné à être consulté sur d'autres appareils que le desktop, il a fallu adapter le code après la phase de développement du front-end.

Néanmoins, ce fut très rapide, ayant une bonne connaissance du css, je n'ai eu aucun mal à adapter le code.

#### Code html sidebar

```
<aside class="side-bar">
    <button class="cancelShow">
        <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 20 20">...
    </svg>
</button>

<nav class="nav_aside">
    <ul class="list-nav-top">
        <li class="container-nav-link">
            <a href="{{path('homepage_admin')}}" class="nav-link">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">...
                    <path fill="currentColor" d="M13 9V3h8v6zM3 13V3h8v10zm10 8V11h8v10zM3 21v-6h8v6z" />
                </svg>
                <span class="nav-link-text">Dashboard</span>
            </a>
        </li>
        <li class="container-nav-link ">
            <a href="{{path('admin_devis')}}" class="nav-link">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">...
                </svg>
                <span class="nav-link-text">Devis</span>
            </a>
        </li>
        <li class="container-nav-link ">
            <a href="{{path('admin_pieces')}}" class="nav-link">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">...
                </svg>
                <span class="nav-link-text">Pièces</span>
            </a>
        </li>
        <li class="container-nav-link">
            <a href="{{ path('admin_inbox') }}" class="nav-link">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">...
                </svg>
                <span class="nav-link-text">Inbox</span>
            </a>
        </li>
        <li class="container-nav-link">
            <a href="{{path('admin_marque')}}" class="nav-link">...
            </a>
        </li>
        <li class="container-nav-link">
            <a href="{{path('admin_modele')}}" class="nav-link">...
            </a>
        </li>
    </ul>
</nav>
```

J'utilise la fonction path qui permet de renvoyer la route citée dans les parenthèses. J'ai choisi d'utiliser des SVG directement dans le code afin de pouvoir changer la couleur facilement ainsi avoir un contrôle total sur les images.

#### Code css sidebar

```
1  @media screen and (max-width: 1050px) {  
2      .container-empty {  
3          width: auto;  
4      }  
5  
6      .side-bar {  
7          display: none;  
8          padding: 4rem;    You, 2 wee  
9      }  
10  
11     .cancelShow {  
12         position: absolute;  
13         top: 20px;  
14         right: 20px;  
15         color: #000;  
16     }  
17  
18     .container-menu {  
19         display: flex;  
20         align-items: center;  
21         justify-content: center;  
22         padding-left: 1.8rem;  
23     }  
24  
25     .btn-menu {  
26         background: none;  
27         border: none;  
28         color: #fff;  
29         cursor: pointer;  
30     }  
31  
32     .btn-menu:hover {  
33         color: #fffff72;  
34     }  
35  
36     .main {  
37         margin: 1rem;  
38     }  
39  
40     .toggleShow {  
41         display: block;  
42         width: 100vw;  
43         padding: 4rem;  
44         z-index: 2;  
45     }  
46  
47  .side-bar {  
48      position: fixed;  
49      width: var(--sidebar-width);  
50      height: 100%;  
51      background: #fff;  
52      top: 0;  
53      left: 0;  
54  }  
55  
56  .nav_aside ul {  
57      list-style: none;  
58  }  
59  
60  .logo {  
61      margin-left: 2rem;  
62      margin-bottom: 0.625rem;  
63  }  
64  
65  .list-nav-top {  
66      height: 25%;  
67  }  
68  
69  .list-nav-mid {  
70      height: 40vh;  
71  }  
72  
73  .list-nav-bot {  
74      height: 5%;  
75  }  
76  
77  .container-nav-link {  
78      position: relative;  
79      height: 4rem;  
80      padding-left: 2rem;  
81      display: flex;  
82  }  
83  
84  .container-nav-link>a {  
85      width: 100%;  
86  }  
87  
88  .container-nav-link>a>svg {  
89      color: #273142;  
90  }  
91  
92  .active-page>a {  
93      color: #fff;  
94      background: var(--color-secondary);  
95      padding: 0 2rem;  
96      border-radius: 0.375rem;  
97  }
```

J'utilise un media query pour changer le display de la sidebar, je masque la sidebar qui apparaît en position absolue au-dessus de toutes éléments prenant toute la largeur de l'écran.

J'ajoute lors du clique une class à la sidebar grâce à la création d'un écouteur d'événement en javascript

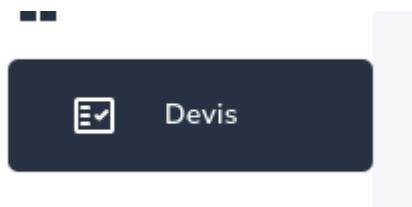
Code javascript toggleSideBar

```
function displayNavBar() {
    const sidebar = document.querySelector('.side-bar');
    const closeBar = document.querySelector('.cancelShow')
    sidebar.classList.add('toggleShow');
    closeBar.style.display = "block";
}

function maskNavBar(){
    const sidebar = document.querySelector('.side-bar');
    const closeBar = document.querySelector('.cancelShow')
    sidebar.classList.remove('toggleShow');
    closeBar.style.display = "none";
}

const btnMenu = document.querySelector(".btn-menu");
const btnClose = document.querySelector(".cancelShow");
if(btnMenu !== undefined || btnMenu !== null){
    btnMenu.addEventListener("click", displayNavBar)
}
if(btnClose !== undefined || btnClose !== null){
    btnClose.addEventListener("click", maskNavBar)
}
```

Je sélectionne les éléments html grâce au querySelector, ensuite je vérifie que l'élément existe bien avant d'ajouter un écouteur d'événement au clique sur le bouton menu (menu burger), enfin j'ajoute également un écouteur d'événement au clique sur le bouton qui permet de fermer le menu de navigation.



Cet élément est également dynamique, il s'adapte à la page courante. J'ajoute une classe au lien s'il est également au chemin de la page en cours .

```
const links = document.querySelectorAll('.container-nav-link');
let currentPage = window.location.pathname;

if (links.length > 0 || links !== undefined || links !== null) {
    links.forEach((link) => {
        if (link.children[0].getAttribute('href') === currentPage) {
            link.classList.add('active-page');
        }
    })
}
```

Les pages Devis font partie des fonctionnalités essentielles de l'application Devfix, je vais détailler ses différentes pages afin d'introduire la partie backend associé. Pour le reste des pages développées, n'ayant que le front-end, je ne mettrai que des captures de pages.

### *Vue de la page pièce*

Pièces	+						
Filtrer par marque	Filtrer par modèle						
Image	Libellé	réf fabricant	Prix	Quantité	Modèle compatible	Délai de livraison	Action
	EcranLCD iphone X	top screen	25.00 €	63	Iphone X, Iphone 12	4 jours	
	EcranLCD iphone X	top screen	25.00 €	63	Iphone X, Iphone 12	4 jours	
	EcranLCD iphone X	top screen	25.00 €	63	Iphone X, Iphone 12	4 jours	

page 1 - 1 lignes 6

### *Vue modifier une pièce*

Ajouter une image

Libellé de la pièce

Référence fabricant

Selectionner une marque

Modèle compatible \*press 'maj' pour multi-select  

Iphone X  
Iphone 11  
Iphone 12  
Iphone 13  
Iphone X

Stock

Prix

Délai de livraison

### Vue de la page Inbox

The screenshot shows the 'Inbox' page interface. On the left, there's a sidebar with 'Mes alertes' containing 'Inbox' (0), 'Archive' (0), and 'Trash' (0). Below it is a 'Label' section with color-coded boxes for 'Devis' (green), 'Pièces' (blue), 'Employé' (orange), and 'Client' (purple). The main area has a search bar at the top. It displays two items: 'Nouveau devis' (Devis tab selected) for 'iphone 15 - écran cassé - piece 00545 - DEV0001' at 10:30, and 'Pièces manquantes' (Pièces tab selected) for 'piece 5498311 ecran 4648 - iphone 35 - stock = 0' at 12:30. At the bottom left, it says 'page 1-1 lignes 6'. Navigation arrows are at the bottom right.

### Vue de page Marque

The screenshot shows the 'Marque' creation page. It features a logo input field with a placeholder 'Ajouter un logo' and a 'Nom de la marque' input field with the value 'apple'. A large 'Valider' button is centered below. At the bottom, there are two cards: one for '30 modèles' with an Apple logo, and another for '10 modèles' with a Samsung logo.

## Les pages Devis

L'administrateur doit pouvoir consulter la liste des devis, les filtrer par date ou par statut, être capable de voir les détails d'un devis, pouvoir changer le statut du devis et son responsable.

J'ai créé trois template Twig pour ses différentes fonctionnalités.

Un pour afficher la liste des devis, un autre pour la liste filtrer des devis et enfin un dernier pour afficher les détails d'une devis.

Voyons tout d'abord les maquettes de ses pages.

#### Maquette liste de devis

The screenshot shows the 'Devis' (Quotation) list page. The left sidebar contains navigation links: Dashboard, Devis (selected), Pièces, Inbox, Marque, and Modèle. Under PAGES, there are Client and Team links. At the bottom left is a Logout button. The main content area has a header 'Devis' with filter buttons for 'Filter By' (dropdown), 'Date' (dropdown), 'Order Status' (dropdown), and a 'Reset Filter' button. Below is a table with columns: ID, NUM\_DEVIS, PRIX\_TTC, DATE, DATE\_RESTITUTION, TELEPHONE, CONTACT, and STATUS. The table contains five rows of data:

ID	NUM_DEVIS	PRIX_TTC	DATE	DATE_RESTITUTION	TELEPHONE	CONTACT	STATUS
00001	DE00001	250.55 €	04 Sep 2019	28 May 2019	iphone 13 pro max	client@example.com	Terminé
00002	DE00002	50 €	04 Sep 2019	28 May 2019	samsung S23	client@example.com	En cours
00003	DE00003	10 €	04 Sep 2019		iphone X	client@example.com	Rejeté
00004	DE00004	366 €	04 Sep 2019	04 Sep 2019	iphone 15	client@example.com	Pièce en attente
00005	DE00005	650€	04 Sep 2019	04 Sep 2019	iphone 14 pro	client@example.com	A valider

At the bottom left of the table area, it says 'page 1-1 lignes 5'. On the right side of the table area are navigation arrows for pagination.

