# Projet Salle de Concert

GARDE Dorian - M2

GOSALBES Cloé - M2

ORTEGA Guillaume - M2

STANKO Jérémy - M2

# SOMMAIRE

# I. PRÉSENTATION DE L'ÉQUIPE

## FORCES ET FAIBLESSES DE CHAQUE MEMBRE

| | FORCES | FAIBLESSES |
|---|---|---|
| **Dorian** | **GIT/PHP** | **Mal organisé (va push une modif en oubliant certaines étapes que les autres devront faire pour que ça fonctionne, et fait perdre 10min par ci par la)** |
| **Jérémy** | **Connaissance Git** | **Tortue** |
| **Guillaume** | **Efficace lors des sprints** | **A besoin d'un sprint pour avancer** |
| **Cloé** | | **Peu habituée à Git – Lenteur dans la prise en main – Difficulté de conceptualisation** |

## RÔLES ET TACHES POUR CHAQUE MEMBRE

Afin de réaliser ce projet, nous avons attribué à chaque membre, un rôle en fonction de ses compétences. Nous nous sommes repartis de la manière suivante :

- Dorian : Développeur Back-End
- Cloé : Développeur Front-End
- Guillaume : Développeur Front-End
- Jérémy : Dévéloppeur Front-End

Notre choix de diviser l'équipe en une majorité de développeur front-end s'explique essentiellement par le fait que nous jugions la charge de travail beaucoup plus conséquentes en Front qu'en Back, notamment parce qu'il nous était nécessaire de nous former à la technologie React. En plus de nos rôles de développeurs, nous nous sommes aussi illustrer grâce à nos compétences supplémentaires tel que :

- Jérémy et Dorian, dans la gestion du projet au niveau Git, afin d'appliquer la répartition des tâches, et éviter les pertes de temps à cause de conflits, ou de perte de code.
- Cloé, en tant que SCRUM Master, qui a permis une répartition rapide et efficace des tâches à réaliser

