

# Dossier Projet

**RNCP 37873 Conceptrice développeuse d'applications**

**Septembre 2024**



## Sommaire :

1. Présentation .....	5
1.1. Présentation personnelle.....	5
1.2. Le Projet .....	5
1.3. The Project .....	6
1.4. Listes de compétences du référentiel .....	6
2. Gestion de projet .....	7
2.1. L'équipe.....	7
2.2. Organisation.....	7
3. Cahier des charges .....	12
3.1. Persona .....	12
3.2. Besoins & réponses.....	12
4. Spécifications fonctionnelles .....	13
4.1. Intro .....	13
4.2. Design.....	13
5. Spécifications techniques.....	18
5.1. Frontend .....	18
5.2. Backend .....	23
5.3. Base de données .....	29
5.4. Sécurité & Test .....	33
5.5. Déploiement.....	39
6. Réalisation personnelle .....	44
6.1. Spécifications .....	44
6.2. Réalisation .....	45
6.3. Tests.....	54
6.4. Résultats.....	56
7. Veille technologique.....	58
8. Remerciements & conclusion.....	59



# 1.Présentation

## 1.1.Présentation personnelle

Je m'appelle Klervy Le Dez, j'ai 28 ans. J'ai d'abord été mécanicienne, puis j'ai travaillé plusieurs années dans l'industrie en tant qu'intérimaire. J'aime les métiers physiques, mais je me suis rendu compte que je ne m'y épanouissais pas intellectuellement.

C'est donc avec une envie de challenge que je me suis lancée dans une reconversion pour devenir développeuse. Après neuf mois de formation initiale, j'ai entamé un an d'apprentissage chez Arpège, une entreprise éditrice de solutions pour les collectivités. Je suis maintenant à la fin de cette année et vais vous présenter mon projet afin d'obtenir mon titre de Conceptrice développeuse d'applications.

## 1.2.Le Projet

Quand il a été question de construire un projet dans le cadre du titre RCNP, le plus dur a sûrement été de trouver une idée. Après une éternité passée à chercher un concept et autant de temps à trouver un nom, est né DERBEEZ.

DERBEEZ est un site communautaire autour du roller derby, un sport relativement récent, aux règles parfois obscures. C'est pourquoi il nous paraît intéressant de créer une plateforme sur laquelle des néophytes peuvent se retrouver et apprendre.

DERBEEZ se place sur quatre grands axes. Découvrir l'histoire de ce sport, les règles, le matériel, les équipes. Se tenir au courant, avec des articles sur les matchs ou la culture. S'amuser en trouvant son futur nom de joueuse, des quiz, ou même son horoscope. Et enfin, trouver une communauté, petites annonces, échanges d'informations et proposition d'événements. En bref tout pour favoriser le lien et faire le premier pas en ayant moins peur d'être seule.

DERBEEZ utilise comme technologies React.js , Asp.Net (C#), Postgresql ainsi que plusieurs Framework et bibliothèques que je présenterais au cours de ce document. Ce projet a été l'occasion de mettre à profil tout ce qui a été appris au cours de la formation initiale et de l'année d'alternance. En consolidant des acquis, ou en essayant de nouvelles choses.

Le projet décrit dans ce document est un MVP (Produit Minimum Viable) de cette plateforme. Les utilisatrices peuvent se connecter, gérer les petites annonces et accéder à des informations sur ce sport.

## 1.3.The Project

When it came to building a project for the RNCP title, the hardest part was surely to come up with an idea. After an eternity spent searching for a concept and as much time finding a name, DERBEEZ was born.

DERBEEZ is a community site around roller derby, a relatively recent sport, with sometimes obscure rules. This is why it seems interesting to us to create a platform on which neophytes can meet and learn.

DERBEEZ stands on four main axes. Discover the history of this sport, the rules, the equipment, the teams. Keep up to date, with articles about games or culture. Have fun by finding your future player's name, quizzes, or even horoscopes. And finally, find a community, classified ads, information exchange, event proposal. In short everything to promote human link and take the first step with less fear of being alone.

DERBEEZ uses React.js , Asp.Net (C#) and Postgresql technologies, as well as several frameworks and libraries that I'll be presenting in this document. This project was an opportunity to take advantage of everything that was learned during initial training and apprenticeship's year. By consolidating what I have learned, or by trying out new things.

The project described in this document is a MVP (Minimum Viable Product) of this platform. Users can log in, manage classified ads and access information about this sport.

## 1.4.Listes de compétences du référentiel

Développer une application sécurisée
<ul style="list-style-type: none"> <li>❖ Installer et configurer son environnement de travail en fonction du projet.</li> <li>❖ Développer des interfaces utilisateur.</li> <li>❖ Développer des composants métier.</li> <li>❖ Contribuer à la gestion d'un projet informatique.</li> </ul>
Concevoir et développer une application sécurisée organisée en couches
<ul style="list-style-type: none"> <li>❖ Analyser les besoins et maquetter une application.</li> <li>❖ Définir l'architecture logicielle d'une application.</li> <li>❖ Concevoir et mettre en place une base de données relationnelle.</li> <li>❖ Développer des composants d'accès aux données SQL et NoSQL.</li> </ul>
Préparer le déploiement d'une application sécurisée
<ul style="list-style-type: none"> <li>❖ Préparer et exécuter les plans de tests d'une application.</li> <li>❖ Préparer et documenter le déploiement d'une application.</li> <li>❖ Contribuer à la mise en production dans une démarche DevOps.</li> </ul>

## 2.Gestion de projet

### 2.1.L'équipe

Nous sommes une équipe de trois développeuses :

- ❖ **Laurie Biguet** : développeuse en alternance chez Beapp  
Swift, React native et NextJs
- ❖ **Nadège Hellemans** : développeuse en alternance chez D-EDGE  
C#, react, typescript
- ❖ **Klervy Le Dez** : développeuse en alternance chez Arpège  
Asp.Net C#, Postgres, Angular

### 2.2.Organisation

#### Agilité

« Toute organisation qui conçoit un système, au sens large, concevra une structure qui sera la copie de la structure de communication de l'organisation. »

M. Conway

Il nous a fallu d'abord réfléchir à notre organisation d'équipe. Nous avons décidé de baser notre organisation sur des principes de l'Agilité adapté à notre rythme de travail. Des weekly ont été organisé le vendredi, les démos ont était faites à chaque ticket fini et des rétros toutes les trois semaines.

#### Historique des décisions prises

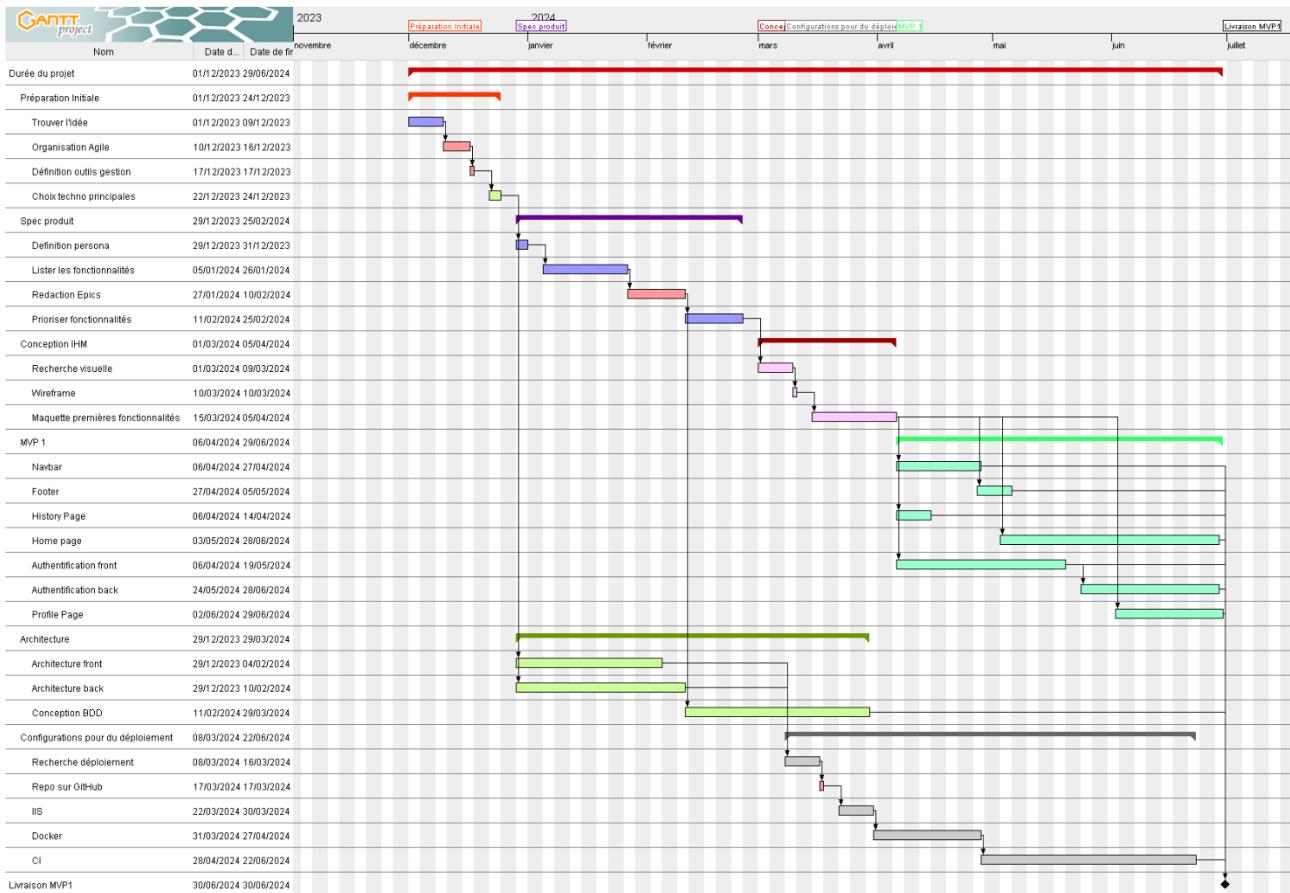
- |                        |                     |                 |
|------------------------|---------------------|-----------------|
| ▶ Calendrier Rapide    | ▶ Maquettes         | ▶ Conception    |
| ▶ Market               | ▶ BDD               | ▶ Planification |
| ▶ Brainstorm d'origine | ▶ Front             | ▶ Back          |
| ▶ Product owner        | ▶ Gestion de Projet | ▶ Infra         |

Nous avons aussi convenu que quand l'une d'entre nous termine un ticket, elle écrit un résumé de ses choix avec la date dans la section “*Historique des décisions prises*” du Notion de l'équipe. Si le choix est important, la documentation qui a motivé ce choix est aussi notée, ce qui permet de débattre de ce choix avec les autres membres de l'équipe.

Après cette première phase de planification, nous avons organisé plusieurs séances de travail en groupe pour définir notre projet et choisir les outils d'organisation du projet.

## Planification du projet

Pour la planification des étapes du projet, il a été décidé d'utiliser un diagramme de Gantt pour assurer le meilleur suivi possible, nous avons utilisé le logiciel *GanttProject*.



Il y a plusieurs phases :

- ❖ Préparation initiale
- ❖ Spécifications produit
- ❖ Architecture
- ❖ Conception IHM
- ❖ Déploiement
- ❖ MVP 1

## Choix des outils d'organisation de projet

- **Slack** : C'est l'outil de messagerie que nous avons utilisé pour communiquer tout au long du projet, que ce soit par messages, mais également en visio quand nous n'étions pas au centre de formation.
- **Notion** : Nous avons utilisé Notion pour créer un historique de nos prises de décisions. Cela nous a également permis d'expliquer les différentes fonctionnalités pour que tout le monde ait une meilleure compréhension du projet. Il nous a aussi permis de regrouper les liens vers les différents outils que nous utilisons ainsi que de répertorier les ressources utiles de chacune.  
C'est aussi l'endroit où nous avons voté certaines questions comme la direction artistique du site, décidé quelles features sont prioritaires ou même le nom du projet.
- **Jira** : Nous avons utilisé Jira pour la gestion des tickets et des sprints. Nous avons découpé le travail en Epic puis en ticket. Selon la nature du ticket, celui-ci est associé à la maquette et a pour description les critères d'acceptations et les cas d'erreurs. Nous nous sommes ensuite assignées les tickets au fur et à mesure, et avons pu suivre clairement leurs avancements en mettant à jour leurs statuts.

The screenshot shows a Jira project board for the 'Bourdons' project. The board is divided into four columns: TO DO (11), IN PROGRESS (4), IN REVIEW (4), and DONE (10). Each column lists several Jira issues with their status, assignee, and progress bar. The 'IN PROGRESS' column includes issues like 'Home Page Cards Component PAGE D'ACCUEIL' and 'Footer PAGE D'ACCUEIL'. The 'DONE' column includes issues like '[DESIGN] Page profil DESIGN/UI' and 'Créer l'architecture de base Back CONFIGURATION ET INSTALLATION'. On the left, there's a sidebar with a search bar, user icons, and dropdown menus for 'Epic' and 'Type'. Below the board, there's a detailed view of an issue titled 'Auth - Intégrer Identity back' with its description, approach, and documentation links.

## Git & Github



Nous avons utilisé Git et Github pour la gestion des versions du code source de l'application. Nous avons créé une organisation afin d'héberger nos 3 repositories qui comprennent le front-end, le back-end ainsi que le projet development qui regroupe la documentation, le troubleshooting et les configurations Docker & IIS.

Pour chacun, nous avons utilisé le workflow GitFlow. Celui-ci permet de travailler ensemble de manière efficace en étant d'accord sur la structure et en évitant les conflits. Il se décompose en branches nommées :

- Main > Cette branche est la branche principale. Elle contient le code stable et prêt à être déployé en production.
- Develop > Cette branche est utilisée pour le développement continu. Les fonctionnalités sont fusionnées ici avant d'être testées et validées pour la production lors d'une version.
- Hotfix > Si un bug critique est découvert en production, une branche Hotfix est créée à partir de Main. Une fois le bug corrigé et testé, la branche est poussée dans Main et Develop.
- Feature > Chaque nouvelle fonctionnalité est développée dans sa propre branche tirée de Develop. Les branches doivent être nommées de la même façon. Exemple : feature/DBZ-02-footer

Chaque fonctionnalité terminée est suivie par une pull request. Cela permet aux autres membres de l'équipe de prendre connaissance du code, de le tester et de le valider si tout est en ordre. Ensuite, celle-ci est mergée par le développeur.

## Choix des stacks techniques

Tous nos choix techniques ont été faits sur deux principes. Premièrement, la cohérence avec le projet, en effet, sa nature de réseau social nécessite de faire des choix de robustesse, scalabilité et de sécurité. L'autre principe est l'aspect personnel, en effet, choisir une stack technique connue (même partiellement) de l'équipe permet un code de meilleure qualité et une plus grande rapidité de développement.

Ces choix seront détaillés dans les spécifications techniques.

## Lister et prioriser les fonctionnalités à développer

Les fonctionnalités prévues ont été déterminées grâce à un persona, puis une liste de ses besoins et leur réponse. Cette étape est détaillée dans le chapitre “Cahier des charges”.

Une fois les idées sélectionnées, nous avons mis en place un système de vote pour prioriser ces fonctionnalités, ainsi que les tâches s'y affiliant. Nous avons pris en compte tout au long de ce projet, les modifications ou imprévus et avons dû par moment reprioriser certaines fonctionnalités.



## Développement du projet

Après toutes cette phase de réflexion est venu le temps de se lancer dans le projet. Pour le début nous avons décidé de nous répartir les tâches selon nos appétences, avec la mise en place de l’architecture front-end, l’architecture back-end et aussi la spécification des fonctionnalités afin d’avoir les bases et un cap clair. Ensuite nous avons décidé de nous répartir les tâches de façon à ce que tout le monde ait l’occasion de toucher à toutes les parties du projet.

## 3.Cahier des charges

### 3.1.Persona

Le public cible de ce site est composé de personnes motivées mais débutantes dans la pratique du roller derby, il est nécessaire de proposer un espace communautaire et bienveillant, dans le respect de l'état d'esprit de ce sport.

*Nous avions déjà une envie commune de débuter le roller derby après avoir assistées à des matchs ensemble, c'était une belle opportunité de travailler sur un sujet où chacune avait sa place et pouvait exprimer ses envies et ses besoins.*

*Finalement, le persona c'est nous.*

### 3.2.Besoins & réponses

Les besoins identifiés à partir de notre persona sont :

- Se sentir en confiance pour oser poser des questions et se lancer.
- Pouvoir échanger avec des personnes dans la même situation.
- S'encourager mutuellement.
- Trouver des bons plans d'achat de matériel de seconde main pour limiter l'investissement de départ.
- Retrouver des fans de roller derby pour aller voir des matchs ensemble.

Nous avons donc ciblé notre réponse sur 4 axes :

- Apprendre :
  - Histoire du roller derby
  - Informations sur la sécurité
  - Glossaire des termes techniques et de l'équipement
  - Entrainement (Liens vers des ressources externes)
  - Profils des équipes
- Se tenir au courant :
  - Matchs et scores
  - Articles sur le roller Derby
  - Articles sur la culture autour du sport
- Trouver une communauté :
  - Petites annonces
  - Post personnels avec des interactions (likes, commentaires...)
  - Annonces d'évènements
- S'amuser :
  - Bingo
  - Générateur de nom
  - Quiz
  - Horoscope

## 4. Spécifications fonctionnelles

### 4.1. Intro

L'utilisatrice de DERBEEZ souhaite se renseigner sur le roller derby, que ce soit sur son histoire et sa pratique. Elle souhaite également avoir un outil qui lui permet d'être en relation avec d'autres sportives et de connaître les évènements à venir.

### 4.2. Design

#### Inspiration

Dans le cadre de la construction de l'identité graphique de l'application, une réflexion approfondie a été menée afin de créer une charte graphique cohérente et en adéquation avec l'image de ce sport.