La liste filtrée ne change pas d'aspect, il n'est donc pas nécessaire de l'afficher.

## Maquette détail devis

The screenshot shows a web application interface for a quotation system. The top navigation bar includes a user profile for 'Moh Admin'. The left sidebar menu has 'Devis' selected. The main content area displays a client detail card for 'DEV0001' and a repair information card for an iPhone 15 screen replacement.

**CLIENT**

Nom	Doe	Date	05/02/24 15h30
Prénom	John	Status	En cours
Email	john.doe@example.com	Responsable	Moh

**REPARATION**

Téléphone	iphone 15 - version A1684
Panne	Changement écran
Observation	Le client souhaite ajouter un verre trempé
Pièces nécessaire	écran iphone 15 - n°12457
Date de restitution estimé	15 fév 2024 - 15h30
Montant	165.99 €

**Pièces disponible ?**

**Logout**

J'affiche sur cette page :

- les informations du client essentiel
- Les identifiants du devis
- la détail technique sur la réparation et la pièce nécessaire

Ces pages sont intégrées dans l'application.

## Vue page devis desktop

Devis

Filter par date: January 27 2024 - February 25 2024, Par statut, Appliquer, Refresh

ID	Num devis	Prix TTC	Date création	Date restitution estimé	Appareil	Contact	Status	Action
1	DEV001	150.000€	24/02/2024 16:39	15/02/2024 00:00	iPhone 11 Pro	jane.doe@example.com	En attente	
2	DEV002	80.000€	24/02/2024 16:39	15/02/2024 00:00	iPhone 11 Pro	jane.doe@example.com	En attente	
3	DEV003	75.000€	24/02/2024 16:39	20/02/2024 00:00	iPhone 11 Pro	jane.doe@example.com	En attente	
4	DEV004	100.000€	24/02/2024 16:39	25/02/2024 00:00	iPhone 11 Pro	john.smith@example.com	En attente	

4 devis

Déconnexion

## Tablette

Devis

Filter par date: January 27 2024 - February 25 2024, Par statut, Appliquer, Refresh

ID	Num devis	Prix TTC	Date création	Date restitution estimé	Appareil	Contact	Status	Action
1	DEV001	150.000€	24/02/2024 16:39	15/02/2024 00:00	iPhone 11 Pro	jane.doe@example.com	En attente	
2	DEV002	80.000€	24/02/2024 16:39	15/02/2024 00:00	iPhone 11 Pro	jane.doe@example.com	En attente	
3	DEV003	75.000€	24/02/2024 16:39	20/02/2024 00:00	iPhone 11 Pro	jane.doe@example.com	En attente	
4	DEV004	100.000€	24/02/2024 16:39	25/02/2024 00:00	iPhone 11 Pro	john.smith@example.com	En attente	

4 devis

## Mobile

The screenshot shows a mobile application interface for managing quotes (Devis). At the top, there is a header bar with a back arrow, a search icon, and a refresh icon. The URL in the address bar is <http://127.0.0.1:8000/admin/devis/>. On the right side of the header is a user profile icon labeled "DU Dupont Admin". Below the header, the word "Devis" is displayed in bold black text.

On the left, there is a sidebar menu represented by three horizontal lines. The main content area has a dark header with a white "Y" icon. Below it, there is a filter section titled "Filtrer par date" with a date range selector set from "January 27 2024 - February 25 2024". There are also buttons for "Par statut" and "Appliquer". A red circular loading icon is visible at the bottom of this section.

The main table displays two rows of quote information:

ID	
Num devis	DEV001
Prix TTC	150.000€
Date création	24/02/2024 16:39
Date restitution estimé	15/02/2024 00:00
Appareil	iPhone 11 Pro
Contact	jane.doe@example.com
Status	En attente
Action	👁️

ID	
Num devis	DEV002
Prix TTC	80.000€

Vue page filtrer date devis

**Devis**

Filter par date January 27 2024 - February 25 2024 Par statut Appliquer ⟳

ID	Nom	Création	Statut	Action
1	D	2024-01-27	En attente	<span style="color: #ccc;">O</span>
2	D	2024-01-27	En attente	<span style="color: #ccc;">O</span>
3	D	2024-01-27	En attente	<span style="color: #ccc;">O</span>
4	D	2024-01-27	En attente	<span style="color: #ccc;">O</span>
4 devis				

Today  
Yesterday  
Last 7 Days  
**Last 30 Days**  
This Month  
Last Month  
Custom Range

Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
31	1	2	3	4	5	6	28	29	30	31	1	2	3
7	8	9	10	11	12	13	4	5	6	7	8	9	10
14	15	16	17	18	19	20	11	12	13	14	15	16	17
21	22	23	24	25	26	27	18	19	20	21	22	23	24
28	29	30	31	1	2	3	25	26	27	28	29	1	2
4	5	6	7	8	9	10	3	4	5	6	7	8	9

01/27/2024 - 02/25/2024 Cancel Apply

Vue page détail devis desktop

**{Dev}\_nx** DU Dupont Admin

Devis n°1 Retour liste devis

- Dashboard
- Devis
- Pièces
- Inbox
- Marque
- Modèle

PAGES

- Client
- Team


Devis n°1 Retour liste devis

IPHONE 11 PRO

Stock: 50 pièces

**CLIENT**

Nom	Doe
Prénom	Jane
Email	jane.doe@example.com

**DEV001**

Date	2024-02-24 16:39
Status	En attente
Responsable	Dupont Jean

**REPARATION**

Appareil	iPhone 11 Pro
Observation	Ecran cassé
Pièces nécessaires	Ecran
Date de restitution estimée	2024-02-15 00:00
Montant	150.000€

Déconnexion

Tablette

☰ DU Dupont Admin ⚙

Devis n°1 ← Retour liste devis

 IPHONE 11 PRO

**CLIENT**

Nom	Doe
Prénom	Jane
Email	jane.doe@example.com

**DEV001**

Date	2024-02-24 16:39
Status	En attente
Responsable	Dupont Jean

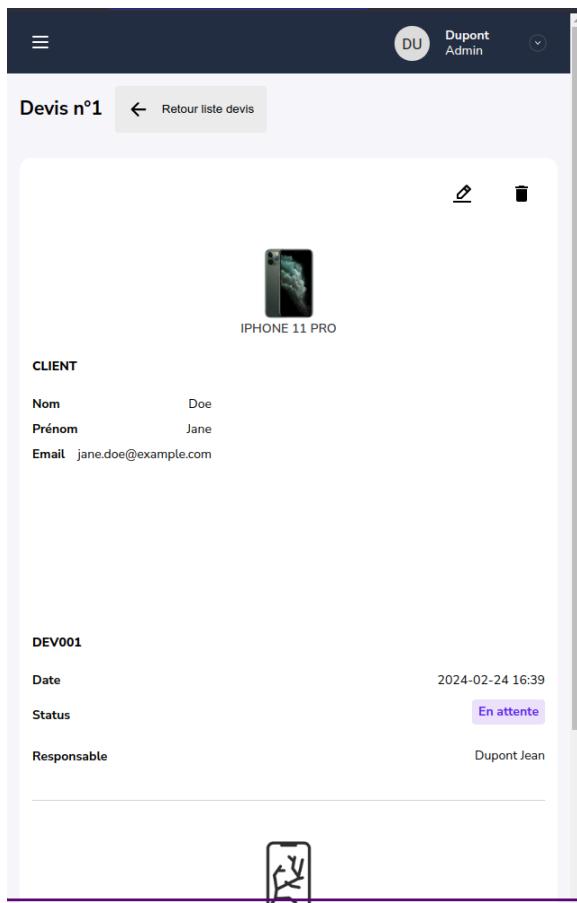


Stock: 50 pièces

**REPARATION**

Appareil	iPhone 11 Pro
Observation	Ecran casse
Pièces nécessaire	Ecran
Date de restitution estimé	2024-02-15 00:00
Montant	150.000€

Mobile



Vue page modifier un devis

The screenshot shows the same "Devis n°1" page as above, but with a sidebar on the left containing navigation links: Dashboard, Devis, Pièces, Inbox, Marque, and Modèle. Below the sidebar, there are sections for "PAGES" (Client, Team) and "Déconnexion". A signature icon is located at the bottom right of the main content area.

Voyons le code plus en détail pour expliquer ce que j'ai mis en place sur ses différentes pages.