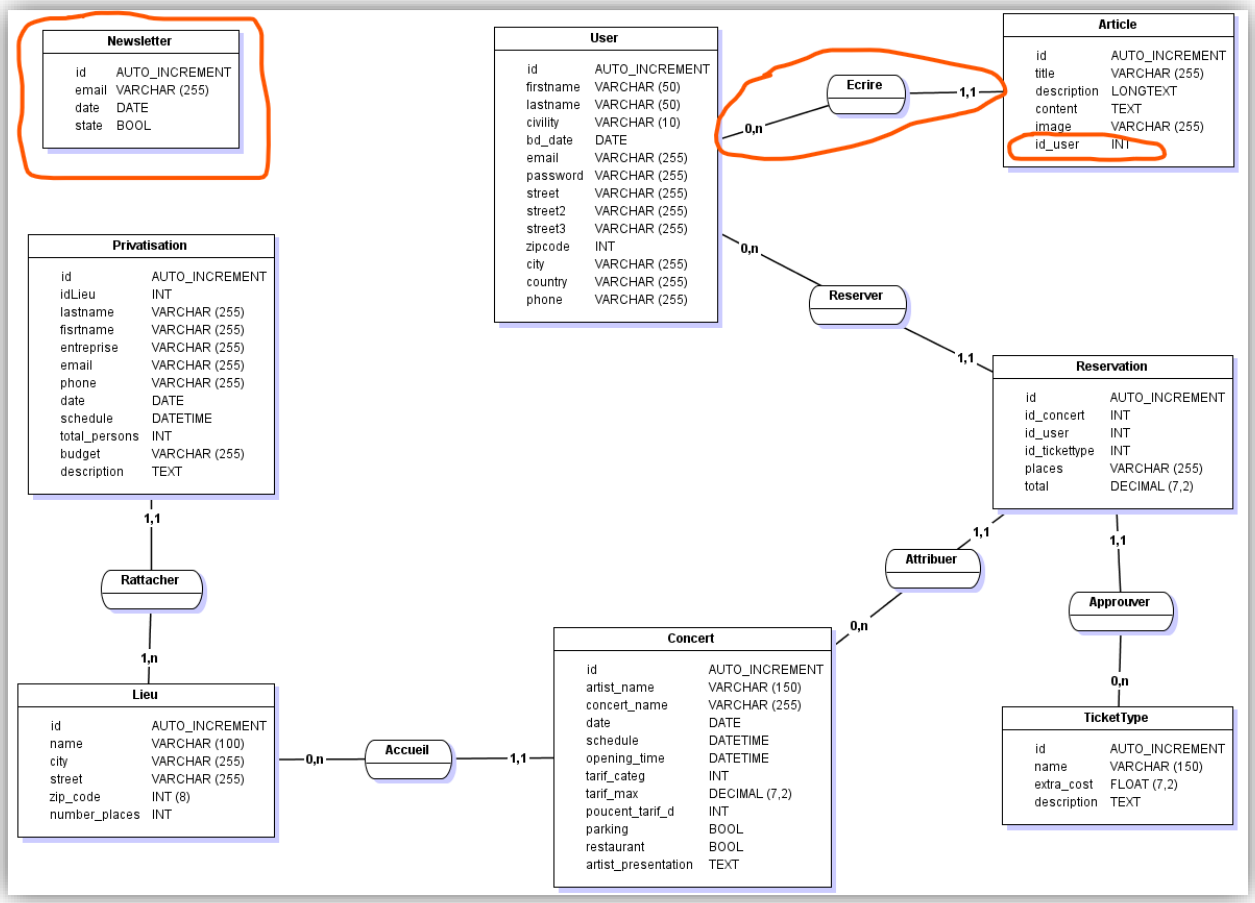
| PAGE | TÂCHES | RÉALISÉ PAR |
|---|---|---|
| Toutes | Navbar & Gestion des routes | Guillaume |
| Toutes | Barre de reacherche | Guillaume |
| Accueil | Footer | Cloé |
| Accueil | Contenu de la page d'accueil | Jérémy |
| Programmation | Contenu de la page de programmation | Jérémy |
| Concert | Contenu de la page présentation d'un concert | Jérémy |
| Compte administrateur | Onglets concert – Ajouter un concert - pagination | Guillaume |
| Réservation - Étape 1 | Plan de salle – Choix du type de ticket – Contenu de la page | Cloé |
| | | |

**Backend**

| ROUTE | TÂCHES | RÉALISÉ PAR |
|---|---|---|
| Concert | Entité<br>Data fixtures | Dorian |
| User | Entité<br>Data fixtures | Dorian |
| Lieu | Entité<br>Data fixtures | Dorian |
| TicketType | Entité<br>Data fixtures | Dorian |
| Privatisation | Entité<br>Data fixtures | Dorian |
| Article | Entité et Data fixtures récupérées de votre initialisation de projet pour ne pas perdre de temps. Puis ajout de la colonnes images dans un second temps | Dorian - Olivier Poussel (bienvenue dans le groupe) |
| /api/login_check | Installation du bundle LexikJWTAuthenticationBundle plus paramétrage de celui-ci. Création des clés publiques et privées. | Dorian |
| /api /concerts/checkdispo/{idconcert} | Création du ConcertController appelant une méthode dans ConcertRepository, méthode appelant la base de données pour connaitre toutes les places déjà réservé pour un concert donné | Dorian |

## II. LES CHOIX TECHNIQUES, ARCHITECTURAUX ET GRAPHIQUES

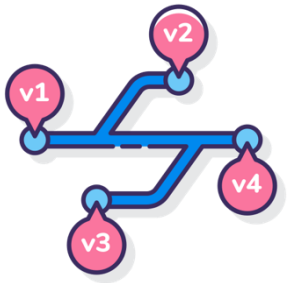
### MODÈLE DE LA BASE DE DONNÉES




Légende : --------- (Trait Orange) => Ce qui aurait bon de faire mais qui n'a pas été fait sur le projet

Avec l'arrivé du MVP en milieu de projet, associé au manque de temps, il est possible que des points soit manquant car non analysé. Les efforts ont ici été concentré sur la maquette MVP.