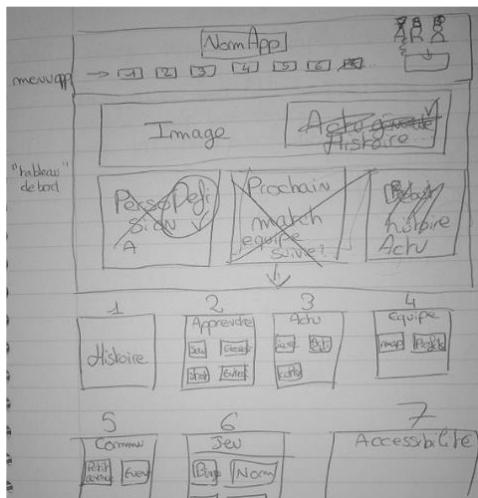
Pour trouver l'inspiration, nous avons effectué des recherches sur les sites spécialisés dans le design tels que 'Dribble', 'Awwward' ou 'Behance'.

▼ Moodboard & inspi

- Choix design
  - Pattern vs. People vs. Color palette
  - Retro vs. Modern vs. NeoRetro
  - Colorful vs. Sobre  
Générateur aléatoire de couleurs → <https://color-wander.surge.sh/?ref=undesign>
  - Flat design → <https://www.justinmind.com/blog/flat-design-vs-material-design-differences/>
  - Minimalism vs. Maximalism
  - Realism vs. Cartoon
  - Accessibilité  
Contraste couleurs → <https://coolors.co/contrast-checker/l12a46-acc8e5>

- ▶ Inspi Nadège :
- ▶ Inspi Klervy :
- ▶ Inspi Laurie :
- ▶ Recherche couleur

#### Wireframe



Pour la structure de nos pages, nous avons dessiné les wireframes sur papier. En petit groupe, c'est la manière la plus efficace pour communiquer. Ils sont très simples mais nous ont permis d'être toutes d'accord sur la structure des pages.

## Couleurs

Le roller derby est connu pour son style très coloré. Nous avons gardé cet esprit en tête lors du choix de notre palette. Nous avons choisi cette palette à l'aide du site “[Coloros.co](#)”.



Nous avons également pris en compte les critères d'accessibilités. Nous avons veillé à ce que les couleurs choisies soient suffisamment contrastées pour être lisibles par toutes les utilisatrices, y compris les personnes atteintes de troubles visuels.

## Color Contrast Checker

Calculate the contrast ratio of text and background colors.

Text color: #000000      Background color: #DAB2D4

Contrast: 11.30      Rating: Very good (5 stars)

Small text: ★★★ Large text: ★★★

Good contrast for small text (below 18pt) and great contrast for large text (above 18pt or bold above 14pt). [Click to enhance](#)

Quote n. 25

We've heard that a million monkeys at a million keyboards could produce the complete works of Shakespeare; now, thanks to the Internet, we know that is not true.

Robert Wilensky

Concernant la typographie, nous avons fait attention à choisir des polices lisibles. Mais également à ce que les textes soient suffisamment aérés et les titres clairement identifiables, pour faciliter la lecture et la navigation sur le site.

## Maquettes

Pour les maquettes, nous avons utilisé l'outil de conception design Figma. Le design a été laissé au soin de Laurie Biguet ayant une plus grande appétence pour cette facette du projet. Néanmoins Figma aidant à la collaboration, nous avons pu participer à l'amélioration de ces maquettes en laissant des remarques ou des questions en commentaire.

The image displays three wireframe prototypes side-by-side, illustrating different design concepts for two websites: Nova Glide and DerbeeZ.

**Nova Glide (Left):** This prototype shows a landing page with a large dark purple central area containing placeholder text. Below it are two call-to-action boxes: a pink one on the left labeled "En savoir plus" and a green one on the right labeled "Rejoins la communauté". A navigation bar at the top includes "Entrainement" (underlined), "Communauté", "Actualité", and "Connexion".

**Nova Glide (Right):** This version features a more compact layout. It has a large dark purple area with placeholder text, a pink "En savoir plus" button, and a green "Rejoins la communauté" button. The navigation bar at the top includes "Nova Glide" (with a logo icon) and a menu icon.

**DerbeeZ (Bottom):** This prototype shows a login/signup screen. On the left is a teal box for "Déjà inscrit-e ?" with fields for "Adresse e-mail" and "Mot de passe", and a "Se connecter" button. On the right is a pink box for "Créer un compte" with fields for "Pseudo", "Adresse e-mail", and "Mot de passe", and a "S'inscrire" button. A navigation bar at the top includes "Histoire", "Entrainement" (underlined), "Communauté", "Actualité", and "Connexion".

Après réflexion, une nouvelle version a été créée afin d'améliorer plusieurs points :

- Accessibilité** : La nouvelle maquette, plus épurée améliore la lisibilité. Une des couleurs a été retirée afin d'avoir un meilleur contraste pour améliorer la lecture du site et alléger l'interface.
- UX** : La nouvelle version offre une meilleure expérience utilisatrice avec une disposition plus logique des éléments et une navigation plus intuitive. Elle donne à l'application un aspect plus moderne et met l'accent sur le contenu.

**Derbeez**

Histoire    **Entrainement**    Communauté    Actualité    Connexion

## Présentation site

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse.

[Inscription](#)

**Histoire**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[En savoir plus](#)

**Entraînements**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[Connexion](#)

**Communauté**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[Rejoins la communauté](#)

**Défi / jeux**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[C'est parti !](#)

**Plus d'actualités**

Copyright    Mentions légales    Contact

**Derbeez**

Présentation site

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse.

[Inscription](#)

**Histoire**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[En savoir plus](#)

**Entraînements**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[Connexion](#)

**Communauté**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[Rejoins-nous](#)

**Défi / jeux**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

[C'est parti !](#)

**Plus d'actualités**

Copyright    Mentions légales    Contact

**DerbeeZ**

- [Histoire](#)
- [Entrainement](#)
- [Communauté](#)
- [Actualité](#)
- [Connexion](#)

## Présentation histoire

Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

**Histoire**

Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

**Histoire**

Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Copyright    Mentions légales    Contact

**DerbeeZ**

- [Histoire](#)
- [Entrainement](#)
- [Communauté](#)
- [Actualité](#)
- [Connexion](#)

**Déjà inscrit·e ?**

Mot de passe oublié ?

Je n'ai pas encore de compte ?

**Se connecter**

Copyright    Mentions légales    Contact

**DerbeeZ**

- [Histoire](#)
- [Entrainement](#)
- [Communauté](#)
- [Actualité](#)
- [Connexion](#)

**Hello, User !**

pronom

**Bio**  
Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

**Mes annonces**

**Filtre**

Copyright    Mentions légales    Contact

**DerbeeZ**

**Hello, User !**

Pronom  
Loreum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse.

**Mes annonces** Filtres

Copyright    Mentions légales    Contact

## 5. Spécifications techniques

### 5.1. Frontend

#### 5.1.1. Technologie & structure

##### Technologies utilisées

**ReactJs** : Nous l'avons choisi en raison de sa capacité à créer une interface utilisateur réactive grâce à l'utilisation du Virtual DOM qui permet de mettre à jour l'interface de manière efficace, en ne modifiant que les parties nécessaires.

**TypeScript** : Son utilisation permet de détecter les erreurs de types, ce qui améliore la maintenabilité et la fiabilité du code.

**Zustand** : La création de stores permet de partager les états dans l'application. Il se prête bien aux applications de taille moyenne.

**Zod** : Nous l'utilisons pour la validation de schéma en spécifiant des règles de saisie, ce qui a deux avantages, guider l'utilisateur sur les erreurs de validation et rendre nos formulaires plus sécurisés.

**I18n** : Permet de centraliser les textes dans un fichier Json. A chaque changement dans ce fichier, les modifications sont répercutées dans l'application. Permet également de gérer la traduction.

**ESLint / Prettier** : Ces deux outils permettent d'améliorer la qualité du code mais également de fournir les mêmes règles d'écriture pour tous les développeurs.

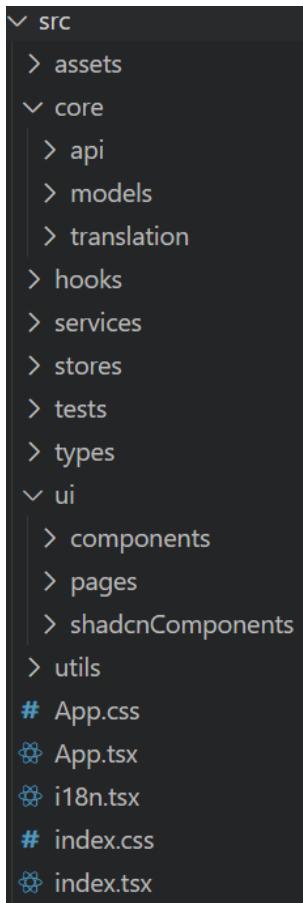
##### Structure du projet

Au plus haut niveau de l'application, on retrouve les fichiers utilisés pour :

- **Les configurations** : `.eslintrc.json`, `.prettierrc.json`, `tailwind.config.js`, `jest.config.js`, `jest.setup.js`
- **Gérer les dépendances** : `package.json`, `package-lock.json`
- **Ignorer des fichiers** : `.gitignore`, `.prettierignore`
- **Variables d'environnements** : `.env`, `.env.modele`
- **Les documentations** : `CHANGELOG.md`, `README.md`, `LICENSE`
- **IIS** : Le fichier `web.config` du projet front
- **Docker** : Le `Dockerfile` du projet front

Ensuite nous retrouvons dans le dossier src :

**Dossier Assets** : contient les images, SVG et les fonts.



- **Dossier Core** : contient plusieurs autres dossiers :
  - **Dossier API** : fichiers pour gestion de la connexion avec l'API.
  - **Dossier Models** : pour les DTO.
  - **Dossier Translation** : pour le fichier Json de *i18n*.
- **Dossier Hooks** : contient les hooks personnalisés.
- **Dossier Service/BackendService** : centralise les appels API.
- **Dossier Store** : contient les stores Zustand.
- **Dossier Tests** : contient les tests unitaires.
- **Dossier Types** : contient les types.
- **Dossier UI** : contient plusieurs autres dossiers :
  - Les composants
  - Les pages
  - Les schémas des composants importés de la librairie Shadcn UI
- **Dossier Utils** : regroupe les fonctions réutilisables.
- **Les fichier racines** : Fichiers définissant la structure, le style et le comportement de base de l'application.

## Routage

```

const App = () => {
  return (
    <main className='h-screen flex flex-col'>
      <header>
        <Navbar />
      </header>
      <section className='flex-grow'>
        <Routes>
          <Route path={paths.home} element={<Home />} />
          <Route path={paths.login} element={<Login />} />
          <Route path={paths.register} element={<Register />} />
          <Route path={paths.forgot} element={<Forgot />} />
          <Route path={paths.history} element={<History />} />
          <Route path={paths.logout} element={<Logout />} />
          <Route path={paths.error} element={<Error />} />
          <Route path={paths.news} element={<News />} />
          <Route
            path={paths.profile}
            element={<PrivateRoute element={Profile} />}
          />
          <Route
            path={paths.training}
            element={<PrivateRoute element={Training} />}
          />
          <Route
            path={paths.community}
            element={<PrivateRoute element={Community} />}
          />
        </Routes>
      </section>
    </main>
  );
}

export default App;

```

Le routage permet à l'application de charger une nouvelle vue en fonction de l'URL demandée, mais également la page correspondante à afficher. ReactJs ne fournit pas de routing intégré, nous avons donc utilisé la librairie *reactrouter-dom* pour nous fournir les Routes.

```

export const paths = {
  home: '/',
  login: '/login',
  logout: '/logout',
  error: '/error',
  register: '/register',
  forgot: '/forgot',
  profile: '/profile',
  history: '/history',
  training: '/training',
  community: '/community',
  news: '/news',
};

```

Pour les routes qui ne sont accessibles que lorsque l'utilisatrice est connecté, j'ai mis en place un composant qui enrobe la route et qui vérifie si celui-ci est connecté grâce au store d'authentification. Si ce n'est pas le cas, il est renvoyé vers la page de connexion.

```
function PrivateRoute({ element, ...rest }: Props) {
  const { isAuth, isLoading, checkAuthentication } = useAuthStore();
  const location = useLocation();

  useEffect(() => {
    checkAuthentication();
  }, [location]);

  if (isLoading) {
    return (
      <main className='h-full flex justify-center items-center overflow-y-auto'>
        <LoadingSpinner size={100} />
      </main>
    );
  }

  if (!isAuth) {
    return <Navigate to={paths.login} state={{ from: location }} replace />;
  }

  return <Element {...rest} />;
}

export default PrivateRoute;
```

### 5.1.2.Communication externe

Pour la communication entre notre front et notre API, nous avons décidé d'utiliser Axios qui est une librairie qui permet de simplifier et centraliser les requêtes HTTP.

```
export type ApiResponse<T> = {
  isSuccess: boolean;
  data: T;
  errorMessage: string;
};
```

J'ai d'abord décidé d'un type, `ApiResponse<T>`, qui a pour but de cadrer quel format on attend de notre API. Il accepte un type générique qui correspond à celui des données dans le champ `data`, celui-ci est précisé au moment de l'appel.

J'ai ensuite mis en place la structure permettant de gérer les appels API, tout d'abord le fichier `api.ts` qui contient la création de la base de l'url avec les variables d'environnement. L'URL est composée du nom de domaine, de l'objet requêté ainsi que la route demandée :

```
const apiUrl =
  process.env.REACT_APP_API_URL_DOCKER ?? process.env.REACT_APP_API_URL;

const api = axios.create({
  baseURL: apiUrl,
  withCredentials: true,
});
```

La deuxième partie de la route est définie dans un fichier `apiNames.ts` permettant une modification simplifiée des noms de route.

```

const api = axios.create({
  baseURL: apiUrl,
  withCredentials: true,
});

api.interceptors.response.use(
  (response: AxiosResponse) => {
    const apiResponse = response.data;
    return apiResponse;
  },
  <T>(error: AxiosError) => {
    return handleError(error as AxiosError<ApiResponse<T>>);
  },
);

function handleError<T>(error: AxiosError<ApiResponse<T>>): never {
  if (error.response) {
    // The request was made and the server responded with
    // a status code out of the range of 2xx
    console.log('Error response:\n', error.response);
  } else {
    // The request was made but no response was received or Something
    // happened in setting up the request that triggered an Error
    console.log('Something happened:\n', error);
  }
  throw error.response?.data?.errorMessage ?? t('error.generic');
}

export default api;

```

Le fichier *api.ts* contient aussi des *interceptors* Axios qui permettent de façonne la réponse à notre besoin en cas d'échec ou de réussite de la promesse. C'est à ce moment qu'on extrait *ApiResponse* des données de *AxiosResponse* pour une utilisation des réponses simplifiées dans lors des appels asynchrones. C'est aussi dans ce fichier que l'erreur est gérée en ne renvoyant que le message de l'erreur.

J'ai aussi décidé de regrouper les appels API dans des fichiers par objets requêtés, toujours dans une logique de simplification.

Exemple de de *backendService* pour l'entité *Person*:

```

export async function getPerson(
  personId: number,
): Promise<ApiResponse<PersonDetails>> {
  return await api.get(GET_PERSON, {
    params: { personId },
  });
}

export async function updatePersonDetails(
  personDetails: PersonDetails,
): Promise<ApiResponse<PersonDetails>> {
  return await api.put(UPDATE_PERSON, personDetails);
}

```

Exemple d'appel d'API dans un composant :

```
updatePersonDetails: async (updatedDetails: PersonDetails) => {
  set({ isLoading: true, error: null });
  try {
    const response = await updatePersonDetails(updatedDetails);
    if (response.isSuccess) {
      set({ personDetails: response.data });
    } else {
      set({ error: response.errorMessage });
    }
  } catch (error) {
    set({ error: error as string });
  } finally {
    set({ isLoading: false });
  }
},
```

### 5.1.3.Interfaces

#### Technologies utilisées

**Shadcn** : Bibliothèque de composants qui fournit des éléments préconçus et personnalisables pour faciliter le développement d'interfaces utilisateur.

**Tailwind** : Tailwind est un framework CSS qui permet de créer des designs personnalisés rapidement et de manière simplifiée avec une approche en priorité mobile.

#### Accessibilité

L'accessibilité numérique consiste à rendre les contenus et services numériques compréhensibles et utilisables par les personnes handicapées.

Il existe un référentiel général d'amélioration de l'accessibilité (RGAA). Il définit un ensemble de règles et de critères à respecter pour que les sites web et les applications mobiles soient accessibles à tous les utilisateurs, quels que soient leurs besoins ou leurs capacités ([accessibilite.numerique.gouv.fr](http://accessibilite.numerique.gouv.fr)).

Pour Derbeez, nous avons fait en sorte d'utiliser :

- Des balises HTML appropriées pour structurer le contenu, de manière qu'il soit clair et facile à comprendre pour les lecteurs d'écran.
- Des attributs ARIA pour fournir des informations supplémentaires.
- Des attributs Alt sur les balises image pour fournir une description textuelle. Ces attributs améliorent également le référencement (SEO) car les moteurs de recherche les utilisent pour améliorer le classement des pages web dans les résultats de recherche.

## Responsivité

La responsivité est une approche de conception web qui vise à offrir une expérience utilisateur optimale, quelle que soit la taille de l'écran de l'appareil utilisé. Nous avons conçu l'application de manière qu'elle soit responsive.