Code page devis html Twig

```

admin_devis.html.twig
You, 2 days ago | author (You)
54  {% extends 'base_admin.html.twig' %}
55
56  {% block title %}Devis
57  {% endblock %}
58
59
60  {% block stylesheets %}
61    {{ parent() }}
62    <link rel="stylesheet" href="{{ asset('admin/css/datepicker.css') }}" type="text/css">
63    <link rel="stylesheet" href="{{ asset('admin/css/devis.css') }}" type="text/css">
64  {% endblock %}
65
66  {% block javascripts %}
67    {{ parent() }}
68    <script src="https://cdn.jsdelivr.net/jquery/latest/jquery.min.js" defer></script>
69    <script src="https://cdn.jsdelivr.net/momentjs/latest/moment.min.js" defer></script>
70    <script src="https://cdn.jsdelivr.net/npm/daterangepicker/daterangepicker.min.js" defer></script>
71    <script src="{{ asset('admin/js/dateFilter.js') }}" defer></script>
72  {% endblock %}
73
74
75  {% block body %}
76
77    <main class="main">
78      <section class="section_hero">
79        <div class="title-container">
80          <h1 class="title">Devis</h1>
81        </div>
82
83        <div class="element-container">
84          <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="lucide-lucide-filter">
85            <polygon points="22 3 2 3 10 12.46 10 19 14 21 14 12.46 22 3" />
86          </svg>
87          <div class="filter_container date">
88            <div class="label">Filtrer par date</div>
89            <div id="reportrange">
90              <svg class="lucide-calendar-search" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">...</svg>
91              <span></span>
92              <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="lucide-lucide-chevron-down">
93                <path d="m6 9 6 6 6-6" />
94              </svg>
95            </div>
96          </div>
97        </div>
98      </section>
99    </main>
100
101
```

J'importe la base admin dans ce template, j'ajoute des fichiers css et javascript personnalisé afin de faciliter les futures modifications du code.

J'utilise la fonction “parent” qui permet d'utiliser les mêmes fichiers cités dans la base Twig écrite en première ligne grâce à “extends”.

Permettre de filtrer les devis étant essentiel pour l'administrateur, je l'ai implémenté grâce à une librairie javascript “daterange picker”, la possibilité de sélectionner un créneau de date, des raccourcis pour sélectionner le mois courant, le mois dernier, la semaine dernière ou du jour.

Cette librairie est dépendante de Jquery que j'ai également utiliser grâce à un lien CDN<sup>2</sup>.

#### Code filtre devis

<sup>2</sup> un système de serveurs distribués géographiquement qui collaborent pour fournir du contenu Internet, notamment des pages web, des images, des vidéos et d'autres fichiers multimédias, de manière efficace et rapide aux utilisateurs finaux.

```
<main class="main">
    <section class="section_hero">
        <div class="title-container">
            </div>

            <div class="element-container">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none"
                    stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="lucide-lucide-filter">
                    <polygon points="22 3 2 3 10 12.46 10 19 14 21 14 12.46 22 3"/>
                </svg>
                <div class="filter_container date">
                    <div class="label">Filtrer par date</div>
                    <div id="reportrange">
                        <span><svg class="lucide-calendar-search" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">...</svg></span>
                        <span><svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="lucide-lucide-chevron-down"><path d="M6 9 6 6 6-6"/></svg></span>
                    </div>
                </div>

                </div>
                <form id="formDatePicker" action="{{ path('admin_devis_filter') }}" method="post">
                    <input hidden="true" id="startDate" name="startDate" value="">
                    <input hidden="true" id="endDate" name="endDate" value="">
                    <div class="filter_container status">
                        <select class="select statut-select" name="statut-filter" id="statut-filter">
                            <option value="">Par statut</option>
                            <option value="1">Valider</option>
                            <option value="0">En attente</option>
                        </select>
                    </div>
                    <button class="apply" type="submit">Appliquer</button>
                </form>
                <div class="filter_container reset">
                    <a href="{{ path('admin_devis') }}>
                        <button class="btn-reset-filter" id="btn-reset-filter">
                            <span><svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 32 32">
                                <path fill="currentColor" d="M18 28A12 12 0 1 0 6 16v6L2-3.6-3.6L1 20L6 6L-1.4-1.4L8 22.2V16a10 10 0 1 1 10 10Z"/>
                            </span>
                        </button>
                    </a>
                </div>
            </div>
        </section>
```

J'utilise un formulaire avec la méthode post qui permet d'envoyer les données remplies dans les inputs et select.

L'action envoie la requête au contrôleur saisie dans la fonction "path", il est également possible d'envoyer des paramètres dans le path. J'ai utilisé cette méthode dans un autre formulaire.

Je masque les inputs dans le formulaire, car c'est la librairie qui me permet de sélectionner deux dates, une de départ et une de fin, je récupère les valeurs issues de datepicker pour les assignés comme valeurs aux inputs correspondants.

*Code javascript datepicker*

```
You, 2 days ago | 1 author (You)
41 $(function () {
40   var start = moment().subtract(29, 'days');
39   var end = moment();
38
37   function cb(start, end) {
36     $('#reportrange span').html(start.format('MMMM D YYYY') + ' - ' + end.format('MMMM D YYYY'));
35   }
34
33   $('#reportrange').daterangepicker({
32     startDate: start,
31     endDate: end,
30     ranges: {
29       'Today': [moment(), moment()],
28       'Yesterday': [moment().subtract(1, 'days'), moment().subtract(1, 'days')],
27       'Last 7 Days': [moment().subtract(6, 'days'), moment()],
26       'Last 30 Days': [moment().subtract(29, 'days'), moment()],
25       'This Month': [moment().startOf('month'), moment().endOf('month')],
24       'Last Month': [moment().subtract(1, 'month').startOf('month'), moment().subtract(1,
23         'month').endOf('month')]
22     },
21   }, cb);
20
19   cb(start, end);
18
17   $('#reportrange').on('apply.daterangepicker', function(e, picker) {
16     e.preventDefault();
15     var startDate = picker.startDate.format('YYYY-MM-DD');
14
13     $('#startDate').val(startDate);
12     $('#endDate').val(endDate);
11   });
10
9
8   $('#btn-reset-filter').on('click',function() {
7     $('#statut-filter').val('');
6     $('#startDate').val('');
5     $('#endDate').val('');
4     cb(start, end);
3   });
2 });
1
```

Lors de validation de la sélection de la date range, j'ajoute les formatés dans les inputs modifiers.

J'ajoute un écouteur d'événement qui reset les valeurs aux seins des deux filtres, je renvoie.

J'affiche la liste des devis en utilisant une boucle dans Twig.

*Code liste devis Twig*

```

1 // admin_devis.html.twig ---> static/admin/app/dev/admin_devis.html.twig
2
3 5  {% block body %}
4
5    <main class="main">
6      <section class="section_list">
7        <table class="tab">
8          <thead>
9            <tr>
10              <th>Date création</th>
11              <th>Date restitution estimé</th>
12              <th>Appareil</th>
13              <th>Contact</th>
14              <th>Status</th>
15              <th>Action</th>
16          </tr>
17        </thead>
18        <tbody>
19          {% for item in devis %}
20            <tr>
21              <td data-label="ID" class="id-devis">{{ item.idDevis }}</td>
22              <td data-label="Num devis" class="num-devis">{{ item.numDevis }}</td>
23              <td data-label="Prix TTC" class="prix-devis">
24                <span>{{ item.prixTtc }}</span>
25              </td>
26              <td data-label="Date création" class="date-creation-devis">{{ item.dateDevis.format('d/m/Y H:i') }}</td>
27              <td data-label="Date restitution estimé" class="date-restitution-devis">{{ item.dateRestitution.format('d/m/Y H:i') }}</td>
28              <td data-label="Appareil" class="appareil-devis">{{ item.idReparation.idAppareil.idModele.getLibModele() }}</td>
29              <td data-label="Contact" class="contact-devis">{{ item.idReparation.idUtilisateur.email }}</td>
30              <td data-label="Status" class="status-devis">
31                {% if item.statut == 0 %}
32                  <span class="status valider">En attente</span>
33                {% elseif item.statut == 1 %}
34                  <span class="status valider">Valider</span>
35                {% endif %}
36              </td>
37              <td data-label="Action">
38                <a href="{{ path('admin_devis_detail', { 'idDevis': item.idDevis}) }}">
39                  <button class="show-devis">
40                    <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">...</svg>
41                  </button>
42                </a>
43              </td>
44            </tr>
45          {% endfor %}
46        </tbody>
47      </table>
48    </section>
49    <div class="info-container">
50      <div class="infoLine-container">
51        {{ nb_devis }}
52        devis
53      </div>
54      <div class="container-navigation"></div>
55    </div>

```

Cette fonctionnalité est très pratique pour insérer des valeurs issues d'un tableau en provenance du contrôleur fournissant cette page.

Je cartographie les données dans les colonnes correspondantes. Le bouton inclus dans chaque ligne me permet de renvoyer vers le détail du devis correspondant, j'utilise pour ceci un lien avec la fonction "path" qui renvoie vers une route avec le fameux paramètre cité précédemment.

Vous avez pu constater que peu de code javascript est nécessaire lorsqu'on utilise Twig, c'est l'un des avantages à utiliser un moteur de template.

Le code du devis filtré ne change que très peu, je récupère simplement les anciennes valeurs sélectionnées dans le formulaire pour les remettre en tant que valeurs des inputs et du select.