## TECHNIQUES / ARCHITECTURAUX




Pour notre Git, nous avons une construction ressemblant à :

- Master
  - Develop
    - Backend
    - Cloe
    - Guillaume

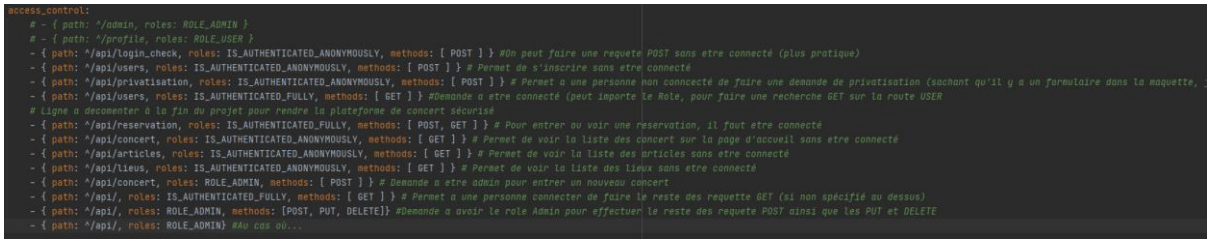
- Jeremy

Le but ici est de découper le développement entre chaque personne. Au début du projet nous n'avions pas la même architecture de branche. Au départ, chaque fonctionnalité devait avoir sa branche. Par exemple sur la page d'accueil, une fois que le développeur termine une partie, comme le slider, il va « commiter » son travail sur une sous branche feature/accueil-slider. Une fois la branche validée, alors elle est merge sur develop, puis supprimée.

C'était le système qui a été réfléchi au départ, pour avoir de la lisibilité et ne pas mélanger les branches. Au final, avec le temps qui commençait à nous manquer nous avons dû abandonner ce système pour passer sur quelque chose de plus simple et plus efficace pour gagner du temps. Chaque développeur possède sa propre branche : il pousse les modifications dessus avant de s'assurer d'être à jour sur develop, et merge sa branche sans la supprimer sur develop, et ainsi de suite. L'avantage de cette méthode est la rapidité pour développer sans se soucier des branche etc, mais cela facilite aussi les conflits entre nous, si nous ne vérifions pas si nous sommes à jour. Pour la branche master, elle existe seulement pour les livraisons, dont la livraison finale. Quand tout est vérifié et validé sur develop, alors on merge develop sur master.

**Sécurité des routes**

Afin de sécuriser notre API nous avons décidé de sécuriser nos routes grâce à un outil proposé par Symfony qui s'appelle Access_control. Pour ce faire, à la manière d'un pare-feu, nous avons ajouté une règle bloquant tout, puis au fur et à mesure des besoins, nous avons rajouter des lignes au-dessus permettant un accès moins strict.



```
access_control:
    # - { path: ^/admin, roles: ROLE_ADMIN }
    # - { path: ^/profile, roles: ROLE_USER }
    - { path: ^/api/login_check, roles: IS_AUTHENTICATED_ANONYMOUSLY, methods: [ POST ] } #On peut faire une requete POST sans etre connecté (plus pratique)
    - { path: ^/api/users, roles: IS_AUTHENTICATED_ANONYMOUSLY, methods: [ POST ] } # Permet de s'inscrire sans etre connecté
    - { path: ^/api/privatisation, roles: IS_AUTHENTICATED_ANONYMOUSLY, methods: [ POST ] } # Permet a une personne non conncecté de faire une demande de privatisation (sachant qu'il y a un formulaire dans la maquette,
    - { path: ^/api/users, roles: IS_AUTHENTICATED_FULLY, methods: [ GET ] } #Demande a etre connecté (peut importe le Role, pour faire une recherche GET sur la route USER
    # Ligne a decomenter à la fin du projet pour rendre la plateforme de concert sécurisé
    - { path: ^/api/reservation, roles: IS_AUTHENTICATED_FULLY, methods: [ POST, GET ] } # Pour entrer ou voir une reservation, il faut etre connecté
    - { path: ^/api/concert, roles: IS_AUTHENTICATED_ANONYMOUSLY, methods: [ GET ] } # Permet de voir la liste des concert sur la page d'accueil sans etre connecté
    - { path: ^/api/articles, roles: IS_AUTHENTICATED_ANONYMOUSLY, methods: [ GET ] } # Permet de voir la liste des articles sans etre connecté
    - { path: ^/api/lieus, roles: IS_AUTHENTICATED_ANONYMOUSLY, methods: [ GET ] } # Permet de voir la liste des lieux sans etre connecté
    - { path: ^/api/concert, roles: ROLE_ADMIN, methods: [ POST ] } # Demande a etre admin pour entrer un nouveau concert
    - { path: ^/api/, roles: IS_AUTHENTICATED_FULLY, methods: [ GET ] } # Permet a une personne connecter de faire le reste des requette GET (si non spécifié au dessus)
    - { path: ^/api/, roles: ROLE_ADMIN, methods: [POST, PUT, DELETE]} #Demande a avoir le role Admin pour effectuer le reste des requete POST ainsi que les PUT et DELETE
    - { path: ^/api/, roles: ROLE_ADMIN} #Au cas où...
```