Pour ce faire, nous avons utilisé les utilitaires fournis par Tailwind basés sur les media queries qui sont des fonctionnalités CSS qui permettent d'appliquer des styles spécifiques en fonctions des caractéristiques de l'appareil utilisé. Tailwind utilise un système de grille de breakpoints :

Breakpoint prefix	Minimum width	CSS
`sm`	640px	`@media (min-width: 640px) { ... }`
`md`	768px	`@media (min-width: 768px) { ... }`
`lg`	1024px	`@media (min-width: 1024px) { ... }`
`xl`	1280px	`@media (min-width: 1280px) { ... }`
`2xl`	1536px	`@media (min-width: 1536px) { ... }`

Exemple d'utilisation dans un de nos composants :

```
<CardHeader className='flex-col min-[450px]:flex-row justify-between items-center pb-1'>...
</CardHeader>
<CardContent className='flex justify-center xl:justify-around flex-wrap mt-5 gap-5 md:gap-3 '>...
</CardContent>
```

## 5.2.Backend

### 5.2.1.Technologies & Structure

#### Technologies utilisées

**ASP.Net avec C#** : C'est la version web du framework .Net, il permet des performances élevées ce qui est important pour les applications web qui doivent gérer de grandes quantités de données et de trafic comme un réseau social, ce qui offre une expérience utilisateur de qualité. De plus, il permet de créer des API sécurisées et robustes.

**Entity framework** : C'est un ORM (Object Relational Mapping), il permet de faciliter la communication avec la base de données en éliminant la nécessité d'utiliser du code SQL manuellement. C'est également une couche d'abstraction qui offre une sécurité supplémentaire, notamment pour les attaques d'injection SQL.

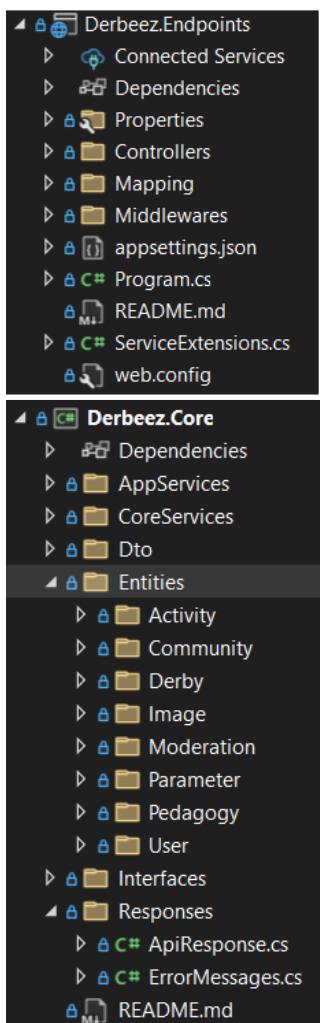
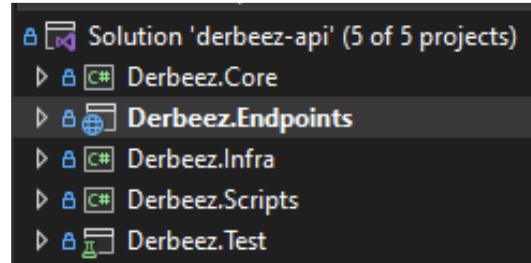
**Identity framework** : C'est un système de gestion d'identité qui permet de gérer les utilisateurs, les rôles et les autorisations. Il prend en charge l'authentification multi-facteurs, la gestion des mots de passe et les verrous de compte. Tout en étant personnalisable. Nous l'avons utilisé pour notre authentification.

**AutoMapper** : C'est une bibliothèque qui facilite la transition d'un objet à l'autre, de manière automatique si les objets sont semblables, si ce n'est pas le cas on peut personnaliser les configurations.

**Serilog** : Utilisé pour les logs de notre back-end.

## Structure du projet

Avec ASP.Net les fichiers sont regroupés dans des projets eux-mêmes regroupés dans une solution ici nommé *derbeez-api*.



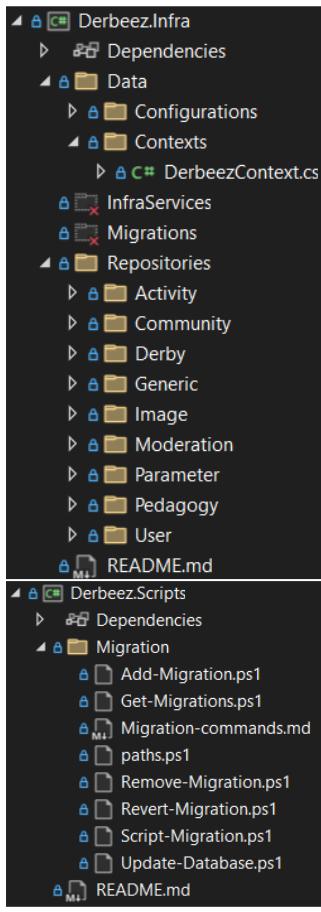
**Derbeez.Endpoints** est responsable des points de terminaison de l'API, facilitant la communication entre l'interface utilisateur et les services back-end. Il contient les contrôleurs qui reçoivent les requêtes de l'utilisateur et les transmettent aux services appropriés.

Il contient également la classe *Program.cs* qui est le point d'entrée par défaut de l'application. C'est à partir de cette classe que l'exécution du programme commence et où certaines des bibliothèques sont configurées.

**Derbeez.Core** constitue le cœur de l'architecture de l'application. Ce module encapsule la logique métier essentielle et les définitions de données, fournissant ainsi une base solide pour les services et les interfaces qui interagissent avec d'autres parties de l'application.

On y trouve :

- Les entités qui définissent la structure des objets conformément à la base de données.
- Les DTO (Data Transfer Objects) utilisés pour encapsuler les données et les transférer vers le front-end.
- Les interfaces qui définissent les contrats pour les services. Elles ne contiennent pas de code, juste les signatures des méthodes. Elles permettent une couche d'abstraction entre les projets.

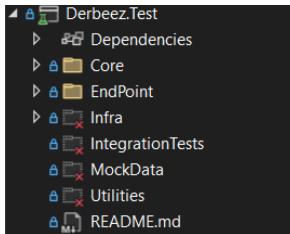


**Derbeeze.Infra** gère l'infrastructure de données de l'application.

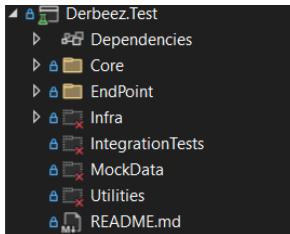
On y trouve :

- Le dossier Data qui contient les configurations des modèles.
- Le DbContext, ici *DerbeezeContext.cs*. C'est est la classe centrale d'Entity Framework qui gère les connexions à la base de données et permet de manipuler les entités via des opérations CRUD.
- Le dossier Migrations qui fournit les scripts de migration de la base de données générés par Entity Framework.
- Le dossier Repositories qui regroupe les fonctionnalités liées aux opération CRUD (Create, Read, Update, Delete) sur les modèles de données.

**Derbeeze.Scripts** contient les scripts utiles de l'application.



**Derbeeze.Test** contient les tests unitaires & d'intégrations.



### 5.2.2. Construction de l'architecture

Pour faire les bons choix lors de la conception de l'architecture de notre back-end, j'ai d'abord lu des articles de personnes ayant beaucoup apporté à ce domaine, comme Robert C. Martin (Uncle Bob), mais aussi des débats de développeurs plus spécialisés dans ASP.NET sur divers forums d'échanges techniques, enfin j'ai consulté des collègues et des encadrants de l'école pour approfondir les points qui n'était pas clairs pour moi.

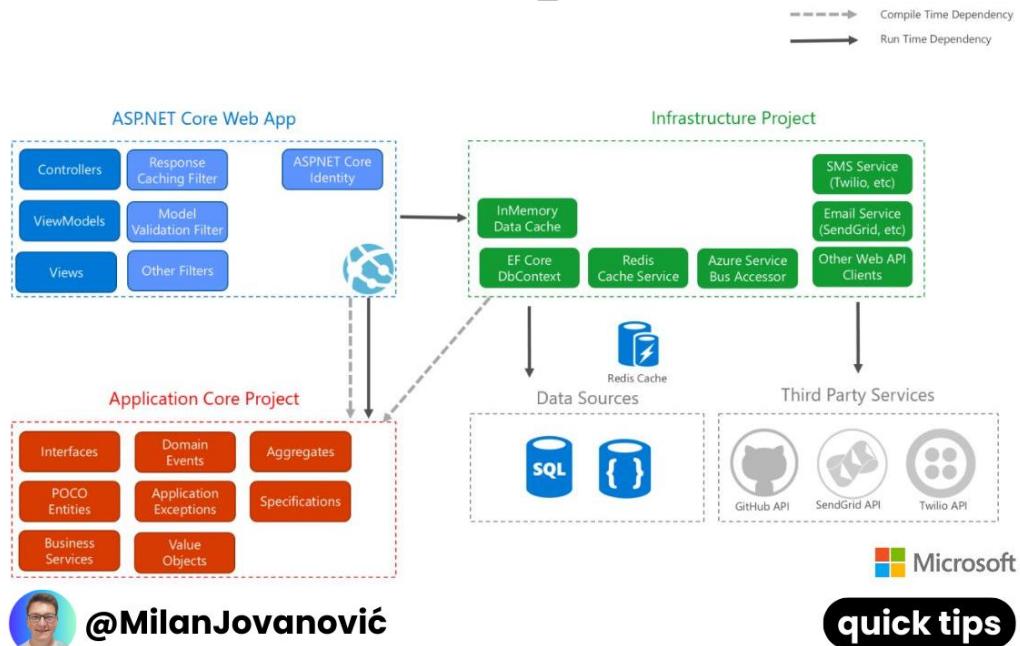
De cette recherche s'est dégagée une architecture, dite "Clean architecture" qui comporte plusieurs principes :

- Indépendance des frameworks, de l'interface utilisateur, de la base de données et des API externes.
- Testabilité élevée grâce à l'isolement des composants.
- Structure en couches concentriques avec des dépendances allant de l'extérieur vers l'intérieur.

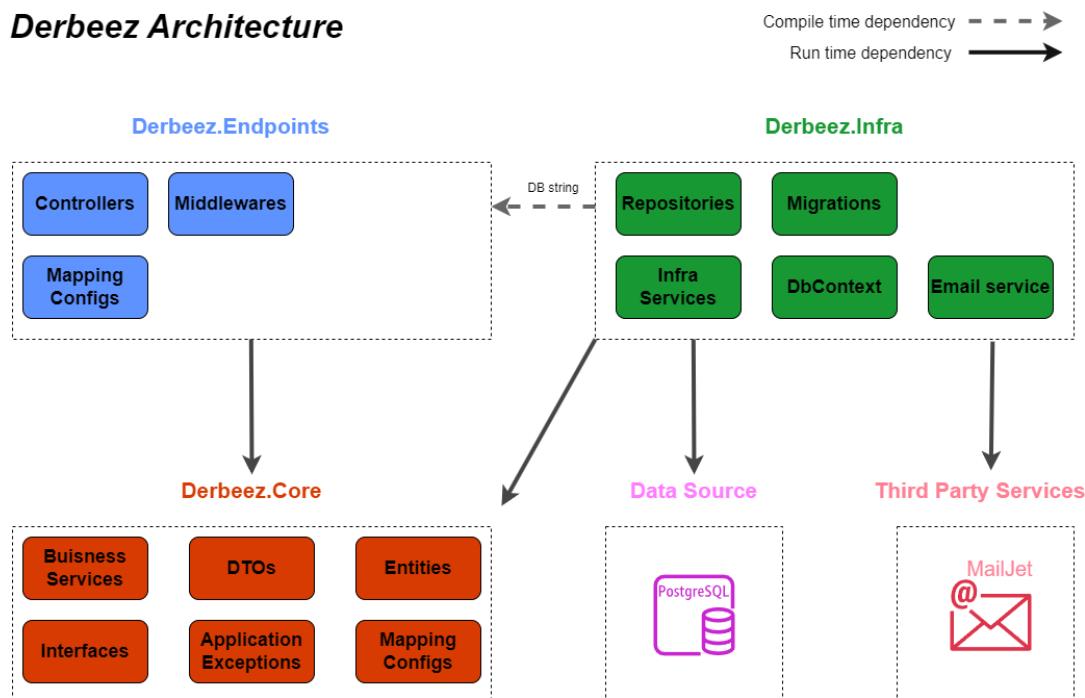
Mais cette architecture peut être complexe pour des débutantes :

# Clean Architecture

## ASP.NET Core diagram



C'est pourquoi j'ai décidé de créer une Clean architecture "légère", pour faciliter une prise en main facile et rapide :



En plus, pour faciliter la prise en main du code j'ai mis en place :

- Configuration de base AutoMapper.
- Un service utilisant un type Générique pour les opérations crud de base ne nécessitant pas de logique.
- Un fichier *ErrorResponse* pour variabiliser les messages d'erreurs pour l'utilisatrice.
- Un middleware de gestion des erreurs, qui lors d'une erreur (non-métier) :
  - Inscrit l'erreur dans le log.
  - Écrit dans la réponse une erreur compréhensible pour l'utilisatrice.
  - Écrit dans la réponse une erreur technique pour le débogage.

```
/// <summary>
///     This middleware intercepts unhandled exceptions in the HTTP request pipeline and performs the following actions:
///     1. Formats the error response and Logs the error using Serilog for logging.
/// </summary>
/// <remarks>
///     Constructor of the middleware.
/// </remarks>
/// <param name="next">The request delegation to call the next middleware in the pipeline.</param>
1 reference
public class ExceptionMiddleware(RequestDelegate next)
{
    private readonly RequestDelegate _next = next;

    /// <summary>
    ///     Asynchronous invocation method of the middleware.
    /// </summary>
    /// <param name="httpContext">The current HTTP context.</param>
0 references
    public async Task InvokeAsync(HttpContext httpContext)
    {
        try
        {
            // Call the next middleware in the pipeline.
            await _next(httpContext);
        }
        catch (Exception ex)
        {
            // If an exception is thrown, handle it using the HandleExceptionAsync method.
            await HandleExceptionAsync(httpContext, ex);
        }
    }

    /// <summary>
    ///     Method to handle unhandled exceptions and format error responses.
    /// </summary>
    /// <param name="context">The current HTTP context.</param>
    /// <param name="exception">The raised exception.</param>
1 reference
    private static Task HandleExceptionAsync(HttpContext context, Exception exception)
    {
        // Set the response content type and HTTP status code.
        context.Response.ContentType = "application/problem+json";
        context.Response.StatusCode = GetStatusCode(exception);

        Log.Error(exception, "An unhandled exception has occurred while processing the request.");

        var response = JsonConvert.SerializeObject(ApiResponse<string>.Failure(errorMessage: ErrorMessages.genericError, data: exception.Message));

        // Serialize the ProblemDetails object to JSON and write the response.
        return context.Response.WriteAsync(response);
    }

    1 reference
    private static int GetStatusCode(Exception exception)
    {
        return exception switch
        {
            ArgumentNullException _ or ArgumentOutOfRangeException _ => (int) HttpStatusCode.BadRequest,
            UnauthorizedAccessException _ => (int) HttpStatusCode.Unauthorized,
            DbUpdateException _ => (int) HttpStatusCode.Conflict,
            _ => (int) HttpStatusCode.InternalServerError
        };
    }
}
```

- Des scripts PowerShell pour effectuer les migrations qui étaient devenues plus complexe à cause de la séparation de la solution en plusieurs projets.

```

param(
    [Parameter(Mandatory=$true)]
    [string]$Name
)

function Add-Migration {
    [CmdletBinding()]
    param(
        [Parameter(Mandatory=$true)]
        [string]$Name
    )

    $pathsScriptPath = Join-Path -Path $PSScriptRoot -ChildPath 'paths.ps1'
    . $pathsScriptPath

    dotnet ef migrations add $Name --project $InfraProjectPath --verbose --startup-project $EndpointsProjectPath
}

Add-Migration -Name $Name

```

### 5.2.3.Communication externe

#### Choix de gestion des retours API

Après le même processus de recherche que décrit précédemment, j'ai décidé en accord avec mes collègues de gérer les retours API sur deux axes.

Comme défini plus tôt dans ce dossier, `ApiResponse<T>` est le standard de communication entre notre API et notre front-end, il contient plusieurs propriétés :

- **IsSuccess** : Si la requête est un succès d'un point de vue métier.
- **Data** : Le contenu demandé, son type est déterminé à la création de la réponse.
- **ErrorMessage** : Contient le message pour l'utilisatrice en cas d'erreur au sens métier, par exemple "Aucun post trouvé."

Cet objet contient aussi deux méthodes d'initialisation :

- **ApiResponse<T>.Success** : il n'y a pas eu d'erreur métier, on précise les données :

```
return ApiResponse<PersonDto>.Success(personDto);
```

- **ApiResponse<T>.Failure** : il y a une erreur métier, on précise le message d'erreur :

```
if (person == null)
    return ApiResponse<PersonDto>.Failure(ErrorMessages.userNotFound);
```

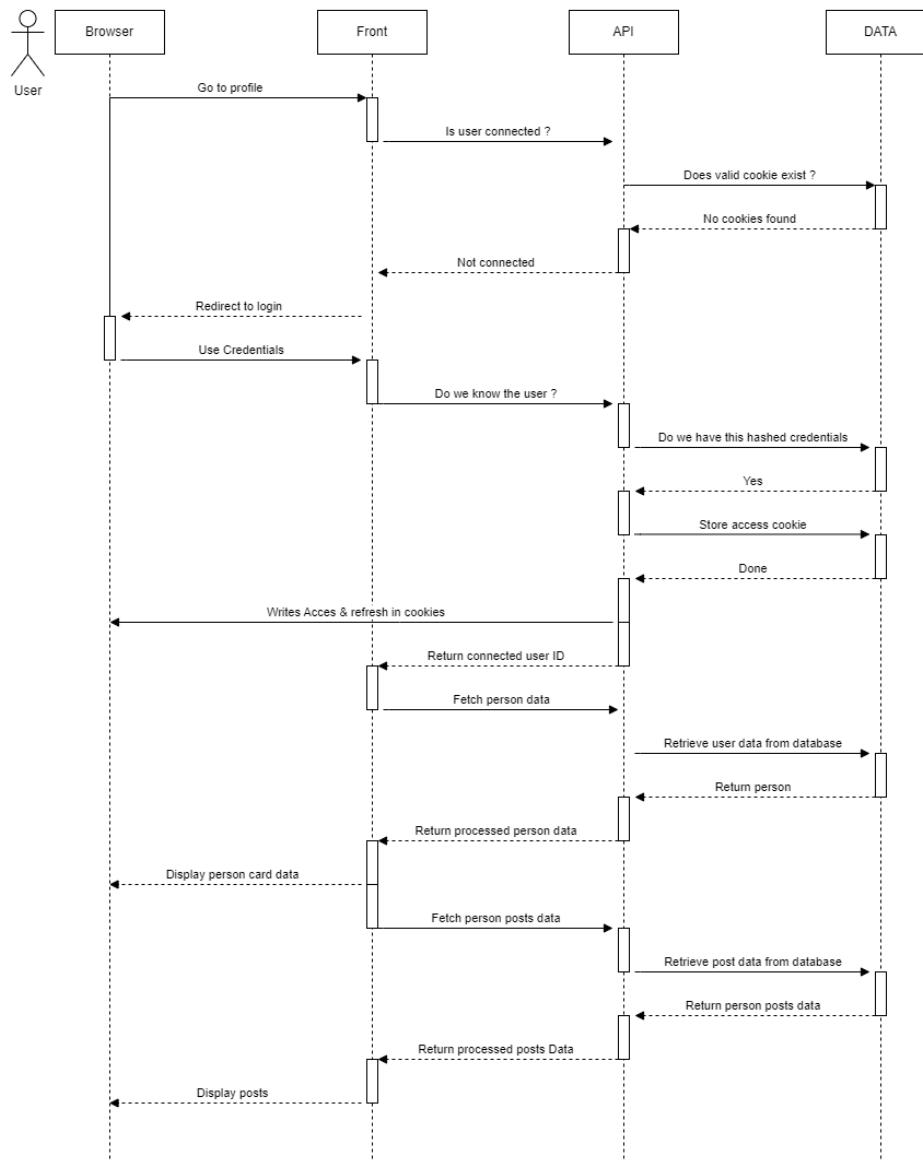
L'autre question est d'utiliser des statuts HTTP comme "401 Unauthorized" ou "404 Not Found" pour les erreurs métier ou alors réserver ces statuts aux erreurs de code.