### Code page devis détails

```

33  {%- block body %} 
34  <main class="main">
35      <section class="section_hero">
36          <div class="title-container">
37              <h1 class="title">Devis n°<span>{{ devis.IdDevis }}</span></h1>
38              <a href="{{path('admin_devis')}}" class="link-button link-back-button">
39                  <button class="back">
40                      <span>Retour liste devis</span>
41                      
42                          <path fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="4" d="M244 400L108 256L144-144M120 256h292" />
43                  </button>
44              </a>
45          </div>
46          <div class="container">
47              <div class="container-devis">
48                  <div class="cta_container">
49                      <button class="edit">...
50                      | You, 2 days ago via PR #1 · feat: devis page and edit devis finish & filter d...
51                      <button class="remove">...
52                  </button>
53              </div>
54              <div class="devis_container">
55                  <div class="info_container">
56                      <div class="wrapper">
57                          <div class="client">
58                              <div class="appareil_container">
59                                  
60                                  <span class="modele">
61                                      {{ devis.idReparation.idAppareil.idModele.getLibModele}}
62                                  </span>
63                              </div>
64                              <div class="client_container">
65                                  <div class="client_title">CLIENT</div>
66                                  <div class="client_container-info">
67                                      <div class="container-label">
68                                          <div class="label-devis">Nom</div>
69                                          <div class="label-devis">Prénom</div>
70                                          <div class="label-devis">Email</div>
71                                      </div>
72                                      <div class="container-content">
73                                          <div class="client_name">{{ devis.idReparation.idUtilisateur.nom }}</div>
74                                          <div class="client_prenom">{{ devis.idReparation.idUtilisateur.prenom }}</div>
75                                          <div class="client_email">{{ devis.idReparation.idUtilisateur.email }}</div>
76                                      </div>
77                                  </div>
78                              </div>
79                          </div>
80                      </div>
81                  </div>
82              </div>
83          </div>
84      </section>
85  </main>

```

Je "map" également les données issues de la route dédiée. Ce code me permet d'afficher les détails d'un devis selon son identifiant unique.

Pour modifier le devis, j'inclus un formulaire dynamique qui ne s'affiche que lors du clique sur le bouton modifier le devis;

Je masque les select que je n'affiche qu'à partir de l'événement, le javascript ne crée pas d'élément, j'ai voulu éviter trop d'interaction entre Twig et javascript, un problème de synchronisme aurait pu générer des comportements indésirables.

J'utilise à mon avantage les possibilités de masqués, changer des valeurs css depuis le script javascript.

*Code modifier le devis*

```
<div class="container-devis">
    <div class="devis">
        <div class="devis__ID">{{ devis.numDevis }}</div>
        <div class="devis__container-info">
            <div class="container-label">
                <div class="label-devis">Date</div>
                <div class="label-devis label-status">Status</div>
                <div class="label-devis label-responsable">Responsable</div>
            </div>
            <form method="post" action="{{ path('admin_devis_update', {'idDevis': devis.idDevis }) }}>
                <div class="container-content">
                    <div class="devis__date">{{ devis.dateDevis.format('Y-m-d H:i') }}</div>
                    <div class="devis__status">
                        <select hidden name="status-devis" id="status-devis">
                            <option value=""></option>
                            <option value="1" {% if devis.statut == '1' %}selected{% endif %}>Terminer</option>
                            <option value="0" {% if devis.statut == '0' %}selected{% endif %}>En attente</option>
                        </select>
                        {% if devis.statut == '0' %}
                            <span id="edit-statut" class="status avalider">En attente</span>
                        {% elseif devis.statut == 1 %}
                            <span id="edit-statut" class="status valider">Valider</span>
                        {% endif %}
                    </div>
                    <div class="devis__responsible">
                        <span id="responsable-name">{{devis.idUtilisateur.nom ? devis.idUtilisateur.nom ~ ' ' ~ devis.idUtilisateur.prenom : 'none'}}</span>
                        <select hidden name="responsable" id="responsable">
                            {% for resp in admin %}
                                <option value="{{ resp.nom }}>{{ resp.nom ~ ' ' ~ resp.prenom }}</option>
                            {% endfor %}
                        </select>
                    </div>
                    <div>
                        <button hidden type="submit" class="btn-valider">
                            <img alt="Submit button icon" data-bbox="100 100 150 150" />
                        </button>
                    </div>
                </form>
            </div>
        </div>
    </div>
<div class="separator"></div>
```

## *Code javascript gestion input*

```
devis.js assets/admin/js/devis.js/...
You, 2 days ago | 1 author (You)
20 document.querySelector('.edit').addEventListener('click', editDevis)
19
18 const valid = document.querySelector('.btn-valider');
17 const selected = document.querySelector('#status-devis');
16 const statut = document.querySelector('#edit-statut');
15 const responsableInput = document.querySelector('#responsable');
14 const respName = document.querySelector('#responsable-name');
13
12
11
10 function editDevis(e) {
9   e.target.hidden = true;
8   selected.hidden = false;
7   statut.style.display = "none";
6   respName.style.display = "none";
5   responsableInput.hidden = false;
4   document.querySelector('.label-responsable').style.marginTop = "2rem";
3   valid.hidden = false;
2 }
1
```

Lorsque la modification est validée, la page se recharge, il n'y a donc pas besoin d'ajouter de code javascript pour réinitialiser les valeurs changées.

Passons à l'étude de la partie back-end de l'application

## <h2>Le back-end</h2>

En partant du cahier des charges, nous proposons un modèle conceptuel de données contenant 15 entités et 19 associations représentant les données nécessaires à mettre en œuvre au sein de l'application.

Nous avons produit 14 versions du MCD avant d'obtenir un résultat probant, le but était de réaliser un modèle qui garantit l'intégrité et la normalisation de nos données tout en répondant aux besoins fonctionnels et opérationnels de l'application, ainsi qu'aux exigences spécifiques de notre entreprise et de nos utilisateurs.

Un MCD qui est à l'origine bien pensé assure d'emblé une bonne compréhension des besoins et clarifie le modèle auprès des différents acteurs participant au processus de développement. Cela contribue également à réduire les erreurs pendant le développement et assure une solidité à la base de données.

Les entités principales de notre application sont "utilisateur", "devis", "réparation" et appareil.

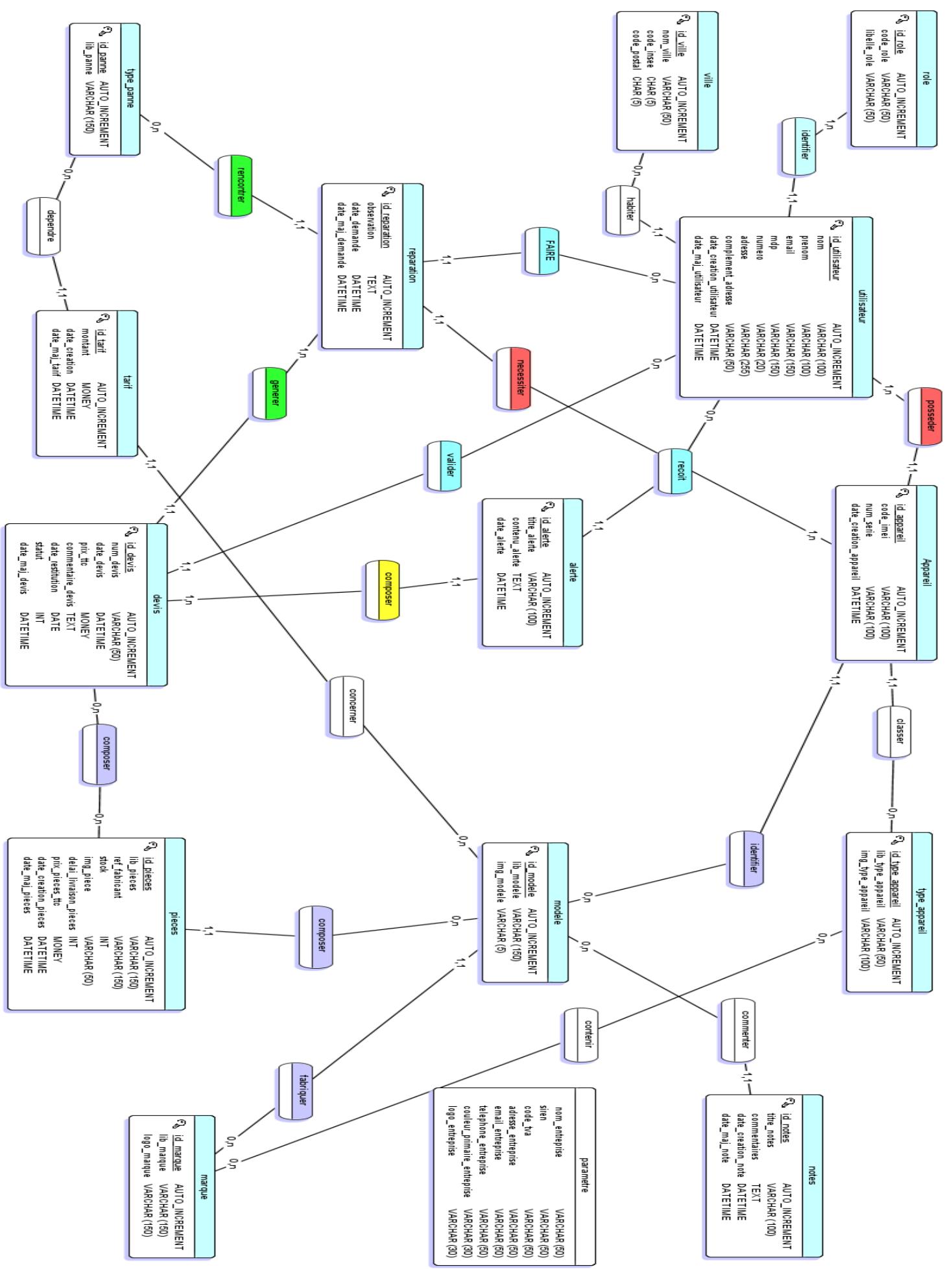
Pour nous aider à réfléchir sur les différentes interactions entre elles, nous avons composé des phrases qui ont du sens. Ex : un client possède un appareil et demande une réparation, le devis doit contenir la réparation, le client et la pièce nécessaire.

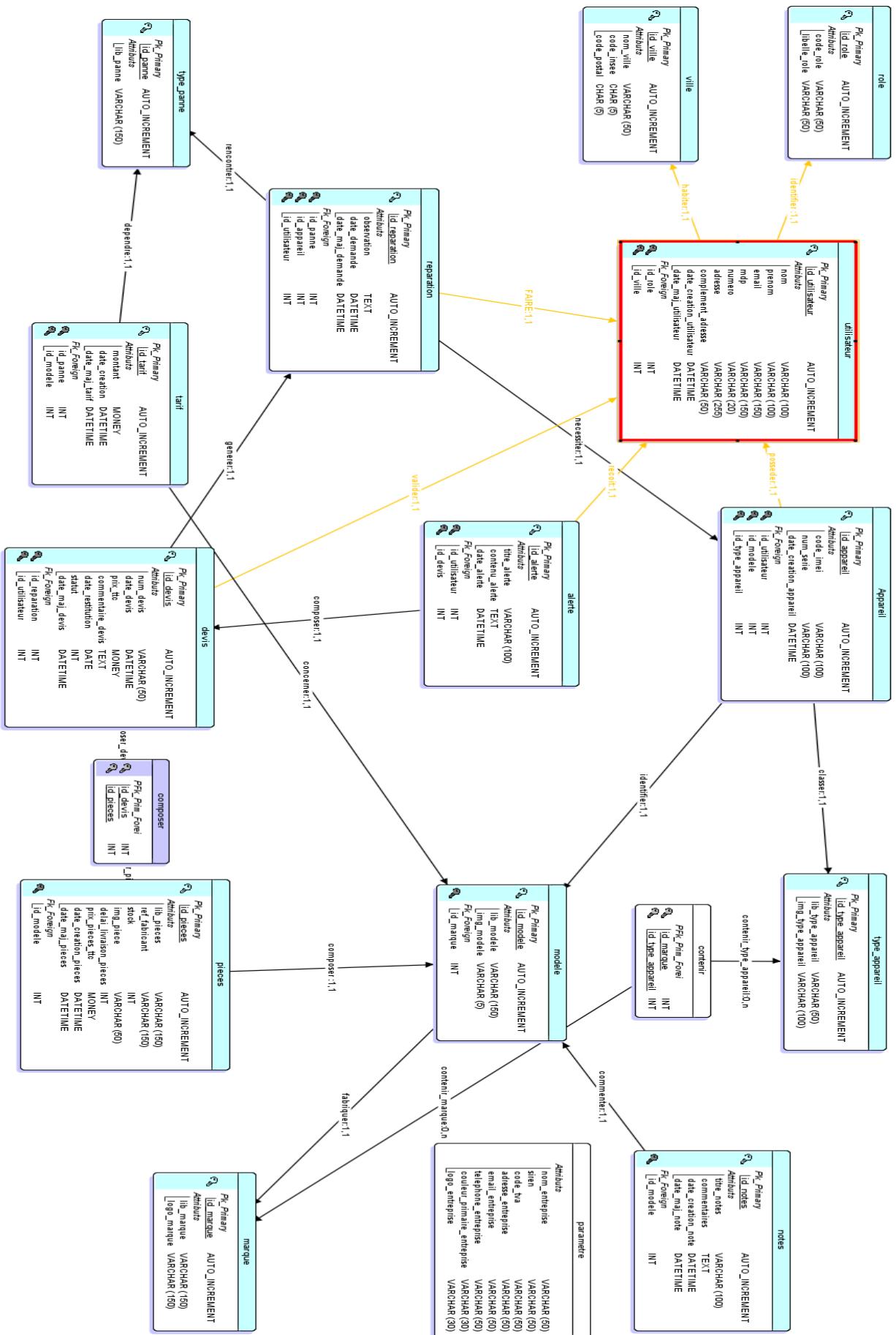
Les deux types d'association courantes sont présentes dans notre MCD, par exemple entre devis et pièces il y a une association maillé (non hiérarchique) avec des cardinalités maximales [n,n] de chaque côté. Une association hiérarchique entre rôle et utilisateur [n,1].

Nous avons utilisés l'outil JMerise pour modéliser le MCD, celui-ci vérifie le modèle avant de le transformer en modèle logique de données (MLD ou MPD), en effet la transformation respect des règles prédefinies :

- Association non hiérarchique = 3eme table avec migrations des clés primaires en tant que clé étrangère, l'union des deux formes la clé primaire de cette table
- Association hiérarchique = migration de clé primaire côté de la cardinalité maximale 1 en tant que clé étrangère

Voici le schéma du modèle conceptuel de données que nous proposons ci dessus:





JMerise génère également le script SQL de base, néanmoins j'ai externalisé la création de contrainte en fin de script SQL, cela permet une plus grande modularité à la base de données. L'ajout de nouvelle table sans se soucier de l'ordre de création fait gagné beaucoup de temps.

### Script SQL schema database

```
11  DROP DATABASE IF EXISTS `devfix`;
12  CREATE DATABASE IF NOT EXISTS `devfix`;
13
14  use `devfix`;
15
16
17  CREATE TABLE
18      type_panne (
19          id_panne INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
20          lib_panne VARCHAR(150) NOT NULL,
21          img_panne VARCHAR(40)
22      ) ENGINE = InnoDB;      You, last week via PR ... * fix: sql script new folder sql in
23
24  CREATE TABLE
25      type_appareil (
26          id_type_appareil INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
27          lib_type_appareil VARCHAR(50) NOT NULL,
28          img_type_appareil VARCHAR(100)
29      ) ENGINE = InnoDB;
30
31  CREATE TABLE
32      marque (
33          id_marque INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
34          lib_marque VARCHAR(150) NOT NULL,
35          logo_marque VARCHAR(150)
36      ) ENGINE = InnoDB;
37
38  CREATE TABLE
39      contenir (
40          id_type_appareil INT NOT NULL,
41          id_marque INT NOT NULL
42      ) ENGINE = InnoDB;
43
44  CREATE TABLE
45      modele (
46          id_modele INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
47          lib_modele VARCHAR(150) NOT NULL,
48          img_modele VARCHAR(40),
49          id_marque INT NOT NULL
50      ) ENGINE = InnoDB;
51
52  CREATE TABLE
53      pieces (
54          id_pieces INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
55          lib_pieces VARCHAR(150) NOT NULL,
56          ref_fabricant VARCHAR(150) NOT NULL,
57          stock INT NOT NULL,
58          img_piece VARCHAR(50) NOT NULL,
59          delai_livraison_pieces INT NOT NULL,
60          prix_pieces_ttc DECIMAL(15, 3) NOT NULL,
61          date_creation_pieces DATETIME NOT NULL
62      ) ENGINE = InnoDB;
```

En mode développement, je supprime la base de données en première ligne et je l'ai recréée juste après, cela me permet de réinitialiser la base sans erreur de contrainte.

Je définit seulement les clés primaires dans la création des tables ensuite en fin de script, je modifie ses tables en ajoutant des contraintes.

## *Script SQL constraints*

Ensuite, on insère des données tests afin de pouvoir vérifier les résultats attendus dans l'application.

```
use `devfix`;

INSERT INTO
    type_panne (lib_panne, img_panne)
VALUES
    ('Ecran casse', 'ecran.png'),
    ('Dommages par l''eau', 'dommages_eau.png'),
    ('Batterie', 'batterie.png'),
    ('Probleme de charge', 'chargement.png'),
    ('Deblocage / Logiciel', 'deblocage.png'),
    ('Camera defectueuse', 'camera.png'),
    ('Dommages a l''arriere', 'arriere.png');

INSERT INTO
    role (code_role, libelle_role)
VALUES
    ('admin', 'Administrateur'),
    ('technicien', 'Technicien'),
    ('client', 'Client');
```

Il faut respecter un certain ordre d'insertion, car certaines nécessitent des clés étrangères et donc doivent préexister.

La solution serait peut être de séparer le script en deux parties et d'insérer les tables sans contraintes de clés étrangère, puis insérez le reste par la suite.

Une fois le processus de conception finalisé, j'ai dockerisé ma base de données (BDD) grâce à un docker compose au format yaml, je configure ma BDD lors de l'exécution, j'ajoute un volume pour la persistance des données entre les différentes exécutions du container.

```
version: '3.8'
services:
  database:
    image: mysql:latest
    container_name: db_mysql
    restart: always
    command: --default-authentication-plugin=caching_sha2_password --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_PASSWORD:-secret}
      MYSQL_DATABASE: ${DB_NAME:-devfix}
    volumes:
      - ./scriptSQL/sql/01_create.sql:/docker-entrypoint-initdb.d/01_create.sql
      - ./scriptSQL/sql/02_init.sql:/docker-entrypoint-initdb.d/02_init.sql
      - mysql_data:/var/lib/mysql
    ports:
      - 3306:3306
    networks:
      my_network:
        ipv4_address: 172.20.0.3
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin
    restart: always
    ports:
      - 7070:80
    environment:
      PMA_HOST: database
    networks:
      my_network:
    volumes:
      mysql_data:
    networks:
      my_network:
        driver: bridge
        ipam:
          config:
            - subnet: 172.20.0.0/16
```

J'ajoute une adresse ip fixe afin d'accéder facilement à ma BDD Mysql

Je copie les deux scripts SQL qui s'exécute dans le container automatiquement, le nom et le mot de passe root peuvent être définis dans les variables d'environnements, le docker est facilement configurable et devrait permettre un environnement de développement stable et cohérent entre développeurs.