J'ai choisi la seconde option, renvoyant toujours "200 (OK)" pour les requêtes réussies et indiquant les erreurs métier via l'objet `ApiResponse<T>`.

Ce choix permet une uniformité des réponses, sépare clairement les préoccupations (techniques vs métier), simplifie le traitement côté frontend, et améliore la maintenabilité en centralisant la gestion des erreurs métier.

### Diagramme de séquences

Ce diagramme décrit les étapes séquentielles et les interactions entre les différents composants du système. Ici il illustre le flux de connexion d'un utilisatrice, en montrant comment les demandes sont traitées à travers les différentes couches du système (BROWSER, WEB, API, BDD). Ce diagramme montre en détail la validation des identifiants et la récupération des informations du profil utilisatrice.



## 5.3. Base de données

### 5.3.1. Postgresql

Nous avons choisi PostgreSQL car c'est un système de gestion de bases de données relationnelles.

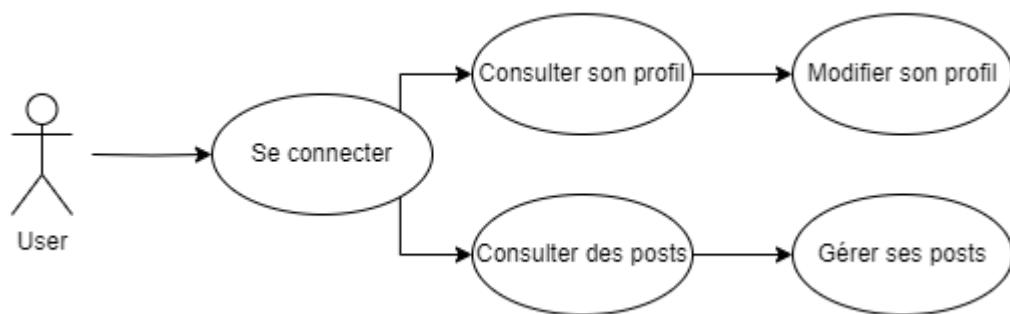
Utiliser une base de données relationnelle avec un projet utilisant Entity Framework est le meilleur choix, car ce framework est spécifiquement conçu pour gérer les entités et leurs relations.

### 5.3.2.Conception

Pour la conception de la base de données, j'ai utilisé la méthode Merise. Pour faire plus simple, je vais présenter les différentes phases de conception sur une seule fonctionnalité, ici la gestion des posts liés à l'utilisatrice qui me semble représentative. Je finirais néanmoins par le diagramme ER (Relation d'entité) complet de la base générée par le logiciel Dbeaver.

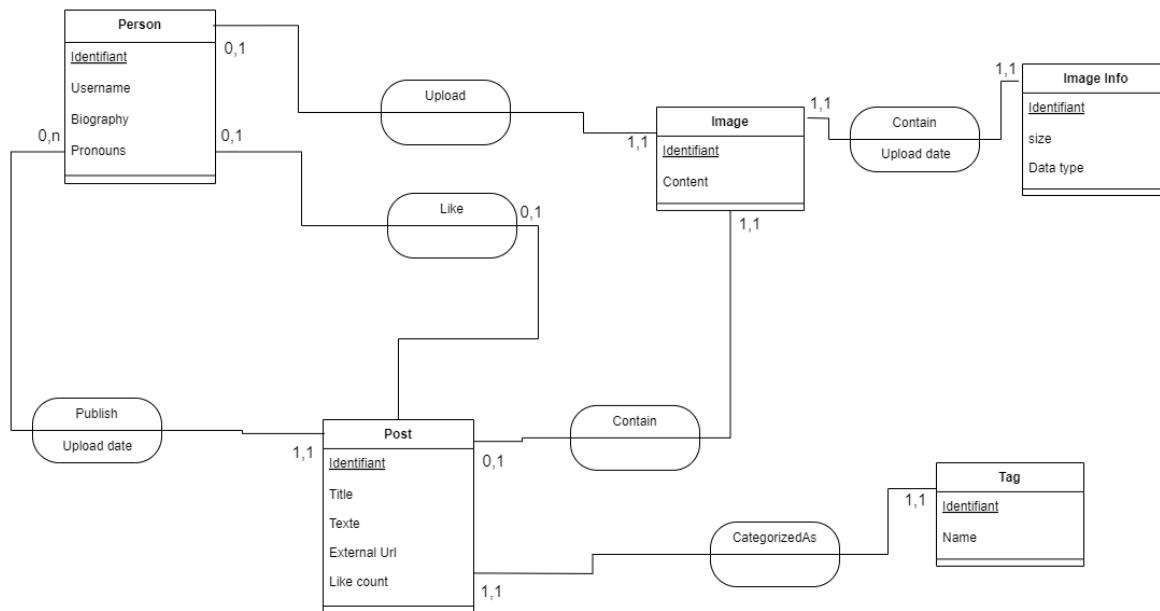
#### Etude préliminaire

Diagramme de cas d'utilisation nous permet d'illustrer les interactions de l'utilisatrice avec l'application. Cela permet de définir les besoins fonctionnels et les scénarios d'utilisation.



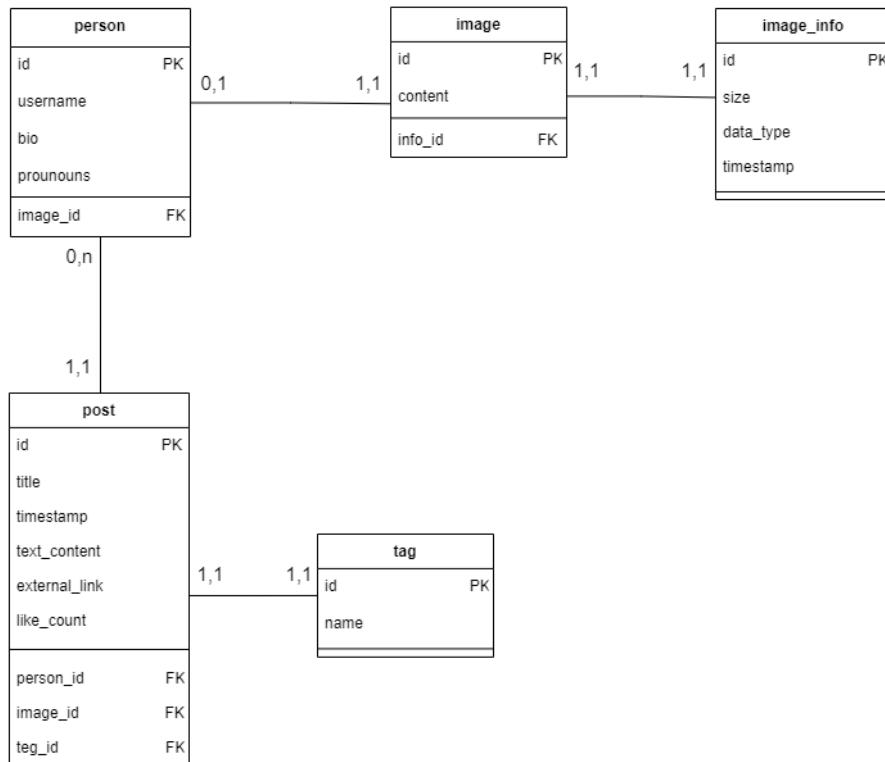
#### Phase conceptuelle

Le MCD (Modèle conceptuel de données) a pour objectif de représenter de manière abstraite les données de l'application indépendamment de toute considération technique en se concentrant sur le métier.



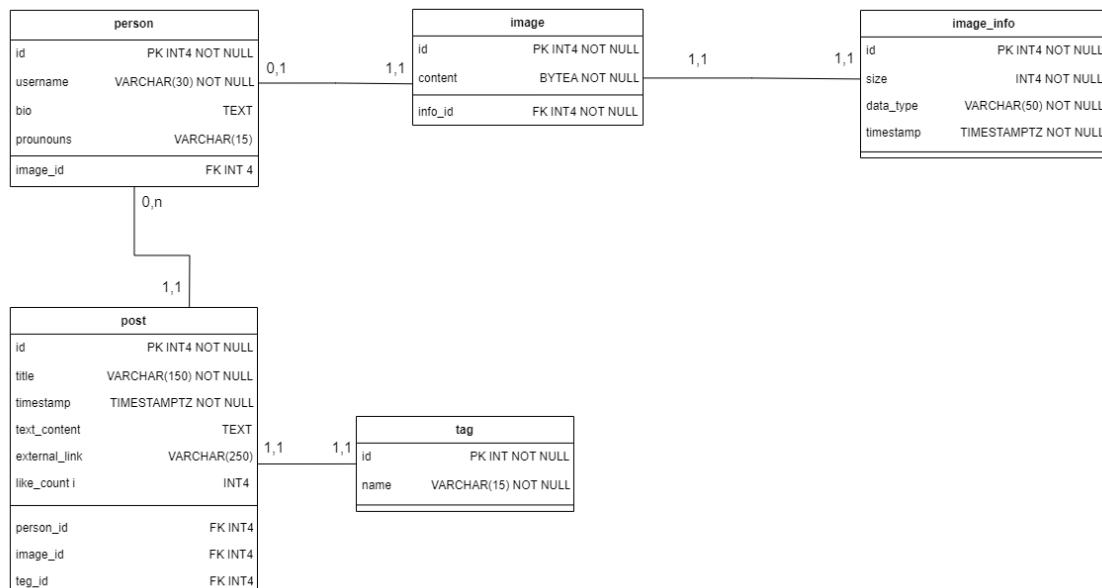
## Phase logique

Le MLD (Modèle logique de données) a pour objectif de transformer le modèle conceptuel en un modèle plus détaillé et adapté à un système de gestion de base de données relationnelle (SGBDR) mais toujours indépendant de tout SGBDR spécifique.

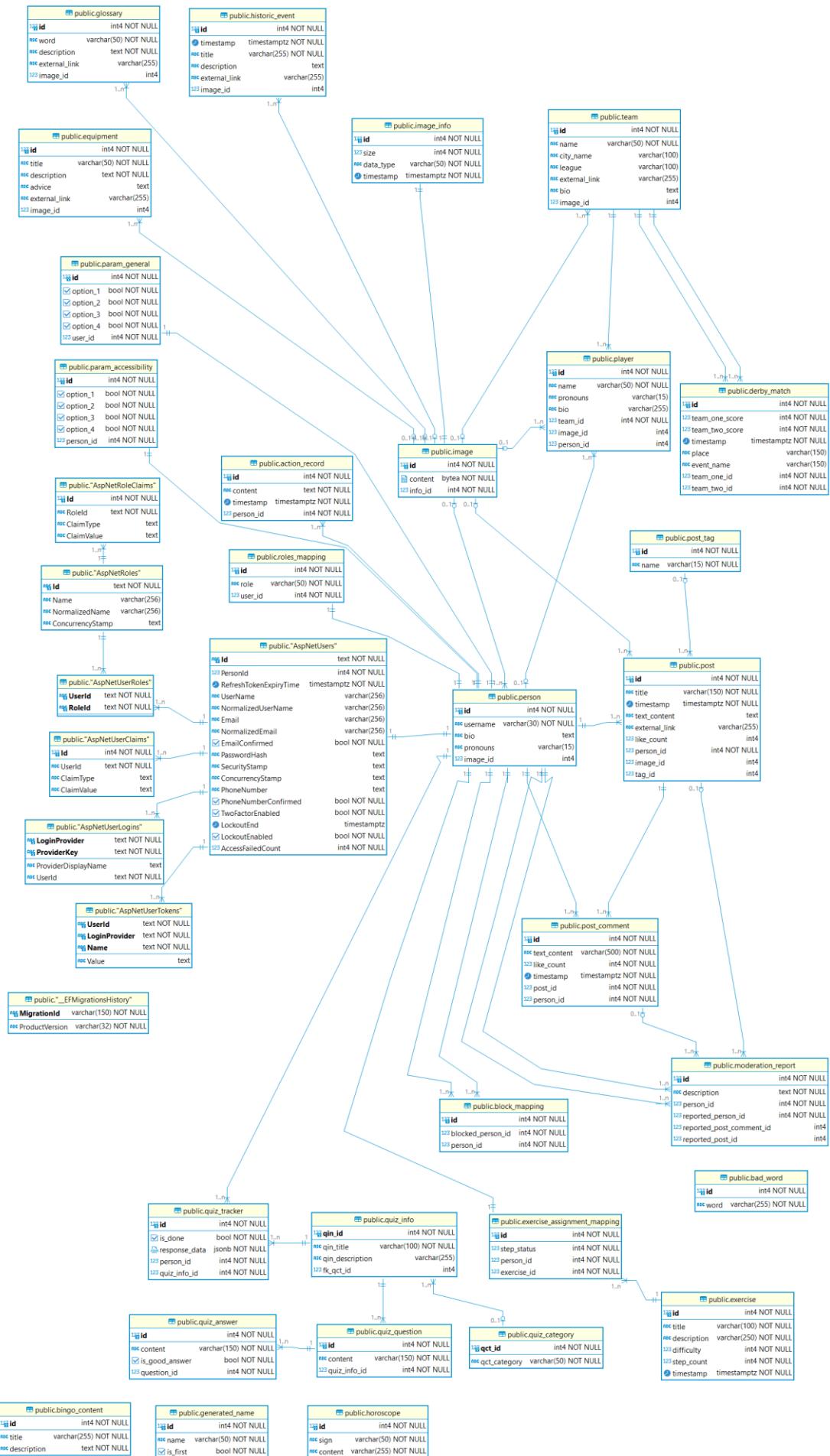


## Phase Physique

Le MPD (Modèle Physique de données) a pour objectif de concrétiser le modèle logique en tenant compte des contraintes techniques et des spécificités du SGBDR choisi.



## Base complète



Vu que les besoins métiers de l'application avaient été bien définis dès le départ, j'ai directement créé la base de données "entièr". En effet une fois lancée dans sa conception, il était moins coûteux en temps de réfléchir toute la base dès le départ plutôt que d'y revenir au fur et à mesure du développement des fonctionnalités, qui nous faisait risquer plus d'incohérences dans notre base.

## 5.4.Sécurité & Test

### 5.4.1.Authentication

Pour assurer une authentification sécurisée des utilisatrices de la plateforme DERBEEZ, plusieurs mesures ont été mises en place.

#### Identity framework & configuration

L'authentification est un sujet trop sensible pour la faire entièrement de zéro, nous avons donc utilisé Identity Framework qui réduit les risques d'erreurs de sécurité souvent présentes dans les implémentations personnalisées.

L'utilisation d'Identity renforce significativement la sécurité de notre application. Il amène des pratiques de sécurité éprouvées, comme le hashage sécurisé des mots de passe avec l'algorithme PBKDF2 (RFC 2898), l'authentification multi-facteurs, et le support des jetons de sécurité pour les opérations sensibles. De plus, il simplifie la gestion des politiques de sécurité, des verrouillages de compte et des récupérations de mot de passe, garantissant ainsi une protection complète et cohérente des données utilisatrice et de l'intégrité de notre application.

Nous avons mis en place plusieurs configurations pour garantir une authentification sécurisée.

#### Sécurité des Cookies :

- **HttpOnly** : Nous avons configuré les cookies contenant les jetons avec l'attribut *HttpOnly* qui les rend inaccessibles via JavaScript et réduit ainsi les risques d'attaques de type XSS (Cross-Site Scripting).

```
[HttpPost("login")]
public async Task< IActionResult> Login(string email, string password)
{
    var user = await _userManager.FindByEmailAsync(email);
    if (user == null || await _userManager.CheckPasswordAsync(user, password))
    {
        return Unauthorized(new { message = "Invalid credentials" });
    }

    var jwtToken = await _authService.GenerateJwtToken(user);
    var refreshToken = await _authService.GenerateRefreshToken();

    // Add the tokens to the database
    await _userManager.SetAuthenticationTokenAsync(user, JwtBearerDefaults.AuthenticationScheme, "AccessToken", jwtToken);
    await _userManager.SetAuthenticationTokenAsync(user, "MyApp", "RefreshToken", refreshToken);

    // Update the refresh token expiration date
    user.RefreshTokenExpiryTime = refreshTokenExpiryTime;
    await _userManager.UpdateAsync(user);

    // Set the tokens as cookies
    Response.Cookies.Append("AccessToken", jwtToken, new CookieOptions
    {
        HttpOnly = true,
        Secure = true,
        SameSite = SameSiteMode.Strict,
        Expires = DateTime.UtcNow.AddMinutes(60)
    });

    Response.Cookies.Append("RefreshToken", refreshToken, new CookieOptions
    {
        HttpOnly = true,
        Secure = true,
        SameSite = SameSiteMode.Strict,
        Expires = refreshTokenExpiryTime
    });

    return Ok(new
    {
        id = user.PersonId
    });
}
```

- **Secure** : Les cookies sont également configurés avec l'attribut *Secure*, ce qui garantit qu'ils ne sont envoyés que sur des connexions HTTPS, assurant que les données transmises sont chiffrées (Norme RFC 6265).
- **SameSite** : Nous utilisons l'attribut *SameSite* (Norme RFC 6265) pour nos cookies, ce qui limite leur envoi aux requêtes provenant du même site, protégeant ainsi contre les attaques CSRF (Cross-Site Request Forgery).

### Gestion des Tokens JWT :

- Création d'Access Tokens :** Nous avons développé une méthode `GenerateJwtToken` appelée par le contrôleur pour créer des jetons JWT signés. Ces jetons sont utilisés pour authentifier les utilisatrices de manière sécurisée (Norme RFC 7519).

```
3 references
public async Task<string> GenerateJwtToken(ApplicationUser user)
{
    var claims = new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, user.Id),
        new Claim(JwtRegisteredClaimNames.Email, user.Email),
        new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
    };

    var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]));
    var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
    var expires = DateTime.UtcNow.AddMinutes(1);

    var token = new JwtSecurityToken(
        issuer: _configuration["Jwt:Issuer"],
        audience: _configuration["Jwt:Audience"],
        claims: claims,
        expires: expires,
        signingCredentials: creds
    );

    return await Task.FromResult(new JwtSecurityTokenHandler().WriteToken(token));
}
```

- Expiration des Tokens :** L'expiration des jetons d'accès est gérée automatiquement par ASP.NET Identity.
- Tokens de Rafraîchissement :** La méthode `GenerateRefreshToken`, également appelée par le contrôleur, génère des jetons de rafraîchissement sécurisés et aléatoires avec une date d'expiration. Ces jetons de rafraîchissement permettent aux utilisatrices de renouveler leurs jetons d'accès sans nécessiter une nouvelle authentification.

```
3 references
public Task<(string, DateTime)> GenerateRefreshToken()
{
    var randomBytes = new byte[32];
    using (var rng = RandomNumberGenerator.Create())
    {
        rng.GetBytes(randomBytes);
        var refreshToken = Convert.ToBase64String(randomBytes);
        var refreshTokenExpiryTime = DateTime.UtcNow.AddMinutes(2);
        return Task.FromResult((refreshToken, refreshTokenExpiryTime));
    }
}
```

- Stockage des Tokens :** Les jetons d'accès et de rafraîchissement sont stockés de manière sécurisée dans la base de données, spécifiquement dans la table `AspNetUsers`.
- Durée de Vie des Tokens :** Nous avons configuré une durée de vie plus courte pour les jetons d'accès par rapport aux jetons de rafraîchissement. Les jetons de rafraîchissement permettent d'obtenir de nouveaux jetons d'accès lorsque ceux-ci expirent, réduisant ainsi la fréquence des authentifications nécessaires.
- Envoi Automatique des Cookies :** Lors des requêtes suivantes, les cookies stockés sont automatiquement envoyés avec chaque requête HTTP vers le serveur, facilitant ainsi la gestion des sessions utilisatrices de manière sécurisée sans intervention du front-end.

```
[Authorize]
[Route("api/[controller]")]
[ApiController]
```

- Protection des Routes :** Nous avons sécurisé les routes sensibles en utilisant l'attribut `[Authorize]`, garantissant que seules les utilisatrices authentifiées peuvent y accéder. Cela renforce la protection de nos API en s'assurant que les opérations critiques sont restreintes aux utilisatrices autorisées.

## Configuration mot de passe

```
// Configuring identity options
builder.Services.Configure<IdentityOptions>(options =>
{
    // User settings
    options.User.RequireUniqueEmail = true;
    options.User.AllowedUserNameCharacters =
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_@";
    
    // Password settings
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequiredLength = 8;

    // Locks settings
    options.Lockout.MaxFailedAccessAttempts = 3;
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
});

```

Lors de la création d'un mot de passe, nous avons configuré des règles de sécurité, en prenant en compte les recommandations de la CNIL.

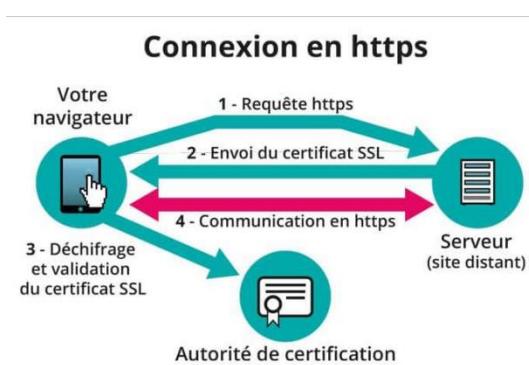
## **5.4.2. Sécurité**

La sécurité de DERBEEZ est une priorité pour garantir la protection des données des utilisatrices et la stabilité du système. Nous avons mis en place diverses mesures de sécurité pour parer aux principales attaques rencontrées sur les applications web.

### Principales Attaques et Mesures de Protection

- **Injection SQL :**
  - **Description :** Insertion de code malveillant dans des champs de saisie utilisateur pour manipuler la base de données.
  - **Mesures de Protection :** Utilisation de l'ORM Entity Framework pour paramétrier automatiquement les requêtes SQL et empêcher les injections.
- **Cross-Site Scripting (XSS) :**
  - **Description :** Insertion de scripts malveillants dans des pages web pour voler des données sensibles.
  - **Mesures de Protection :** Validation de schéma et règles de format sur les formulaires en ReactJs pour filtrer et échapper les entrées utilisateur.
- **Cross-Site Request Forgery (CSRF) :**
  - **Description :** Forçage d'une action non désirée sur une session authentifiée.
  - **Mesures de Protection :** Utilisation de cookies avec l'attribut HttpOnly pour éviter le vol des cookies via JavaScript, et authentification par jetons pour sécuriser les requêtes.
- **Force Brute :**
  - **Description :** Utilisation de programmes automatisés pour essayer différentes combinaisons de noms d'utilisateurs et de mots de passe.
  - **Mesures de Protection :** Implémentation de mécanismes de détection et de blocage des tentatives répétées de connexion. Une autre mesure est aussi prévue, l'ajout de captchas.

## HTTPS



Nous avons mis en place le protocole HTTPS (Hypertext Transfer Protocol Secure) pour assurer la sécurisation des données transmises entre le navigateur et le serveur web. HTTPS utilise une couche de chiffrement SSL/TLS (Secure Sockets Layer / Transport Layer Security), garantissant que même si les données sont interceptées par un tiers, elles ne peuvent pas être lues ou modifiées sans la clé de déchiffrement.

Pour le développement nous utilisons un certificat auto-signé mais pour la production nous avons prévu d'utiliser des fournisseurs comme Let's Encrypt.

### Variables d'Environnement

Pour protéger les données sensibles nécessaires au fonctionnement de l'application, telles que les URL de connexion à la base de données, les identifiants API, les clés secrètes et les ports :

- **Front-end** : Les variables d'environnement sont stockées dans un fichier `.env` qui n'est pas versionné et est ajouté dans le `.gitignore`.
- **Back-end** : Les données sensibles sont regroupées dans le fichier `appsettings.json`.

Dans le cadre d'un déploiement l'utilisation de solutions comme Azure Vault permet une plus grande sécurisation de ces informations.

### **5.4.3.Données Personnelles**

Conformément au RGPD, nous avons mis en place des mesures pour protéger les données personnelles des utilisatrices et garantir leurs droits en tant que collectrices et processeuses de données.

Nous avons mis à disposition les mentions légales obligatoires, telles que l'identité de l'éditeur du site, l'hébergeur et les conditions d'utilisation. Les modalités du RGPD sont également accessibles, visant à protéger les données personnelles et garantir le droit de consultation, modification et suppression des données. Ces éléments contribuent à la sécurité juridique de notre application.

Conformément aux recommandations de la CNIL, les utilisatrices peuvent demander :

- L'accès à leurs données personnelles
- La rectification de leurs données
- La suppression de tout ou partie de leurs données personnelles

Ces demandes peuvent être initialement effectuées par e-mail. Nous prévoyons également d'ajouter un bouton de suppression du compte pour faciliter cette démarche.

## 5.4.4. Tests

L'intégration de tests dans le développement permet de garantir :

- La qualité du code
- La maintenabilité / non-régression
- Le bon fonctionnement des fonctionnalités

Il existe différents types de tests, chacun ayant un objectif différent :

- **Les tests unitaires** : Ce sont des tests qui vérifient le fonctionnement d'une petite partie du code, d'une fonction ou d'une méthode.
- **Les tests d'intégrations** : Ils vérifient comment les différentes parties du code interagissent entre elles. Ils sont utilisés pour détecter des problèmes d'intégration et de communication entre les composants.
- **Les tests fonctionnels** : Les tests fonctionnels sont conçus pour vérifier que chaque fonctionnalité du système fonctionne correctement en entrant des données d'entrée appropriées et en vérifiant les résultats attendus.
- **Les tests de bout en bout** : Les tests de bout en bout reproduisent le comportement d'un utilisateur avec le logiciel dans un environnement applicatif complet.

Voici un exemple d'un fichier de test unitaire que j'ai écrit qui vérifie le bon fonctionnement de `api.ts` avec Jest:

```

① 6  describe('API module', () => {
    7    let mock: MockAdapter;
    8
    9    beforeEach(() => {
10      mock = new MockAdapter(axios);
11    });
12
13    afterEach(() => {
14      mock.reset();
15    });
16
17    it('should handle response errors with a generic error message when no message is provided', async () => {
18      mock.onGet('/test-endpoint').reply(400);
19
20      try {
21        await api.get('/test-endpoint');
22      } catch (error) {
23        expect(error).toBe(t('error.generic'));
24      }
25    });
26
27    it('should handle no response error', async () => {
28      mock.onGet('/test-endpoint').networkError();
29
30      try {
31        await api.get('/test-endpoint');
32      } catch (error) {
33        expect(error).toBe(t('error.generic'));
34      }
35    });
36
37    it('should handle request setup errors', async () => {
38      const mockApi = axios.create();
39      mockApi.interceptors.request.use(() => {
40        throw new Error('Request setup error');
41      });
42
43      try {
44        await mockApi.get('/test-endpoint');
45      } catch (error) {
46        const err = error as Error;
47        expect(err.message).toBe('Request setup error');
48      }
49    });
50  });
51

```

PASS src/tests/core/api/api.test.tsx
 API module
 ✓ should handle response errors with a generic error message when no message is provided (276 ms)
 ✓ should handle no response error (18 ms)
 ✓ should handle request setup errors (79 ms)

 Test Suites: 1 passed, 1 total
 Tests: 3 passed, 3 total
 Snapshots: 0 total
 Time: 2.284 s

Voici un exemple d'un fichier de test unitaire que j'ai écrit qui vérifie le bon fonctionnement de `ImageService.cs` avec NUnit:

```
[TestFixture]
0 references
public class ImageServiceTests
{
    private Mock<IImageRepository> _imageRepositoryMock;
    private ImageService _imageService;

    [SetUp]
0 references
    public void SetUp()
    {
        _imageRepositoryMock = new Mock<IImageRepository>();
        _imageService = new ImageService(_imageRepositoryMock.Object);
    }

    [Test]
0 | 0 references
    public async Task SaveImageAsync_ShouldSaveImageAndReturnImage()
    {
        // Arrange
        var base64Image = "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAUAAAABCIAAAACUJ/gA";
        var imageBytes = Convert.FromBase64String(base64Image.Split(',')[1]);
        var image = new Image
        {
            Content = imageBytes,
            Info = new ImageInfo
            {
                DataType = "image/png",
                Size = imageBytes.Length,
                Timestamp = DateTime.UtcNow
            }
        };

        _imageRepositoryMock.Setup(repo => repo.AddAsync(It.IsAny<Image>()))
            .Returns(Task.CompletedTask)
            .Callback<Image>(i => i.Id = 1);

        // Act
        var result = await _imageService.SaveImageAsync(base64Image);

        // Assert
        Assert.That(result, Is.Not.Null);
        Assert.Multiple(() =>
        {
            Assert.That(result.Content, Is.EqualTo(imageBytes));
            Assert.That(result.Id, Is.EqualTo(1));
        });
    }

    [Test]
0 | 0 references
    public async Task DeleteImageAsync_ShouldDeleteImage_WhenImageExists()
    {
        // Arrange
        var image = new Image { Id = 1, Content = [], Info = new ImageInfo{DataType = "image/png"} };

        _imageRepositoryMock.Setup(repo => repo.GetByIdAsync(1))
            .>ReturnsAsync(image);

        _imageRepositoryMock.Setup(repo => repo.DeleteAsync(image))
            .>Returns(Task.CompletedTask);

        // Act
        await _imageService.DeleteImageAsync(1);

        // Assert
        _imageRepositoryMock.Verify(repo => repo.DeleteAsync(image), Times.Once);
    }
}
```

Dans les deux cas on peut voir que la structure reste la même, Arrange, Act, Assert et que les tests ont pour but de vérifier les cas où tout se passe bien, mais aussi les cas où il y a une erreur pour être sûr que tous les cas soient prévus.

- **Arrange :**
  - Prépare les objets et les données nécessaires pour le test.
  - Initialise les objets à tester et configure l'environnement.
- **Act :**
  - Exécute la méthode ou la fonction à tester avec les données préparées.
- **Assert :**
  - Vérifie que les résultats obtenus correspondent aux résultats attendus.
  - Utilise des assertions pour valider les comportements

## 5.5.Déploiement

### 5.5.1.IIS

IIS (Internet Information Services) est un serveur web développé par Microsoft, il est utilisé principalement pour héberger des applications web sur les systèmes d'exploitation Windows.

Comme Nginx dans l'écosystème Linux, IIS fournit beaucoup d'outils servant aux applications web (gérer des requêtes HTTP, support de SSL/TLS, l'équilibrage de charge). Grâce à son intégration étroite avec d'autres produits Microsoft, notamment ASP.NET, IIS est le bon choix pour DERBEEZ en développement ainsi qu'en production.

Je me suis chargée de la mise en place de la configuration IIS pour notre projet, pour ce faire j'ai créé un script PowerShell qui automatise la tâche. Je l'ai créé en gardant en tête plusieurs points :