J'ai réalisé cette User Stories : Consulter les devis

**En tant que administrateur** de l'application,

**Je veux** pouvoir consulter la liste des devis créés par les clients, filtrer les devis par statut et par date, visualiser les détails de chaque devis, et avoir la possibilité de modifier le statut ou le responsable des devis,

**Afin de** gérer efficacement les demandes de devis et assurer un suivi approprié avec les clients.

Critères d'acceptation :

Dans l'espace administrateur, je dois avoir accès à une section dédiée pour consulter la liste des devis.

Je dois pouvoir filtrer les devis par statut (par exemple : en attente, accepté, refusé, en cours, terminé) pour mieux organiser les demandes.

Je dois pouvoir filtrer les devis par date de création pour retrouver facilement les demandes récentes ou anciennes.

Chaque devis dans la liste doit afficher les informations essentielles telles que le numéro de devis, le client associé, la date de création, le statut actuel, et le responsable actuel.

Je dois pouvoir accéder aux détails complets de chaque devis en cliquant sur celui-ci dans la liste.

Dans les détails du devis, je dois avoir la possibilité de modifier le statut (par exemple : en attente, accepté, refusé, en cours, terminé) et le responsable attribué au devis.

Scénario :

Étape 1 : En tant qu'administrateur, je me connecte à l'espace administrateur de l'application.

Étape 2 : Je navigue vers la section devis.

Étape 3 : J'applique des filtres pour afficher les devis selon le statut et/ou la date.

Étape 4 : Je consulte la liste des devis avec les informations de base affichées.

Étape 5 : Je clique sur un devis spécifique pour voir ses détails.

Étape 6 : Dans les détails du devis, je modifie le statut et/ou le responsable si nécessaire.

Tests à effectuer :

Vérifier que l'administrateur peut accéder à la section des devis dans l'espace administrateur.

Vérifier que les filtres de statut et de date fonctionnent correctement et affichent les devis correspondants.

Vérifier que les informations de base des devis sont correctement affichées dans la liste.

Vérifier que l'administrateur peut accéder aux détails complets de chaque devis.

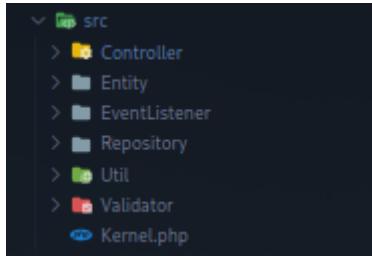
Vérifier que l'administrateur peut modifier le statut et/ou le responsable d'un devis avec succès.

A partir de la base de données, Symfony via Doctrine (ORM<sup>3</sup>) permet de générer les entités avec leur repository correspondant.

Coté back-end, l'essentiel au sein de Symfony se passe dans ce dossier :

---

<sup>3</sup> L'ORM (Object-Relational Mapping) est un outil de programmation utilisé dans le développement de logiciels pour faciliter la manipulation et l'interaction avec une base de données relationnelle. L'objectif principal d'un ORM est de permettre aux développeurs de travailler avec des objets dans leur langage de programmation plutôt que d'écrire directement des requêtes SQL.



Je vais me focaliser sur mon User Stories et donc sur les devis.

### Code Entity Devis

```

Devis.php 2 src/Entity/Devis.php...
...
<?php

namespace App\Entity;

use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

/**
 * Devis
 *
 * @ORM\Table(name="devis", indexes={@ORM\Index(name="fk_devis_reparation", columns={"id_reparation"}), @ORM\Index(name="fk_devis_utilisateur", columns={"id_responsable"})})
 * @ORM\Entity(repositoryClass= "App\Repository\DevisRepository")
 */
class Devis
{
    /**
     * @var int
     *
     * @ORM\Column(name="id_devis", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $idDevis;

    /**
     * @var string
     *
     * @ORM\Column(name="num_devis", type="string", length=50, nullable=false)
     */
    private $numDevis;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="date_devis", type="datetime", nullable=false)
     */
    private $dateDevis;

    /**
     * @var string
     *
     * @ORM\Column(name="prix_ttc", type="decimal", precision=15, scale=3, nullable=false)
     */
    private $prixTtc;
}

```

La classe Devis permet de représenter l'objet Devis relier grâce aux annotations Doctrine à la BDD, ici devis. De plus, les attributs sont également “mappés” aux colonnes de la BDD.

Doctrine permet également de “mapper” les relations entre les tables à travers les entités correspondantes.

En utilisant le même principe que la méthode Merise, au sujet des associations, cardinalité, doctrine utilise une syntaxe assez similaire. (ManyToOne ou ManyToMany...)

```

    /**
     * @var \Reparation
     *
     * @ORM\ManyToOne(targetEntity="Reparation")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="id_reparation", referencedColumnName="id_reparation")
     * })
     */
    2 references
    private $idReparation;

    /**
     * @var \Utilisateur
     *
     * @ORM\ManyToOne(targetEntity="Utilisateur")
     * @ORM\JoinColumn(name="id_responsable", referencedColumnName="id_utilisateur")
     */
    2 references
    private $idResponsable;

    /**
     * @var \Doctrine\Common\Collections\Collection
     *
     * @ORM\ManyToMany(targetEntity="Pieces", inversedBy="idDevis")
     * @ORM\JoinTable(name="composer",
     *   joinColumns={
     *     @ORM\JoinColumn(name="id_devis", referencedColumnName="id_devis")
     *   },
     *   inverseJoinColumns={
     *     @ORM\JoinColumn(name="id_pieces", referencedColumnName="id_pieces")
     *   }
     * )
     */
    5 references
    private $idPieces = array();

```

Les attributs sont déclarés en privé pour respecter le principe d'encapsulation que permet la programmation orienté objet.

Le but est de limiter l'accès à ses données et de choisir celles qui pourront être modifiées grâce à des Getters et Setters.

```

0 references | 0 overrides
public function getIdDevis(): ?int
{
    return $this->idDevis;
}

2 references | 0 overrides
public function getNumDevis(): ?string
{
    return $this->numDevis;
}

2 references | 0 overrides
public function setNumDevis(string $numDevis): static
{
    $this->numDevis = $numDevis;

    return $this;
}

0 references | 0 overrides
public function getDateDevis(): ?\DateTimeInterface
{
    return $this->dateDevis;
}

2 references | 0 overrides
public function setDateDevis(\DateTimeInterface $dateDevis): static
{
    $this->dateDevis = $dateDevis;

    return $this;
}

```

Symfony implémente le design pattern MVC, nous avons déjà vu la partie Vue et Model, il nous reste à voir la partie Controller.

Avant ceci, il existe dans Symfony une couche d'abstraction entre le Model (entity) et le Controller : le Repository.

C'est une couche qu'on appelle également DAO (data access object), elle est responsable de la récupération et de la persistance des entités dans la base de données. Elle fournit des méthodes pour effectuer des opérations de lecture, d'écriture, de mise à jour et de suppression sur les entités (CRUD). Ces méthodes peuvent inclure des opérations de recherche complexes, des filtres, des tris, etc.

Cela permet d'encapsuler la logique d'accès aux données dans des classes distinctes, qui favorise la modularité, la réutilisation du code et la maintenabilité de l'application.

### Code DevisRepository

```
DevisRepository.php src\Repository/DevisRepository.php App\Repository\DevisRepository
You, 2 days ago | 2 authors (tozonizi and others)
41 <?php
42
43 namespace App\Repository;
44
45 use App\Entity\Devis;
46 use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
47 use Doctrine\Persistence\ManagerRegistry;
48
49 /**
50 * @extends ServiceEntityRepository<Devis>
51 *
52 * @method Devis|null find($id, $lockMode = null, $lockVersion = null)
53 * @method Devis|null findOneBy(array $criteria, array $orderBy = null)
54 * @method Devis[] findAll()
55 * @method Devis[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
56 */
57
58 4 references | 0 implementations | You, 2 days ago | 2 authors (You and others)
59 class DevisRepository extends ServiceEntityRepository
60 {
61     16 references | 0 overrides
62     public function __construct(ManagerRegistry $registry)
63     {
64         parent::__construct($registry, Devis::class);
65     }
66
67     /**
68      * @param string $statut
69      * @param \DateTime $startDate
70      * @param \DateTime $endDate
71      * @return Devis[]
72     */
73     0 references | 0 overrides
74     public function findByStatutAndDateDevis(string $statut, \DateTime $startDate, \DateTime $endDate): array
75     {
76         return $this->createQueryBuilder('d')
77             ->andWhere('d.statut = :statut')
78             ->andWhere('d.dateDevis BETWEEN :startDate AND :endDate')
79             ->setParameter('statut', $statut)
80             ->setParameter('startDate', $startDate)
81             ->setParameter('endDate', $endDate)
82             ->getQuery()
83             ->getResult();
84     }
85
86     /**
87      * You, 2 days ago + Uncommitted changes
88      * @param \DateTime $startDate
89      * @param \DateTime $endDate
90      * @return Devis[]
91     */
92     0 references | 0 overrides
93     public function findByDateDevis(\DateTime $startDate, \DateTime $endDate): array
94     {
95     }
```

Symfony implémente par défaut les méthodes cités dans la PHPDoc au-dessus de la classe. Le ManagerRegistry permet de récupérer le gestionnaire d'entités afin d'effectuer des opérations de persistance des données dans l'application. Dans le code, le ManagerRegistry est injecté dans le constructeur du DevisRepository, ce qui permet au repository d'interagir avec la base de données via le gestionnaire d'entités associé à l'entité Devis.