- Il doit être transparent, j'ai beaucoup commenté les actions du script. J'ai aussi écrit une documentation pour l'expliquer.
- Il doit être optionnel, j'ai ajouté à la documentation toutes les étapes nécessaires pour reproduire la configuration sans le script.
- Il doit être sécurisé, le script analyse les ports notés dans le .env dédié et affiche leurs utilisations (ou leurs absences) et demande confirmation pour utiliser ces ports.
- Il doit être proche d'un environnement de production pour éviter des différences de fonctionnement, qui rendent le débogage parfois complexe. Pour ça avec IIS on peut utiliser des certificats auto-signés qui nous permettent d'utiliser HTTPS pour nos communications.

```

2  param (
3      [String]$frontEndName = "derbeez-web",
4      [String]$backEndName = "derbeez-web-back-end",
5      [String]$frontEndModuleName = "derbeez-web-pool",
6      [String]$backEndModuleName = "derbeez-appl-pool"
7  )
8
9  # Import module administration module
10 # Import-Module WebAdministration
11
12 Function CleanEnv {
13
14     # Function to clear environment variables
15
16     # Function to load environment variables from .env file
17     Function LoadEnv {
18
19         # Load environment variables
20         $envPath = ".\..\derbeez-web\.env"
21         If (-Not (Test-Path $envPath)) {
22             Load-Env -envPath $envPath
23
24         # Retrieve ports from environment variables
25         $env:FrontendHttpsPort = [System.Environment]::GetEnvironmentVariable("FRONT_HTTPS")
26         $env:BackendHttpsPort = [System.Environment]::GetEnvironmentVariable("BACK_HTTPS")
27         $env:FrontendHttpPort = [System.Environment]::GetEnvironmentVariable("FRONT_HTTP")
28         $env:BackendHttpPort = [System.Environment]::GetEnvironmentVariable("BACK_HTTP")
29
30         # Logging messages to check the values
31         Write-Output "FRONTEND HTTPS PORT: $env:FrontendHttpsPort"
32         Write-Output "BACKEND HTTPS PORT: $env:BackendHttpsPort"
33         Write-Output "FRONTEND HTTP PORT: $env:FrontendHttpPort"
34         Write-Output "BACKEND HTTP PORT: $env:BackendHttpPort"
35
36         # Check if ports are correctly set
37         If ($env:FrontendHttpsPort -or -not $FrontendHttpsPort -or -not $BackendHttpsPort) {
38
39             # Function to check existing SSL bindings
40             Function CheckExistingSSLBinding {
41
42                 # Ports to check
43                 $portsToCheck = @($FrontendHttpsPort, $BackendHttpsPort)
44
45                 # Check existing SSL bindings
46                 $existingBindings = Get-ExistingSSLBinding -ports $portsToCheck
47
48                 # Check existing SSL bindings
49                 $existingBindings = Get-ExistingSSLBinding -ports $portsToCheck
50
51                 # If no https binding exist, we will create and recreate the https bindings on the following ports: $existingBindings
52                 If(-Not ($existingBindings -contains "https")) {
53                     Write-Output "No https binding exist, we will create and recreate the https bindings on the following ports: $existingBindings"
54
55                     $existingBindings | % {
56                         Write-Output "Creating https binding for port: $_"
57                         Set-SSLBinding -port $_ -Protocol "https" -thumbprint $certThumbprint
58
59                         Write-Output "-----"
60
61                     } #foreach
62
63                     confirmation = Read-Host "Do you want to continue? (y/n)"
64
65                     If ($confirmation -eq 'y') {
66
67                         # Relative paths
68                         $relativePaths = Get-RelativePath -path $certThumbprint
69                         $frontEndRelativePath = ".\..\derbeez-web\build"
70                         $backEndRelativePath = ".\..\derbeez-appl\Derbeez\bin\Deployment\inet0\0"
71
72                         # Resolve relative paths to absolute paths
73                         $frontEndPhysicalPath = (Resolve-Path -Path $frontEnd -ChildPath $frontEndRelativePath).Path
74                         $backEndPhysicalPath = (Resolve-Path -Path $backEnd -ChildPath $backEndRelativePath).Path
75
76                     } #if
77
78                 } #if
79
80             } #function
81
82             # Function to get the certificate thumbprint and subject
83             Function Get-CertThumbprintAndSubject {
84
85                 # Function to remove existing SSL bindings with switch
86                 Function Remove-ExistingSSLBinding {
87
88                     # Function to get the SSL binding with netsh
89                     Function Get-SSLBinding {
90                         Get-Netsh Web Get-SSLBinding
91
92                         # Function to remove existing ssl bindings
93                         Function Remove-ExistingSSLBinding {
94
95                             # Get certificates thumbprint and subject
96                             $certThumbprint = Get-Netsh Web Get-SSLBinding | Select-Object thumbprint
97                             $certSubject = $certThumbprint.subject
98
99                             Write-Output "Deleting certificate: $certSubject with thumbprint $certThumbprint"
100
101                             # If cert is physical port exist
102                             If (-Not (Test-Path -Path $certThumbprint)) {
103                                 Get-Netsh Web Get-SSLBinding | Select-Object thumbprint
104
105                                 # remove and recreate application pool if necessary
106                                 Write-Output "Configurable application pool removed"
107                                 Set-ApplicationPool -name $certSubject -recycleAction SilentlyContinue" {
108
109                                 # Set Application Pool
110                                 Write-Output "Application pool $certSubject created."
111
112                                 Get-Netsh Web Get-SSLBinding -Name $certSubject
113
114                                 # Configure application pool
115                                 Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
116                                 Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
117
118                                 # Remove and recreate website if necessary
119                                 Get-Netsh Web Get-Website -Name $certSubject
120
121                                 # If website exist
122                                 If (-Not (Test-Path -Path $certThumbprint)) {
123                                     Get-Netsh Web Get-SSLBinding -Name $certSubject
124
125                                     # Configure application pool
126                                     Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
127                                     Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
128
129                                     # Remove and recreate website if necessary
130                                     Get-Netsh Web Get-Website -Name $certSubject
131
132                                     # If website exist
133                                     If (-Not (Test-Path -Path $certThumbprint)) {
134                                         Get-Netsh Web Get-SSLBinding -Name $certSubject
135                                         Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
136                                         Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
137
138                                         # Remove and recreate website if necessary
139                                         Get-Netsh Web Get-Website -Name $certSubject
140
141                                         # If website exist
142                                         If (-Not (Test-Path -Path $certThumbprint)) {
143                                             Get-Netsh Web Get-SSLBinding -Name $certSubject
144                                             Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
145                                             Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
146
147                                             # Remove and recreate website if necessary
148                                             Get-Netsh Web Get-Website -Name $certSubject
149
150                                             # If website exist
151                                             If (-Not (Test-Path -Path $certThumbprint)) {
152                                                 Get-Netsh Web Get-SSLBinding -Name $certSubject
153                                                 Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
154                                                 Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
155
156                                                 # Remove and recreate website if necessary
157                                                 Get-Netsh Web Get-Website -Name $certSubject
158
159                                                 # If website exist
160                                                 If (-Not (Test-Path -Path $certThumbprint)) {
161                                                     Get-Netsh Web Get-SSLBinding -Name $certSubject
162                                                     Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
163                                                     Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
164
165                                                     # Remove and recreate website if necessary
166                                                     Get-Netsh Web Get-Website -Name $certSubject
167
168                                                     # If website exist
169                                                     If (-Not (Test-Path -Path $certThumbprint)) {
170                                                         Get-Netsh Web Get-SSLBinding -Name $certSubject
171                                                         Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
172                                                         Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
173
174                                                         # Remove and recreate website if necessary
175                                                         Get-Netsh Web Get-Website -Name $certSubject
176
177                                                         # If website exist
178                                                         If (-Not (Test-Path -Path $certThumbprint)) {
179                                                             Get-Netsh Web Get-SSLBinding -Name $certSubject
180                                                             Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
181                                                             Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
182
183                                                             # Remove and recreate website if necessary
184                                                             Get-Netsh Web Get-Website -Name $certSubject
185
186                                                             # If website exist
187                                                             If (-Not (Test-Path -Path $certThumbprint)) {
188                                                                 Get-Netsh Web Get-SSLBinding -Name $certSubject
189                                                                 Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedMachineVersion" -Value "v4.0"
190                                                                 Set-SSLBinding -Name $certSubject -thumbprint $certThumbprint -Name "managedRuntimeVersion" -Value "v4.0"
191
192                                                                 # Remove and recreate website if necessary
193                                                                 Get-Netsh Web Get-Website -Name $certSubject
194
195                                                                 # If website exist
196                                                                 If (-Not (Test-Path -Path $certThumbprint)) {
197
198 }
```

Une configuration IIS peut être créée pour un environnement de production.

## 5.5.2.Docker

Je me suis chargée de la containerisation de notre application avec Docker. Cette installation a pour but de simplifier la collaboration grâce à un environnement isolé et aussi la mise en production en adaptant la configuration à un déploiement sécurisé. J'ai souhaité laisser le choix de l'utilisation de IIS ou Docker pour l'environnement de développement.

Les étapes ont été :

```

1  services:
2    backend:
3      container_name: derbeez-api
4      build:
5        context: ../../derbeez-api
6        dockerfile: Dockerfile
7      ports:
8        - "5000:5000"
9      environment:
10     - ASPNETCORE_ENVIRONMENT=Development
11     - DB_HOST=${DB_HOST}
12     - DB_PORT=${DB_PORT}
13     - DB_USER=${DB_USER}
14     - DB_PASSWORD=${DB_PASSWORD}
15     - DB_NAME=${DB_NAME}
16     - ALLOWED_ORIGINS=${ALLOWED_ORIGINS}
17     - IS_DOCKER=true
18   volumes:
19     - type: bind
20       source: ../../derbeez-api/Derbeez.Endpoints/appsettings.json
21       target: /app/appsettings.json
22
23   frontend:
24     container_name: derbeez-web
25     build:
26       context: ../../derbeez-web
27       dockerfile: Dockerfile
28     args:
29       REACT_APP_API_URL_DOCKER: ${REACT_APP_API_URL_DOCKER}
30     ports:
31       - "3000:3000"
32     environment:
33       - IS_DOCKER=true
34     depends_on:
35       - backend
36
37   volumes:
38     - postgres_data:

```

- Création du fichier *docker-compose.yml*, utilisé par Docker Compose pour gérer des applications multi-conteneurs. On y déclare les différents services à conteneuriser, le back-end et le front-end, leurs ports ainsi que leurs variables d'environnement.

```

1  FROM node:lts AS build
2
3  WORKDIR /app
4
5  # Install dependencies
6  COPY package.json package-lock.json .
7  RUN npm install
8
9  # Copy the rest of the application code
10 COPY . .
11
12 # Set environment variables
13 ARG REACT_APP_API_URL_DOCKER
14 ENV REACT_APP_API_URL_DOCKER ${REACT_APP_API_URL_DOCKER}
15
16 RUN npm run build
17
18 FROM nginx:alpine
19
20 # Copy config nginx
21 COPY ./nginx/nginx.conf /etc/nginx/conf.d/default.conf
22
23 WORKDIR /usr/share/nginx/html
24
25 # Remove default nginx static assets
26 RUN rm -rf ./*
27
28 # Copy static assets from builder stage
29 COPY --from=build /app/build .
30
31 # Containers run nginx with global directives and daemon off
32 ENTRYPOINT ["nginx", "-g", "daemon off;"]
33

```

- Création du fichier Dockerfile dans le projet front-end et la configuration d'un serveur web spécifique pour un OS Linux, Nginx.

```

1 # Use the official .NET SDK 8.0 image for building
2 FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
3 # Set the working directory inside the container
4 WORKDIR /src
5
6 # Copy the csproj files and restore dependencies
7 COPY ["Derbeez.Core/Derbeez.Core.csproj", "Derbeez.Core/"]
8 COPY ["Derbeez.Endpoints/Derbeez.Endpoints.csproj", "Derbeez.Endpoints/"]
9 COPY ["Derbeez.Infra/Derbeez.Infra.csproj", "Derbeez.Infra/"]
10
11 # Restore the dependencies specified in the csproj file
12 RUN dotnet restore "Derbeez.Endpoints/Derbeez.Endpoints.csproj"
13
14 # Copy all the content from the current directory to the container
15 COPY . .
16
17 # Set the working directory to the main project directory
18 WORKDIR /src/Derbeez.Endpoints
19 # Build the project in Release mode
20 RUN dotnet build "Derbeez.Endpoints.csproj" -c Release -o /app/build
21
22 # Publish stage: use the build image
23 FROM build AS publish
24 # Publish the application in Release mode without restoring dependencies
25 RUN dotnet publish --no-restore -c Release -o /app/publish
26
27 # Use the official ASP.NET Core 8.0 runtime image
28 FROM mcr.microsoft.com/dotnet/aspnet:8.0
29
30 # Set the ASP.NET Core URLs environment variable
31 ENV ASPNETCORE_URLS=http://+:5000
32
33 # Expose port 5000
34 EXPOSE 5000
35
36 # Set the working directory for the application
37 WORKDIR /app
38
39 # Copy the published files from the publish stage
40 COPY --from=publish /app/publish .
41
42 # Set the entry point to run the application
43 ENTRYPOINT ["dotnet", "Derbeez.Endpoints.dll"]

```

- Création du fichier Dockerfile dans le projet back-end.

- Création de fichiers d'environnements spécifiques pour IIS & Docker.
- Rendre les deux configurations viables en parallèle.

J'ai aussi créé une documentation pour expliquer les étapes nécessaires à la mise en place de docker, avec notamment un rappel des principales commandes.

### **5.5.3.Documentation**

Comme évoqué précédemment j'ai écrit plusieurs documentations sur le projet DERBEEZ. Pour éviter de chercher dans quelle partie du projet est telle documentation je crée un nouveau dossier versionné *Derbeez-Development*. Ce projet réunit les scripts et leurs documentations :

ApiTroubleShootingBasics.md WebTroubleShootingBasics.md  AutoMapper.md EntityFramework.md IdentityFramework.md Microsoft.NET.Test.Sdk.md Moq.md Newtonsoft.Json.md NUnit.md Serilog.md Swashbuckle.AspNetCore.md	<pre><code># Documentation de base pour lancer le projet Derbees  # Prérequis Assurez-vous d'avoir les éléments suivants installés sur votre système : Api :   - .NET Core SDK (version requise par le projet)   - Visual Studio ou Visual Studio Code avec les extensions :     - C#     - La charge de travail "Développement ASP.NET et web".   - L'extension .NET pour Visual Studio Code avec la charge de travail "Développement ASP.NET et web".   - L'option ASP.NET et développement web sous la charge de travail "Développement ASP.NET et web"  Web :   - Visual Studio Code   - Node   - L'extension Prettier sur Visual Studio Code et de configurer le formatOnSave.  Environnements :   - PostgreSQL     - pgAdmin (Ou autre outil de gestion de base de données PostgreSQL)     - IIS (Internet Information Services) voir Configuration d'IIS  Docker :   - Docker : Assurez-vous que Docker est installé sur votre machine.   - Docker Compose : Docker Compose doit également être installé. Il est généralement inclus avec Docker Desktop.  # Récupération du projet 1. Clonez les dépôts des projets back et front et development à partir de github. 2. Pour les scripts les chemins doivent être (sinon modifier InitialIIS.ps1) :    \derbees-api    \derbees-web    \derbees-development  # Initialisation du projet Api : 1. Restarez les dépendances du projet en exécutant la commande 'dotnet restore' dans le répertoire du projet. (Normalement lancé par le build) 2. Vous pouvez aussi ouvrir le fichier solution (.sln) et restaurer les packages NuGet manquants en cliquant sur le bouton "Restaurer les dépendances". 3. Build la solution  Web : 1. Restarez les dépendances du projet en exécutant la commande 'npm i' dans le répertoire du projet. 2. Créer le build avec npm run build  # Configurations Environnements Deux type de configuration entre le web et l'api est disponible, IIS et docker. les deux peuvent être utilisés en même temps.  ## IIS Suivre la documentation derbees-development/IIS/README.md  ## Docker Suivre la documentation derbees-development/Docker/README.md  # BD 1. Installez et lancez pgAdmin. 2. Connectez-vous à votre serveur PostgreSQL local en utilisant les informations d'identification appropriées. 3. Normalement la migration va créer la base de données en jouant la première migration au cas où :   - Crée une nouvelle base de données pour votre projet.   - Nom d'hôte : l'adresse IP ou le nom d'hôte du serveur PostgreSQL.   - Port : 5432 (le port par défaut pour PostgreSQL).   - Base de données : le nom de la base de données à laquelle vous souhaitez vous connecter.   - Nom d'utilisateur : le nom d'utilisateur PostgreSQL avec les autorisations appropriées.   - Mot de passe : le mot de passe associé au nom d'utilisateur. 4. Ouvrez une invite de commande dans le répertoire Derbees/Test/Migration et exécutez les commandes suivantes pour appliquer les migrations EF à la base de données.   - 'dotnet ef database update' (pour toutes les migrations en attente)   - ou '.Add-Migration -Name "v1.0.0"' (pour appliquer une migration spécifique)</code></pre>
---	---

## 5.5.4. Intégration continue

L'intégration continue vise à automatiser la vérification et la validation du code source de l'application. L'objectif est de détecter les erreurs et problèmes de qualité le plus tôt possible.

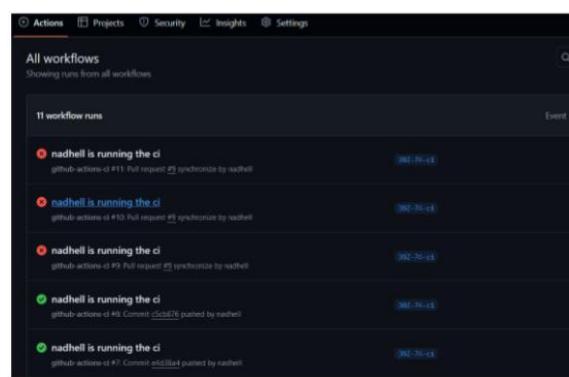
Nous avons utilisé Github Actions qui est un outil d'automatisation qui permet de créer des workflows personnalisés qui peuvent automatiser les builds, les tests et le déploiement du code.

Les workflows sont définis dans un fichier YAML. Il décrit les étapes à suivre pour exécuter les tâches et les conditions sous lesquelles elles doivent être exécutées.

Le processus :

- À chaque fois qu'une modification de code est poussée sur la branche develop et main, le système de CI détecte automatiquement les changements.
- Github Actions exécute alors automatiquement une série de tests pour s'assurer que les modifications ne causent pas de régressions.
- Les résultats des tests sont générés et nous sommes informés en cas d'échecs.

```
1 name: github-actions-ci
2 run-name: ${{ github.actor }} is running the ci
3 on:
4   pull_request:
5     branches: ["main", "develop"]
6     pull_request_review:
7       types: [edited, dismissed]
8   jobs:
9     check-bats-version:
10       name: build-and-test
11       runs-on: windows-latest
12       steps:
13         - uses: actions/checkout@v4
14         - uses: actions/setup-node@v4
15         - with:
16           node-version: "latest"
17         - run: npm install
18         - run: npm run build
19         - run: npm test
20
```



## 5.5.5.Déploiement

Le déploiement de notre application se fera via Azure Web App for Containers, une solution parfaitement adaptée à notre environnement majoritairement Microsoft.

Nous utiliserons GitHub Actions, une plateforme d'intégration continue (CI) soutenue par Microsoft, pour automatiser notre processus de déploiement. Cette action spécifique sera configurée pour construire un conteneur Docker à partir de nos fichiers Docker existants. Une fois le conteneur construit, il sera automatiquement publié sur Azure, la plateforme cloud de Microsoft. Ce processus permettra de s'assurer que notre application est toujours à jour et accessible sans intervention manuelle.

Avec Azure il est possible de choisir où résident nos données, assurant que toutes les données restent au sein de l'Union européenne pour garantir la conformité avec le RGPD.

**Résidence des données**  
Téléchargez le graphique sous « Sélectionner votre zone géographique » pour plus d'informations.

**Conformité**  
Découvrez des informations sur les [offres de conformité régionales applicables à l'échelle mondiale](#).

**Produits disponibles**  
Pour en savoir plus sur la disponibilité des produits, visitez [Produits disponibles par région](#).

# 6.Réalisation personnelle

## 6.1.Spécifications

Ayant déjà présenté plusieurs de mes réalisations, j'ai choisi pour ce passage de présenter la création de la page profil que j'ai développée pour les utilisatrices de DERBEEZ car c'est un ensemble de fonctionnalités full-stack.

Le but de cette page est de :

- Visualiser ses posts communautaires.
- Pouvoir les trier selon plusieurs critères.
- Avoir une pagination des posts pour éviter les requêtes inutiles.
- Modifier les informations de son profil :
  - Avatar
  - Pseudo, pronoms, biographie

Les besoins techniques sont :

- Ne pas pouvoir modifier un profil qui n'est pas le sien.
- Une gestion de l'état de la page correct selon si c'est mon profil et si je suis en édition.
- Des formulaires et validations robustes.
- La mise à jour du profil après enregistrement.
- Des composants réutilisables.
- Téléchargement et validation des Fichiers.
- Internationalisation des textes avec i18next.
- Accessibilités.
- Intégration avec le Store pour la mise à jour.
- Bonne responsivité et mise en Page.