J'implémente deux méthodes supplémentaires pour filtrer les devis par date ou/et statut. J'utilise des requêtes DQL<sup>4</sup> spécifique à Doctrine qui permet de "bind" les valeurs issues des paramètres de la méthode. Cela devrait suffire pour parer les injections SQL, connus en termes de failles de sécurité.

J'ajoute d'autres contrôles en amont pour ne pas arriver à cette étape, cela serait contre productive. Il faut intervenir le plus tôt possible pour économiser des ressources au niveau du serveur.

#### Code Devis Contrôleur route 1

```
2 references | 0 implementations | You, 2 days ago | 1 author (You)
class AdminDevisController extends AbstractController
{
    #[Route('/admin/devis', name: 'admin_devis')]
    6 references | 0 overrides
    public function index(EntityManagerInterface $entityManager): Response
    {
        try {
            $user = $this->getUser();
            if (!$user) {
                return $this->redirectToRoute('app_homepage');
            }
            You, 2 days ago via PR ... * feat: devis page and edit devis finish & filter d...
            $role = $user->getRoles();
            if (in_array('ROLE_ADMIN', $role)) {
                $role = 'Admin';
            } elseif (in_array('ROLE_EMPLOYEE', $role)) {
                $role = 'Employe';
            } else {
                return $this->redirectToRoute('app_homepage');
            }

            $devis = $entityManager->getRepository(Devis::class)->findAll();

        } catch (Exception $e) {
            return $this->redirectToRoute('app_error', array('error' => $e->getMessage(), 'code' => $e->getCode()));
        }

        return $this->render('admin_app/devis/admin_devis.html.twig', [
            'controller_name' => 'AdminDevisController',
            'devis' => $devis,
            'nb_devis' => count($devis),
            'role' => $role
        ]);
    }
}
```

Cette route permet d'afficher la liste des devis, le rôle de l'utilisateur connecté.

Si l'utilisateur, soit administrateur ou employé, n'est pas authentifié, il n'aura pas accès à cette route.

De plus, uniquement ses deux types d'utilisateurs ont accès à la partie administrateur.

<sup>4</sup> Le DQL (Doctrine Query Language) est un langage de requête spécifique à Doctrine,

J'utilise l'entityManagerInterface pour accéder aux méthodes du Repository correspondant à l'entity précisé entre parenthèses.

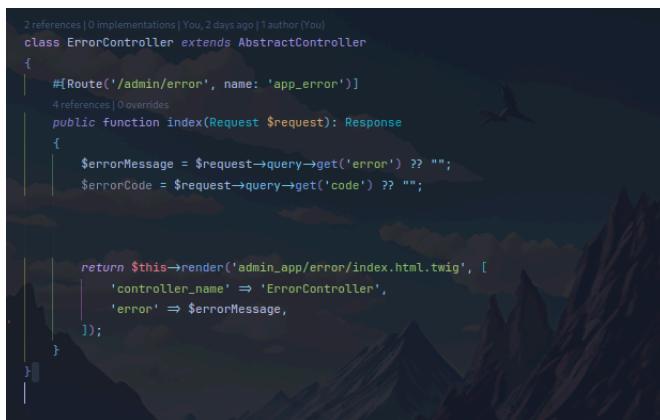
Je peux également accéder directement au repository devis, mais je perdrais une partie des avantages suivants :

- Gestion centralisée des transactions : L'entityManager gère les transactions pour vous. Lorsqu'on appelle une méthode d'un repository via l'entityManager, la méthode est exécutée dans le contexte d'une transaction, ce qui garantit l'atomicité et la cohérence des opérations sur la base de données.
- Gestion de la mise en cache : L'entityManager gère également la mise en cache des entités. Les résultats peuvent être mis en cache pour améliorer les performances de l'application.
- Gestion de l'hydratation des entités : L'entityManager s'occupe de l'hydratation des entités à partir des résultats de la requête. Cela signifie que les entités récupérées sont automatiquement transformées en objets PHP que l'on peut manipuler directement dans votre code.
- Meilleure abstraction et cohérence : l'entityManager permet une meilleure abstraction et cohérence dans votre code. Utiliser un point d'entrée commun pour interagir avec la couche de persistance des données, rend le code plus lisible et maintenable.

Enfin j'utilise "render" pour renvoyer vers le template correspondant avec les valeurs nécessaires vues dans la partie front-end.

Je mets en place un try catch qui renvoie vers la route ci-dessous, avec le message d'erreur dans la page.

Celle-ci récupère l'erreur message ou/et code et l'ajoute en tant que variable au sein du template error.



```
2 references | 0 implementations | You, 2 days ago | 1 author | You
class ErrorController extends AbstractController
{
    #[Route('/admin/error', name: 'app_error')]
    public function index(Request $request): Response
    {
        $errorMessage = $request->query->get('error') ?? "";
        $errorCode = $request->query->get('code') ?? "";

        return $this->render('admin_app/error/index.html.twig', [
            'controller_name' => 'ErrorController',
            'error' => $errorMessage,
        ]);
    }
}
```

Code route 2 devis detail

```

#[Route('admin/devis/{idDevis}', name: 'admin_devis_detail')]
6 references | 0 overrides
public function detail(string $idDevis, EntityManagerInterface $entityManager): Response
{
    $responsable = $this->getUser();
    if (!$responsable) {
        return $this->redirectToRoute('admin_devis');
    }
    try {
        $idDevis = (int) $idDevis;
        $devis = $entityManager->getRepository(Devis::class)->findOneBy(['idDevis' => $idDevis]);
        if (!$devis) {
            return $this->redirectToRoute('admin_devis');
        }

        $admin = $entityManager->getRepository(Utilisateur::class)->findByRole(1);
        $empoye = $entityManager->getRepository(Utilisateur::class)->findByRole(2);

        $resp = [...$admin, ...$empoye];
        if (!$resp) {
            return $this->redirectToRoute('admin_devis');
        }

        $piece = $devis->getIdPieces();
        if (!$piece) {
            return $this->redirectToRoute('admin_devis');
        }

        $role = $responsable->getRoles();

        if (in_array('ROLE_ADMIN', $role)) {
            $role = 'Admin';
        } elseif (in_array('ROLE_EMPLOYEE', $role)) {
            $role = 'Empoye';
        } else {
            return $this->redirectToRoute('app_homepage');
        }

        return $this->render('admin_app/devis/admin_devis_detail.html.twig', [
            'controller_name' => 'AdminDevisController',
            'devis' => $devis,
            'admin' => $resp,
            'responsable' => $responsable,
            'piece' => $piece[0],
            'role' => $role,
        ]);
    } catch (Exception $e) {
        return $this->redirectToRoute('app_error', array('error' => $e->getMessage(), 'code' => $e->getCode()));
    }
}

```

La requête possède un paramètre, le numéro de devis qui ne devrait pas être possible de modifier, néanmoins j'effectue un double contrôle castant le type de l'idDevis et ensuite en utilisant un validator créer pour ceci. En PHP, (int) est utilisé pour forcer la conversion d'une variable en entier.

```
1 references | 0 implementations | You, 2 minutes ago | 1 author (You)
2 class Validator
3 {
4     Reference
5     const STATUS = [0, 1];
6     Reference | 0 overrides
7     public static function isValidateId($id): InvalidArgumentException | bool    You, 2 minutes ago • Uncommitted changes
8     {
9         if (!is_numeric($id) || $id <= 0) {
10             throw new InvalidArgumentException("L'ID doit être un entier positif.");
11         }
12
13         if (!preg_match('/^\d+$/', $id)) {
14             throw new InvalidArgumentException("L'ID doit être une chaîne de chiffres.");
15         }
16         return true;
17     }
18
19     4 references | 0 overrides
20     public static function isValidStatut($statut): InvalidArgumentException | bool
21     {
22
23         if (!in_array($statut, self::STATUS)) {
24             return false;
25         }
26         return true;
27     }
28
29     4 references | 0 overrides
30     public static function isValidTimestamp(int $timestamp): InvalidArgumentException | bool
31     {
32
33         $result = $timestamp > 0 && $timestamp <= PHP_INT_MAX;
34         if(!$result) {
35             throw new InvalidArgumentException('Timestamp invalide.');
36         }
37         return true;
38     }
39 }
```

Ensuite j'utilise l'entityManager pour accéder aux données nécessaires.

### Code route update devis

```
##[Route('/admin/devis/update/{idDevis}', name: 'admin_devis_update')]
6 references | 0 overrides
public function updateDevis(string $idDevis, Request $request, EntityManagerInterface $entityManager): Response
{
    try {
        if ($request->isMethod('POST')) {
            $idDevis = (int) $idDevis;
            Validator::isValidId($idDevis);
            $devis = $entityManager->getRepository(Devis::class)->findOneBy(['idDevis' => $idDevis]);

            $responsable = $request->request->get('responsable');
            $validator = Validation::createValidator();
            $constraints = new Assert\Collection([
                'responsable' => [
                    new Assert\Type(['type' => 'string']),
                    new Assert\Regex([
                        'pattern' => '/<[a-z][\s\S]*>/i',
                        'match' => false,
                        'message' => 'La valeur ne doit pas contenir de balises HTML.'
                    ]),
                    new Assert\Regex([
                        'pattern' => '/[;\'"]/i',
                        'match' => false,
                        'message' => 'La valeur ne doit pas contenir de caractères spécifiques SQL.'
                    ])
                ]
            ]);
            $errors = $validator->validate($responsable, $constraints);

            if (count($errors) > 0) {
                return $this->redirectToRoute('app_error', array('error' => $errors));
            }

            $responsable = $entityManager->getRepository(Utilisateur::class)->findOneBy(['nom' => $request->request->get('responsable')]);

            $newStatus = $request->request->get('status-devis');
            if ($newStatus === '1') {
                $newStatus = 1;
            } else if ($newStatus === '0') {
                $newStatus = 0;
            }

            $devis->setIdUtilisateur($responsable);
            $devis->setStatut($newStatus);
            $entityManager->persist($devis);
            $entityManager->flush();
        }
    }
    return $this->redirectToRoute('admin_devis_detail', ['idDevis' => $idDevis]);
}
```

Je modifie un devis en utilisant son identifiant que je valide préalablement avec de récupérer le devis correspondant. Ensuite j'utilise un validator en utilisant un Assert de type et un Regex pour exclure tout texte contenant de l'html ou du SQL.