## 6.2.Réalisation

Stores :

```
fetchPersonDetails: async personId => {
  set({ isLoading: true, error: null });
  try {
    const response = await getPerson(personId);
    if (response.isSuccess) {
      set({ personDetails: response.data });
    } else {
      set({ error: response.errorMessage });
    }
  } catch (error) {
    set({ error: error as string });
  } finally {
    set({ isLoading: false });
  }
},
```

Je modifie d'abord le *AuthStore.ts* que j'avais créé pour gérer l'état de l'authentification de l'utilisatrice. Maintenant, en cas de connexion réussie, on enchaîne sur un autre appel, *fetchPersonDetails* qui va aller chercher les informations métier de l'utilisatrice. J'ajoute aussi l'appel à la suppression des informations en cas de déconnection.

Je ne vais pas trop détailler le travail fait sur l'API pour la fonction *fetchPersonDetails* car il est assez simple. Je détaillerais mon travail sur le back-end un peu plus bas sur d'autres cas.

Ensuite je crée un store *ProfileStore.ts* pour gérer l'état des informations de l'utilisatrice. J'implémente la fonction *fetchPersonDetails* et je conserve les informations reçues dans la session storage car ces informations sont très souvent utilisées à travers le site.

### Page Profile :

Je crée la page Profile, la référence dans le routage en utilisant le composant `<PrivateRoute />` pour s'assurer que la page est accessible uniquement à une utilisatrice connectée.

```
const Profile = () => {
  const navigate = useNavigate();
  //Requested id
  const { id } = useParams<{ id: string }>();
  //AuthenticatedId
  const { personId, error, isLoading } = usePersonAuthStore();
  const { personDetails } = usePersonProfileStore();
  const [requestedPersonDetails, setRequestedPersonDetails] =
    useState<PersonDetails | null>(null);
  const [isMyProfile, setIsMyProfile] = useState(false);

  useEffect(() => {
    const fetchData = async () => {
      if (!id) {
        return navigate(paths.error);
      }

      if (personId === +id) {
        setRequestedPersonDetails(personDetails);
        setIsMyProfile(true);
        return;
      }

      const response = await getPerson(+id);
      if (response.isSuccess) {
        setRequestedPersonDetails(response.data);
      } else {
        return navigate(paths.error);
      }
    };

    fetchData();
  }, [navigate, id, personId, personDetails]);

  if (error) {
    navigate(paths.error);
  }
}
```

Cette page fonctionne ainsi : si j'accède à ma propre page, les informations nécessaires sont déjà connues et je ne relance pas d'appels API, sinon je fais l'appel. Je précise aux composants enfants si c'est ma page pour que les boutons liés à la modification de la page soient disponibles.

### Composant *GroupUserPost*:

Ce composant contient l'appel à *getPostsSummaryByFilter* chargé de récupérer les posts de l'utilisatrice selon plusieurs paramètres, *skip* et *nbToGet* pour gérer la pagination, *isAsc* et *OrderBy* pour gérer le filtre et l'ordre.

Une fois les posts récupérés je vais appeler le simple composant d'affichage < *PostCardSummary*> pour chaque objet de ma réponse.

```

const [filter, setFilter] = useState<PostsFilter>({
  personId: personId,
  nbToGet: postsPerPage,
  skip: 0,
  orderBy: OrderByChoices.DATE,
  isAsc: true,
  isSummary: true,
});

const handleFilterChange = (newFilter: PostsFilter) => {
  setFilter(newFilter);
  setCurrentPage(1);
};

const fetchPosts = async (page: number) => {
  setLoading(true);
  const skip = (page - 1) * postsPerPage;
  const postsFilter: PostsFilter = { ...filter, skip: skip };

  try {
    const response = await getPostsSummaryByFilter(postsFilter);
    if (response.isSuccess) {
      setPosts(response.data);
    } else {
      setError(response.errorMessage);
    }
  } catch (error) {
    setError(error as string);
  } finally {
    setLoading(false);
  }
};

useEffect(() => {
  fetchPosts(currentPage);
}, [currentPage, filter]);

const paginate = (page: number) => setCurrentPage(page);

return (
  <Card className='flex flex-col w-full md:px-6 border-2 border-card-border bg-group-grey'>
    <CardHeader className='flex-col min-[450px]:flex-row justify-between items-center pb-1'>
      <CardTitle className='text-header'>{t('profile.posts')}</CardTitle>
      <FilterOrderGroupPost
        initialFilter={filter}
        onFilterChange={handleFilterChange}
      />
    </CardHeader>
    <CardContent className='flex justify-center xl:justify-around flex-wrap mt-5 gap-5 md:gap-3'>
      {loading ? (
        <LoadingSpinner size={100} />
      ) : error ? (
        <InfoMessage message={error} />
      ) : (
        posts.map((post: PostSummary, index) => (
          <PostCardSummary
            key={index}
            title={post.title}
            date={post.date}
            content={post.content}
          />
        ))
      )}
    </CardContent>
    <CardFooter className='flex justify-center p-0 mt-auto mb-4'>
      <Button
        onClick={() => paginate(currentPage - 1)}
        disabled={currentPage === 1 || loading}
        variant='ghost'
        title={t('paginate.previous')}
        className={`${currentPage === 1 || loading ? 'opacity-50 cursor-not-allowed' : ''}`}
      />
      <Button
        onClick={() => paginate(currentPage + 1)}
        disabled={loading || posts.length < postsPerPage}
        variant='primary'
        title={t('paginate.next')}
        className={`${loading || posts.length < postsPerPage ? 'opacity-50 cursor-not-allowed' : ''}`}
      />
    </CardFooter>
  </Card>
);
};

```

Du côté de l'API, je vais détailler le service *PostService.cs*. D'abord je vais chercher le bon nombre (pagination) de posts dans la base via le repository. Ensuite avec un switch dans la méthode *GetPostOrderValue* j'applique le filtre, puis le sens demandé (ascendant ou descendant) gardant ainsi la logique métier dans le service.

```

public class PostService(IPostRepository postRepository, IMapper mapper) : IPostService
{
    private readonly IPostRepository _postRepository = postRepository;
    private readonly IMapper _mapper = mapper;

    4 references | ● 2/2 passing
    public async Task<ApiResponse<IEnumerable<PostSummaryDto>>> GetPostsSummaryByFilterAsync(PostFilterDto filter)
    {

        IEnumerable<Post>? targetPosts = await _postRepository.GetPersonPostsBySkipAndTakeAsync(filter.PersonId, filter.Skip, filter.NbToGet);

        if (targetPosts == null || !targetPosts.Any())
        {
            string errorMessage = filter.Skip == 0 ? ErrorMessages.noPostsFound : ErrorMessages.noMorePostsFound;
            return ApiResponse<IEnumerable<PostSummaryDto>>.Failure(errorMessage);
        }

        Func<Post, object> orderBy = GetPostOrderValue(filter.OrderBy);

        IEnumerable<Post> orderedPosts = filter.IsAsc ? targetPosts.OrderBy(orderBy) : targetPosts.OrderByDescending(orderBy);

        IEnumerable<PostSummaryDto> postSummaries = orderedPosts.Select(post => new PostSummaryDto
        {
            Title = post.Title,
            Date = post.Timestamp.ToString("dd-MM-yyyy"),
            Content = post.TextContent ?? ""
        }).ToList();

        return ApiResponse<IEnumerable<PostSummaryDto>>.Success(postSummaries);
    }

    /// <summary>
    /// Returns a lambda expression to sort posts based on the specified sorting criterion.
    /// </summary>
    1 reference
    public Func<Post, object> GetPostOrderValue(PostOrderBy orderBy)
    {
        return orderBy switch
        {
            PostOrderBy.Date => post => post.Timestamp,
            PostOrderBy.Likes => post => post.LikeCount,
            PostOrderBy.Comments => post => post.Comments.Count,
            PostOrderBy.Tag => post => post.Tag,
            => post => post.Timestamp
        };
    }
}

```

### Composant *ProfileCard* :

Ce composant gère l'affichage des informations de l'utilisatrice et leurs modifications.

Quand l'utilisatrice sauvegarde ses nouvelles informations on met à jour les informations.

```
const ProfileCard = ({ personDetails, isMyProfile }: Props) => {
  const [isEditing, setIsEditing] = useState(false);
  const [isFileImported, setIsFileImported] = useState(false);
  const { updatePersonDetails, error } = usePersonProfileStore();

  const form = useForm<PersonDetails>({
    resolver: zodResolver(profileSchema),
    defaultValues: personDetails,
  });

  useEffect(() => {
    form.reset(personDetails);
  }, [personDetails, form.reset]);

  const handleSubmit = async (data: PersonDetails) => {
    if (!isFileImported) {
      data.image = personDetails.image;
    }
    const completePersonDetails = { ...data, id: personDetails.id };
    await updatePersonDetails(completePersonDetails);
    setIsFileImported(false);
    setIsEditing(false);
  };

  const handleAvatarChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    if (e.target.files && e.target.files[0]) {
      const file = e.target.files[0];
      const result = profileSchema.shape.image.safeParse(file);
      if (!result.success) {
        form.setError('image', {
          message: result.error.issues[0].message,
        });
        return;
      }
      const reader = new FileReader();
      reader.readAsDataURL(file);
      reader.onloadend = () => {
        form.setValue('image', reader.result as string);
        personDetails.image = reader.result as string;
        setIsFileImported(true);
      };
      form.clearErrors('image');
    }
  };

  const handleCancel = () => {
    form.reset(personDetails);
    setIsEditing(false);
  };
}
```

La fonction *handleAvatarChange* gère le changement d'image de profil utilisatrice. Lorsqu'un fichier est sélectionné, elle vérifie si le fichier est valide en utilisant les règles de validation définies dans le schéma Zod. Si le fichier est valide, il est lu en tant que base64 via un FileReader et la valeur de l'image est mise à jour dans le formulaire. Si le fichier est invalide, un message d'erreur est affiché dans le formulaire.

```

return (
  <Card
    className="min-w-[17rem] md:max-w-[25rem] bg-card-lilac rounded-large"
    role="form"
  >
  {error && <ErrorMessage message={error} />}
  <Form {...form}>
    <form onSubmit={form.handleSubmit(handleSubmit)}>
      <CardHeader className="flex items-center">
        <div className="relative">
          {isEditing && form.control._defaultValues.image && (
            <DeleteButton onClick={handleDeleteImage} />
          )}
          <Avatar className="h-[11rem]">
            <AvatarImage src={personDetails.image} alt={t('ai-avatar')} />
            <AvatarFallback>{personDetails.username}</AvatarFallback>
          </Avatar>
          {isEditing && (
            <GenericField ...>
              <GenericField>
            </GenericField>
          )}
        </div>
        {form.formState.errors.image && isEditing && (
          <FormMessage>{form.formState.errors.image.message}</FormMessage>
        )}
      </CardHeader>
      <CardContent className="mx-2 space-y-8 font-medium">
        <div className="space-y-2">
          <CardTitle>
            {isEditing ? (
              <GenericField
                name="username" ...
                control={form.control}
              />
            ) : (
              personDetails.username
            )}
          </CardTitle>
          <CardDescription>
            {isEditing ? (
              <GenericField
                name="pronouns" ...
                control={form.control}
              />
            ) : (
              personDetails.pronouns
            )}
          </CardDescription>
        </div>
        <p className="text-m leading-5 whitespace-pre-wrap text-secondary">
          {isEditing ? (
            <GenericField
              name="bio" ...
              control={form.control}
            />
          ) : (
            personDetails.bio
          )}
        </p>
        <CardFooter className="flex pb-0 justify-center space-x-4">
          {isEditing && (
            <button
              type="submit" ...
              aria-label={t("buttonAction.validate")}
            />
            <button
              type="button" ...
              onClick={handleCancel}
            />
          )}
          {isEditing & isMyProfile && (
            <button
              type="button" ...
              onClick={() => setIsEditing(true)}
            />
          )}
        </CardFooter>
      </CardContent>
    </form>
  </Card>
);

```

J'ai fait des champs de formulaire un composant à part pour faire en sorte qu'il soit réutilisable.

J'ai aussi séparé le schéma de validation du composant pour plus de visibilité.

Je vais maintenant parler de la route API permettant à l'utilisateur de modifier son profil.

```

export const profileSchema = z.object({
  username: z
    .string({ required_error: i18n.t('formValidation.required') })
    .min(3, i18n.t('formValidation.usernameMinLength'))
    .max(20, i18n.t('formValidation.usernameMaxLength')),
  pronouns: z
    .string()
    .max(14, i18n.t('formValidation.pronounsToLong'))
    .optional(),
  bio: z.string().max(80, i18n.t('formValidation.bioToLong')).optional(),
  image: z
    .union([
      z
        .instanceof(File)
        .refine(file => file && ACCEPTED_IMAGE_MIME_TYPES.includes(file.type), {
          message: i18n.t('formValidation.invalidImageType'),
        })
        .refine(file => file && file.size <= MAX_FILE_SIZE, {
          message: i18n.t('formValidation.imageTooLarge'),
        }),
      z.string().optional().nullable(),
    ])
    .optional()
    .nullable(),
});

```

Tout d'abord le Controller, il appelle le service *PersonService* et sa méthode *UpdatePersonWithAvatar* en utilisant son interface.

```
[Route("api/[controller]")]
[ApiController]
0 references
public class PeopleController(IGenericService<PersonDto, Person> service, IPersonService personService)
    : GenericController<PersonDto, Person>(service)
{
    private readonly IPersonService _personService = personService;

    [HttpGet("GetPerson")]
    0 references
    public async Task<ApiResponse<PersonDto>> GetByIdWithAvatar([FromQuery] int personId)
    {
        ApiResponse<PersonDto> response = await _personService.GetByIdWithAvatar(personId);
        return response;
    }

    [HttpPost("UpdatePerson")]
    0 references
    public async Task<ApiResponse<PersonDto>> UpdatePersonWithAvatar([FromBody] PersonDto newPersonDto)
    {
        ApiResponse<PersonDto> response = await _personService.UpdatePersonWithAvatar(newPersonDto);
        return response;
    }
}
```

```
public interface IPersonService
{
    4 references | 2/2 passing
    Task<ApiResponse<PersonDto>> GetByIdWithAvatar(int personId);
    2 references
    Task<ApiResponse<PersonDto>> UpdatePersonWithAvatar(PersonDto newPersonDto);
}
```

La méthode *UpdatePersonWithAvatar* met à jour les informations d'une personne, y compris son avatar.

Elle commence par récupérer la personne avec son avatar. Si la personne n'existe pas, elle retourne une erreur. Si un avatar existe déjà, il est supprimé avant de sauvegarder le nouvel avatar si fourni.

Les informations du DTO sont ensuite mappés sur la personne, qui est mise à jour dans la base. La méthode retourne enfin le DTO mis à jour en cas de succès.

```

3 references
public class PersonService(IPersonRepository personRepository, IImageService imageService, IMapper mapper) : IPersonService
{
    private readonly IPersonRepository _personRepository = personRepository;
    private readonly IImageService _imageService = imageService;
    private readonly IMapper _mapper = mapper;

    4 references | 2/2 passing
    public async Task<ApiResponse<PersonDto>> GetByIdWithAvatar(int personId)
    {
        Person? person = await _personRepository.GetByIdWithAvatar(personId);
        if (person == null)
            return ApiResponse<PersonDto>.Failure(ErrorMessages.userNotFound);

        PersonDto personDto = _mapper.Map<PersonDto>(person);

        return ApiResponse<PersonDto>.Success(personDto);
    }

    2 references
    public async Task<ApiResponse<PersonDto>> UpdatePersonWithAvatar(PersonDto newPersonDto)
    {
        var person = await _personRepository.GetByIdWithAvatar(newPersonDto.Id);

        if (person == null)
            return ApiResponse<PersonDto>.Failure(ErrorMessages.userNotFound);

        if (person.ImageId.HasValue)
        {
            await _imageService.DeleteImageAsync(person.ImageId.Value);
            if (!string.IsNullOrEmpty(newPersonDto.Image))
                person.ImageId = (await _imageService.SaveImageAsync(newPersonDto.Image)).Id;
            else
                person.ImageId = null;
        }
        else
        {
            if (!string.IsNullOrEmpty(newPersonDto.Image))
                person.ImageId = (await _imageService.SaveImageAsync(newPersonDto.Image)).Id;
        }

        _mapper.Map(newPersonDto, person);

        await _personRepository.UpdatePerson(person);
        var updatedPersonDto = _mapper.Map<PersonDto>(person);

        return ApiResponse<PersonDto>.Success(updatedPersonDto);
    }
}

```

La méthode `SaveImageAsync` de `ImageService` utilisée dans le service que je viens de présenter enregistre une image dans la base de données. Elle commence par décoder l'image base64 en un tableau de bytes. Ensuite, elle crée une nouvelle instance de `Image` et `ImageInfo` qui contient les informations de l'image (type de données, taille et horodatage). Enfin, elle ajoute cette image à la base de données via le `IImageRepository` et retourne l'image enregistrée.

```
3 references
public class ImageService(IImageRepository imageRepository) : IImageService
{
    private readonly IImageRepository _imageRepository = imageRepository;

    4 references | 1/1 passing
    public async Task<Image> SaveImageAsync(string base64Image)
    {
        // Decode base64 image and save it to the database
        var imageBytes = Convert.FromBase64String(base64Image.Split(',')?[1]);
        var imageWithInfo = new Image
        {
            Content = imageBytes,
            Info = new ImageInfo
            {
                DataType = "image/png",
                Size = imageBytes.Length,
                Timestamp = DateTime.UtcNow
            }
        };
        await _imageRepository.AddAsync(imageWithInfo);
        return imageWithInfo;
    }
    3 references | 1/1 passing
    public async Task DeleteImageAsync(int imageId)
    {
        var image = await _imageRepository.GetByIdAsync(imageId);
        if (image != null)
        {
            await _imageRepository.DeleteAsync(image);
        }
    }
}
```

```

1 reference
public class PersonRepository(DerbeezContext context) : IPersonRepository
{
    private readonly DerbeezContext _context = context;

    5 references | 2/2 passing
    public async Task<Person?> GetByIdWithAvatar(int id)
    {
        return await _context.People
            .Include(p => p.Image)
            .FirstOrDefaultAsync(p => p.Id == id);
    }

    2 references
    public async Task UpdatePerson(Person person)
    {
        _context.Entry(person).State = EntityState.Modified;
        await _context.SaveChangesAsync();
    }
}

```

La méthode *UpdatePerson* met à jour les informations d'une personne dans la base de données. Elle marque l'entité *Person* comme modifiée dans le contexte de la base de données et enregistre les changements en appelant *SaveChangesAsync*.