Je récupère ensuite le responsable grâce à son nom dans la liste d'utilisateurs. Enfin, je modifie le devis grâce à des setters.

Il serait préférable de déplacer l'écriture en BDD dans la couche repository qui corresponde à sa responsabilité et non ici. J'ai donc effectué la modification ci-dessous.

### Code DevisRepository

```
/*
 * @param Devis $devis
 * @param Utilisateur $responsable
 * @param int $newStatus
 */
0 references | 0 overrides
public function updateDevis(Devis $devis, Utilisateur $responsable, int $newStatus)
{
    $entityManager = $this->getEntityManager();

    $devis->setIdUtilisateur($responsable);
    $devis->setStatut($newStatus);

    $entityManager->persist($devis);      You, 1 second ago • Uncommitted changes
    $entityManager->flush();
}
```

### Code DevisController route update devis

```
$newStatus = $request->request->get('status-devis');
if ($newStatus == '1') {
    $newStatus = 1;
} else if ($newStatus == '0') {
    $newStatus = 0;
}
$entityManager->getRepository(Devis::class)->updateDevis($devis, $responsable, $newStatus);
```

Enfin, j'ai créé une route pour filtrer les devis.

Je contrôle la méthode de la requête http, ensuite je manage les différentes situations grâce à des conditions.

Je convertis également la date en DateTime, la date arrive en tant que string que je transforme en timestamp pour enfin finir en DateTime, qui est le format utilisé dans notre BDD.

## Code route devis filter

```
#Route('/admin/devis-filtered', name: 'admin_devis_filter')
6 references | 0 overrides
public function filtered(Request $request, EntityManagerInterface $entityManager): ?Response
{
    $responsible = $this->getUser();
    if (!$responsible) {
        return $this->redirectToRoute('admin_devis');
    }
    $role = $responsible->getRoles();

    if (in_array('ROLE_ADMIN', $role)) {
        $role = 'Admin';
    } elseif (in_array('ROLE_EMPLOYEE', $role)) {
        $role = 'Employe';
    } else {
        return $this->redirectToRoute('app_homepage');
    }

    if ($request->isMethod('POST')) {
        $status = $request->request->get('statut-filter');
        $status = (int) $status;
        $startDate = strtotime($request->request->get('startDate'));
        $endDate = strtotime($request->request->get('endDate'));

        if (Validator::isValidStatut($status) && $startDate == false && $endDate == false) {
            try {
                You, 2 days ago via PR ... - feat: devis page and edit devis finish & filter d...
                $devis = $entityManager->getRepository(Devis::class)->findBy([
                    'statut' => $status,
                ]);
            } catch (Exception $e) {
                return $this->redirectToRoute('app_error', array('error' => $e->getMessage(), 'code' => $e->getCode()));
            }
        } else if (Validator::isValidStatut($status) && $startDate && $endDate) {
            try {
                Validator::isValidTimestamp($startDate);
                Validator::isValidTimestamp($endDate);
                $endDate_Datetime = DateTimeConverter::convertTimestampToDate($endDate);
                $startDate_Datetime = DateTimeConverter::convertTimestampToDate($startDate);
            } catch (Exception $e) {
                return $this->redirectToRoute('app_error', array('error' => $e->getMessage(), 'code' => $e->getCode()));
            }
        }
    }
}
```

## Code route devis filter

```
    }
} catch (Exception $e) {
    return $this->redirectToRoute('app_error', array('error' => $e->getMessage(), 'code' => $e->getCode()));
}

} else if (Validator::isValidStatut($status) && $startDate && $endDate) {
    try {
        Validator::isValidTimestamp($startDate);
        Validator::isValidTimestamp($endDate);
        $endDate_Datetime = DateTimeConverter::convertTimestampToDate($endDate);
        $startDate_Datetime = DateTimeConverter::convertTimestampToDate($startDate);
        $devis = $entityManager->getRepository(Devis::class)->findByDateDevis($startDate_Datetime, $endDate_Datetime);

        return $this->render('admin_app/devis/admin_devis_filter.html.twig', [
            'controller_name' => 'AdminDevisController',
            'devis' => $devis,
            'nb_devis' => count($devis),
            'role' => $role,
            'statut' => $status,
            'startDate' => $startDate,
            'endDate' => $endDate,
        ]);
    } catch (InvalidArgumentException $e) {
        return $this->redirectToRoute('app_error', array('error' => $e->getMessage(), 'code' => $e->getCode()));
    }
}

} else if(Validator::isValidStatut($status) && $startDate && $endDate){
    try {
        Validator::isValidTimestamp($startDate);
        Validator::isValidTimestamp($endDate);

        $status = (int) $status;
        Validator::isValidStatut($status);

        $endDate_Datetime = DateTimeConverter::convertTimestampToDate($endDate);
        $startDate_Datetime = DateTimeConverter::convertTimestampToDate($startDate);

        $devis = $entityManager->getRepository(Devis::class)->findByStatutAndDateDevis($status, $startDate_Datetime, $endDate_Datetime);

        return $this->render('admin_app/devis/admin_devis_filter.html.twig', [
            'controller_name' => 'AdminDevisController',
            'devis' => $devis,
            'nb_devis' => count($devis),
            'role' => $role,
            'statut' => $status,
            'startDate' => $startDate,
            'endDate' => $endDate,
        ]);
    } catch (InvalidArgumentException $e) {
    }
}
```

Voici les différentes mesures de sécurités mises en place dans notre applications :

- Utilisation de Validator, Regex
- Typage PHP
- Activation de la protection contre les CSRF dans Symfony
- Echappement de l'HTML et SQL
- Utilisation de Doctrine DQL pour les requêtes avec bind des paramètres, éviter les injections SQL
- Préventions contre les attaques XSS
- Accès restreint par rôle
- Encouragement des mots de passe solides

A mettre en place :

- Configurez les serveurs web pour utiliser HTTPS
- Contrôle des fichiers chargés dans les formulaires

- Captcha pour les formulaires accessible sans authentifications
- Clickjacking : Se prémunir d'être inclus dans une iframe d'un autre site en configurant les bonnes en-têtes HTTP.
- Token d'authentification par un serveur tierce

source :

- [https://developer.mozilla.org/fr/docs/Learn/Server-side/First\\_steps/Website\\_security](https://developer.mozilla.org/fr/docs/Learn/Server-side/First_steps/Website_security)
- <https://cheatsheetseries.owasp.org/IndexTopTen.html>

D'autres aspects que l'on n'a pas pris assez en compte dans notre projet.

Conformité aux normes et réglementations :

Respecter les normes et réglementations applicables, telles que le RGPD pour la protection des données personnelles, les normes d'accessibilité WCAG pour les personnes handicapées, etc.

Compatibilité des navigateurs :

Assurer que l'application fonctionne correctement sur une variété de navigateurs Web (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, etc.) ainsi que sur différentes plateformes (Windows, macOS, iOS, Android).

Accessibilité :

Concevoir et développer l'application de manière à ce qu'elle soit accessible à tous les utilisateurs, y compris ceux ayant des besoins spécifiques tels que les personnes handicapées visuelles ou auditives.

Performances de l'application :

Optimiser les performances de l'application pour garantir des temps de chargement rapides, une réactivité fluide et une expérience utilisateur agréable, même dans des conditions de charge élevée.

Test unitaire, fonctionnel :

Chaque test unitaire cible une petite partie du code, comme une fonction ou une méthode, et valide son comportement attendu dans différentes conditions.

Les tests fonctionnels sont souvent effectués du point de vue de l'utilisateur, simulant des interactions réelles avec l'application pour valider son comportement selon divers scénarios d'utilisation.

# <h1>Conclusion</h1>

À ce stade, après plusieurs semaines d'efforts intenses, je me prépare à conclure cette étape cruciale de ma nouvelle vie de développeur avec un sentiment de satisfaction et d'accomplissement.

Dans les prochains jours, j'ai l'intention de prendre quelques jours de repos bien mérités pour me ressourcer et recharger mes batteries. Mais mon parcours ne fait que commencer. Avec une application sur le point d'être finalisée, je regarde vers l'avenir avec optimisme et détermination. J'ai bien l'intention de finaliser ce projet.

J'ai hâte de voir mon travail prendre vie et de le partager avec le monde. Cette expérience m'a permis non seulement de contribuer à un projet passionnant, mais aussi de découvrir ma véritable passion et de trouver ma voie dans le domaine du développement web.

Je suis reconnaissant pour cette opportunité et je suis enthousiaste à l'idée de ce que l'avenir me réserve.

Je suis prêt à relever les défis à venir et à continuer à progresser vers de nouveaux sommets.

Merci à tous ceux qui ont été impliqués dans cette aventure.

L'avenir m'attend, et je suis prêt à le saisir avec confiance et détermination.