## 6.3. Tests

Je vais maintenant vous présenter deux des tests que j'ai créés pour cette nouvelle page profile.

Côté front-end :

```

7 jest.mock('../services/backendServices/personBackendService');
8
9 describe('usePersonProfileStore', () => {
10     it('should fetch person details successfully', async () => {
11         const personDetails: PersonDetails = {
12             id: 1,
13             username: 'johndoe',
14             bio: 'bio',
15             pronouns: 'he/him',
16             image: 'image_url',
17         };
18         (getPerson as jest.Mock).mockResolvedValue({ isSuccess: true, data: personDetails });
19
20         const { result } = renderHook(() => usePersonProfileStore());
21
22         act(() => {
23             result.current.fetchPersonDetails(1);
24         });
25
26         await waitFor() => {
27             expect(result.current.personDetails).toEqual(personDetails);
28         });
29     });
30
31     it('should handle fetch person details failure', async () => {
32         (getPerson as jest.Mock).mockResolvedValue({ isSuccess: false, errorMessage: 'Error fetching details' });
33
34         const { result } = renderHook(() => usePersonProfileStore());
35
36         act(() => {
37             result.current.fetchPersonDetails(1);
38         });
39
40         await waitFor() => {
41             expect(result.current.error).toBe('Error fetching details');
42         });
43     });
44
45     it('should update person details successfully', async () => {
46         const updatedDetails: PersonDetails = {
47             id: 1,
48             username: 'janedoe',
49             bio: 'updated bio',
50             pronouns: 'she/her',
51             image: 'updated_image_url',
52         };
53         (updatePersonDetails as jest.Mock).mockResolvedValue({ isSuccess: true, data: updatedDetails });
54
55         const { result } = renderHook(() => usePersonProfileStore());
56
57         act(() => {
58             result.current.updatePersonDetails(updatedDetails);
59         });
60
61         await waitFor() => {
62             expect(result.current.personDetails).toEqual(updatedDetails);
63         });
64     });
65
66     it('should handle update person details failure', async () => {
67         (updatePersonDetails as jest.Mock).mockResolvedValue({ isSuccess: false, errorMessage: 'Error updating details' });
68
69         const { result } = renderHook(() => usePersonProfileStore());
70
71         act(() => {
72             result.current.updatePersonDetails({
73                 id: 1,
74                 username: 'janedoe',
75                 bio: 'updated bio',
76                 pronouns: 'she/her',
77                 image: 'updated_image_url',
78             } as PersonDetails);
79         });
80
81         await waitFor() => {
82             expect(result.current.error).toBe('Error updating details');
83         });
84     });
85 });

```

Ces tests couvrent les scénarios de

```

PASS  src/test/Auth/ProfileStore.test.tsx
usePersonProfileStore
  ✓ should fetch person details successfully (89 ms)
  ✓ should handle fetch person details failure (80 ms)
  ✓ should update person details successfully (81 ms)
  ✓ should handle update person details failure (76 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:  0 total
Time:        5.553 s

```

succès et d'échec pour la récupération et la mise à jour des détails de personne en utilisant le store *usePersonProfileStore* :

- On récupère bien les données de person avec le test *Fetch person details successfully*.
- On gère de manière correcte les échec au moment de récupérer les détails de la personne avec *Handle fetch person details failure*.
- On vérifie que les données sont bien actualisées quand la personne modifie son profil avec *Update person details successfully*.
- On gère correctement les erreurs au moment de la mise a jour d'un profil avec *Handle update person details failure*.

## Côté back-end :

[TestFixture]			
0 references			
public class PostServiceTests			
{			
private Mock<IPostRepository> _postRepositoryMock;	✓ DeleteImageAsync_ShouldDeleteImage_WhenImageExists	131 ms	
private Mock<IMapper> _mapperMock;	✓ GetByIdWithAvatar_ShouldReturnFailureResponse_WhenPersonsNotFound	23 ms	
private PostService _postService;	✓ GetByIdWithAvatar_ShouldReturnSuccessResponse_WhenPersonsFound	5 ms	
[Setup]	✓ GetPostsByFilterAsync_ShouldReturnFailureResponse_WhenNoPostsAreFound	16 ms	
0 references	✓ GetPostsSummaryByFilterAsync_ShouldReturnSuccessResponse_WhenPostsA...	20 ms	
public void SetUp()	✓ SaveImageAsync_ShouldSaveImageAndReturnImage	28 ms	
{			
_postRepositoryMock = new Mock<IPostRepository>();			
_mapperMock = new Mock<IMapper>();			
_postService = new PostService(_postRepositoryMock.Object, _mapperMock.Object);			
}			
[Test]			
● 0 references			
public async Task GetPostsSummaryByFilterAsync_ShouldReturnSuccessResponse_WhenPostsAreFound()			
{			
// Arrange			
var filter = new PostFilterDto { PersonId = 1, Skip = 1, NbToGet = 1 };			
var posts = new List<Post>();			
{			
new() {			
Title = "Test Title",			
Creator = new Person			
{			
Username = "test"			
},			
Timestamp = DateTime.UtcNow,			
TextContent = "Test Content",			
};			
};			
var postSummaries = new List<PostSummaryDto>();			
{			
new() {			
Title = "Test Title",			
Date = posts[0].Timestamp.ToString("dd-MM-yyyy"),			
Content = "Test Content"			
};			
};			
_postRepositoryMock.Setup(repo => repo.GetPersonPostsBySkipAndTakeAsync(filter.PersonId, filter.Skip, filter.NbToGet))			
.ReturnsAsync(posts);			
// Act			
var result = await _postService.GetPostsSummaryByFilterAsync(filter);			
// Assert			
Assert.Multiple(() =>			
{			
Assert.That(result.IsSuccess, Is.True);			
Assert.That(result.Data, Is.Not.Null);			
Assert.That(result.Data.Count(), Is.EqualTo(postSummaries.Count));			
for (int i = 0; i < result.Data.Count(); i++)			
{			
Assert.That(result.Data.ElementAt(i).Title, Is.EqualTo(postSummaries[i].Title));			
Assert.That(result.Data.ElementAt(i).Date, Is.EqualTo(postSummaries[i].Date));			
Assert.That(result.Data.ElementAt(i).Content, Is.EqualTo(postSummaries[i].Content));			
};			
};			
}			
}			
[Test]			
● 0 references			
public async Task GetPostsByFilterAsync_ShouldReturnFailureResponse_WhenNoPostsAreFound()			
{			
// Arrange			
var filter = new PostFilterDto { PersonId = 1 };			
_postRepositoryMock.Setup(repo => repo.GetPersonPostsBySkipAndTakeAsync(filter.PersonId, filter.Skip, filter.NbToGet))			
.ReturnsAsync([]);			
// Act			
var result = await _postService.GetPostsSummaryByFilterAsync(filter);			
// Assert			
Assert.Multiple(() =>			
{			
// Assert			
Assert.That(result.IsSuccess, Is.False);			
Assert.That(result.ErrorMessage, Is.EqualTo(ErrorMessages.noPostsFound));			
};			
}			

Ces tests couvrent les scénarios de succès et d'échec pour la récupération et la mise à jour des détails de personne en utilisant le store *PostService*:

- Dans le premier test on vérifie la bonne marche du service dans le cas où des posts sont présents grâce à des mocks.
- Dans le deuxième test, on vérifie le bon fonctionnement si il n'y a pas de posts, on vérifie qu'on reçoit bien le bon message d'erreur.

## 6.4.Résultats

- Si c'est mon profil sans modification en cours :

Derbeez

Histoire Entraînement Communauté Actualités



**Klervy**  
she/her  
Trop contente d'être là !!  
👉 ❤️ 👉

**Éditer**

**Posts**

Post Title	Date	Description
First Roller Derby Skates	26-07-2024	Just got my first pair of roller derby skates! Excited to start practicing.
Learning Roller Derby Rules	26-07-2024	Learning the rules of roller derby. It is more complex than I thought!
Joined a Roller Derby Team	26-07-2024	Joined a local roller derby team for beginners. Can not wait for our first practice!
Roller Derby Tips for Beginners	26-07-2024	Found some great tips for beginners. Practicing my balance and falls.
First Roller Derby Practice	26-07-2024	First practice session was intense but fun! Lots to learn.
Roller Derby Protective Gear	26-07-2024	Invested in some protective gear. Safety first!

Précédent **Suivant**

Ascendant

Filtrer

Trier par

- Date
- Likes
- Commentaires
- Tag

- Si c'est mon profil et que je suis en train de le modifier :

Derbeez

Histoire Entraînement Communauté Actualités



Pseudo  
**Klervy**

Pronoms  
she/her

Biographie  
Trop contente d'être là !!  
👉 ❤️ 👉

**Valider** **Annuler**

**Posts**

Post Title	Date	Description
First Roller Derby Skates	26-07-2024	Just got my first pair of roller derby skates! Excited to start practicing.
Learning Roller Derby Rules	26-07-2024	Learning the rules of roller derby. It is more complex than I thought!
Joined a Roller Derby Team	26-07-2024	Joined a local roller derby team for beginners. Can not wait for our first practice!
Roller Derby Tips for Beginners	26-07-2024	Found some great tips for beginners. Practicing my balance and falls.
First Roller Derby Practice	26-07-2024	First practice session was intense but fun! Lots to learn.
Roller Derby Protective Gear	26-07-2024	Invested in some protective gear. Safety first!

Précédent **Suivant**

Ascendant

Filtrer

- Version tablette et smartphone :

The image shows two side-by-side screenshots of the Derbeez mobile application. Both screens feature a purple header with the 'Derbeez' logo and a three-line menu icon. On the left (tablet view), there's a user profile section with a placeholder profile picture, 'Pseudo' (Klervy), 'Pronoms' (she/her), and a 'Biographie' box containing the text 'Trop contente d'être là !!' with a small emoji. Below this is a 'Posts' section with a green header bar containing 'Ascendant' and 'Filtrer' buttons. The posts are displayed in a grid:

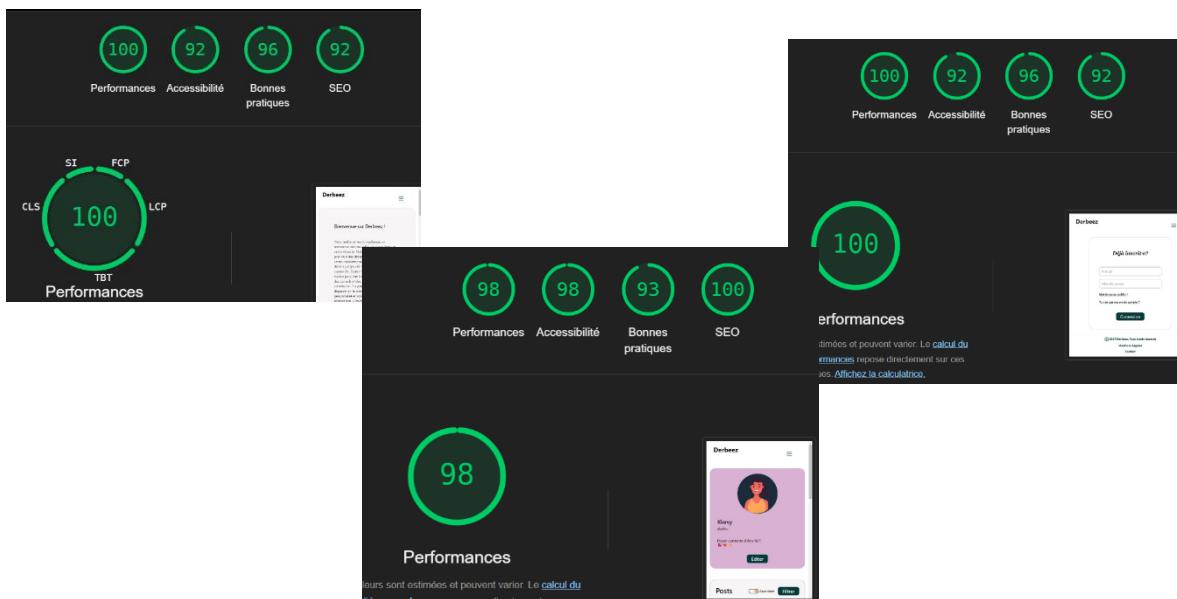
- First Roller Derby Skates** (26-07-2024): Just got my first pair of roller derby skates! Excited to start practicing.
- Learning Roller Derby Rules** (26-07-2024): Learning the rules of roller derby. It is more complex than I thought!
- Joined a Roller Derby Team** (26-07-2024): Joined a local roller derby team for beginners. Can not wait for our first practice!
- Roller Derby Tips for Beginners** (26-07-2024): Found some great tips for beginners. Practicing my balance and falls.
- First Roller Derby Practice** (26-07-2024): First practice session was intense but fun! Lots to learn.
- Roller Derby Protective Gear** (26-07-2024): Invested in some protective gear. Safety first!

At the bottom of the tablet screen are 'Précédent' and 'Suivant' buttons. On the right (smartphone view), the user profile section is identical. The 'Posts' section also has a green header bar with 'Ascendant' and 'Filtrer' buttons. The posts are displayed in a grid:

- First Roller Derby Skates** (26-07-2024): Just got my first pair of roller derby skates! Excited to start practicing.
- Learning Roller Derby Rules** (26-07-2024): Learning the rules of roller derby. It is more complex than I thought!
- Joined a Roller Derby Team** (26-07-2024): Joined a local roller derby team for beginners. Can not wait for our first practice!
- Roller Derby Tips for Beginners** (26-07-2024): Found some great tips for beginners. Practicing my balance and falls.
- First Roller Derby Practice** (26-07-2024): First practice session was intense but fun! Lots to learn.
- Roller Derby Protective Gear** (26-07-2024): Invested in some protective gear. Safety first!

At the bottom of the smartphone screen are 'Précédent' and 'Suivant' buttons.

- Résultats de l'audit avec LightHouse :



## 7. Veille technologique

Pour rester à jour dans un domaine en constante évolution, je mets en place une veille technologique, en utilisant plusieurs plateformes et outils pour m'informer sur les dernières innovations et bonnes pratiques.

**Reddit** est une source précieuse pour ma veille technologique, notamment grâce à des subreddits dédiés aux technologies que j'utilise quotidiennement.

Je suis abonné à de nombreux subreddits comme ceux liés directement à ma stack technique (*r/Angular, r/PostgreSQL, r/dotnet, r/ASPNET*), d'autres plus axés sur la sécurité, l'architecture et le devops (*r/sysadmin, r/netsec, r/devops, r/softwarearchitecture*).

Ces communautés sont très actives et offrent un flux constant d'articles, de discussions, de questions/réponses et de partages d'expériences pratiques qui m'aident à rester informée des dernières mises à jour, des astuces et des problèmes courants rencontrés par d'autres développeurs.

**Medium** est une autre source de veille. Les articles sur Medium sont souvent détaillés et offrent des perspectives approfondies sur l'architecture, les bonnes pratiques de codage, les tutoriels ainsi que des études de cas.

**C#Corner** qui comme son nom l'indique offre des articles sur C# et comme son nom l'indique moins parle aussi des technologies gravitant autour de C#, comme les frameworks .NET ainsi qu'Angular car il est très souvent choisi pour le front des projets de web ASP.NET.

En combinant ces différentes sources et en restant curieuse et engagée, je m'assure de rester à jour dans ma veille et de pouvoir apporter des solutions innovantes et efficaces à mes projets professionnels.

## 8.Remerciements & conclusion

### Remerciements

Je veux remercier mon école ainsi que les encadrants pour avoir crus en nous, même quand on n'arrivait pas à comprendre ce qu'était un paramètre au bout de 20 minutes d'explications en début de formation.

Je tiens à remercier mon entreprise, Arpège et mes collègues de travail, pour la confiance et la patience dont ils ont fait preuve tout au long de mon apprentissage.

J'aimerais aussi remercier mon entourage, particulièrement mon conjoint qui ma soutenue sans relâche lors des longs week-end de travail pour ce projet.

### Conclusion

Le projet DERBEEZ a été une expérience d'apprentissage riche et gratifiante qui m'a permis d'explorer des domaines que je n'avais pas encore abordés au cours de mon alternance. En effet, travaillant dans une entreprise avec des produits déjà bien établis, certaines parties de la conception d'application comme l'architecture ne sont pas au centre de mon travail de développeuse de tous les jours.

De plus, j'ai redécouvert React, une technologie que j'ai moins l'occasion d'utiliser au quotidien, car chez Arpège j'utilise uniquement Angular pour le front-end. Ce projet m'a permis de comparer et de contraster les deux frameworks.

Les discussions avec mes collègues de travail ont également été une grande source d'enrichissement. J'ai eu des débats passionnants sur divers sujets, confirmant que dans le monde du développement, il n'y a pas souvent de consensus clair.

Cette diversité d'opinions et d'approches est ce que j'aime le plus dans ce domaine, elle pousse constamment à l'innovation et à l'apprentissage, je suis sûre de ne jamais m'ennuyer dans ce métier.

En conclusion, cette année a confirmé mon choix de réorientation vers le développement. Je suis plus convaincue que jamais que c'est le bon chemin pour moi et je suis impatiente de continuer à explorer et à grandir dans ce domaine en constante évolution.