Par exemple ici, la première ligne permet de faire une requête GET vers "/api/login_check" qui permet de se connecter (règle primordiale pour obtenir un token et effectuer d'autres requêtes qui demanderont à l'utilisateur d'être connecté), grâce au rôles "IS_AUTHENTICATED_ANONYMOUSLY" qui autorise une personne non connectée.

Quelque ligne après, on peut voir apparaitre la notion de "IS_AUTHENTICATED_FULLY" qui demande à être connecté, peu importe le rôle.

Encore plus loin, nous voyons le rôle "ROLE_ADMIN" qui lui demande un token de personne connectée ayant le rôle ADMIN pour faire la requête.

Les token sont générés par le bundle *LexikJWTAuthenticationBundle* qui s'occupe de créer et gérer la route de connexion. En lui envoyant un Json composé d'un email (élément identifiant l'utilisateur, à choisir dans le security.yaml) et d'un mot de passe. Il va s'occuper de trouver et vérifier que les identifiants passé soient corrects et retourne ensuite un token qui permettra à l'utilisateur de prouver qui il est (le token comprend, le rôle, l'email, le nom et le prénom de l'utilisateur connecté).

PS : Ces lignes ne sont pour le moment pas activées (commentées) car le front ne prend à ce jour pas encore compte du token, la partie authentification étant toutes fraiche et non fonctionnel.

### GRAPHIQUES / VISUELS

Concernant la partie graphique et visuelle de notre application, nous avons fait le choix d'utiliser la libraire *React-bootstrap*. Cela nous a permis de gagner énormément de temps, notamment sur la partie design de notre interface, puisque la librairie met à notre disposition des éléments déjà stylisés grâce à l'intégration de Bootstrap dans le projet, en plus de nous offrir les avantages du responsive design.

## III. RÉALISATION DE LA MAQUETTE

### ON A FAIT

| | |
|---|---|
| **UNIQUEMENT LE MVP** | **X** |
| **MVP + AUTRES PAGES** | |

Nous n'avons réalisé seulement le MVP à cause d'un lancement du projet assez long du a des problèmes technique. Nous avons par la suite orienté le développement vers une méthode la plus efficace possible afin de réaliser le plus de features possible dans un temps serré.

### IDENTIFIANTS D'UN UTILISATEUR ET D'UN ADMINISTRATEUR

Non fonctionnel sur le front

| RÔLE | IDENTIFIANT | MOT DE PASSE |
|---|---|---|
| ADMINISTRATEUR | admin@concert.com | test |
| UTILISATEUR | user@concert.com | test |

### DÉVELOPPEMENT DES COMPTES ADMINISTRATEUR ET UTILISATEUR

### COMPTE ADMINISTRATEUR

a. CE QUI A ÉTÉ RÉALISÉ

b. CE QUI N'A PAS ÉTÉ RÉALISÉ **ET POURQUOI**

c. LES DIFFICULTÉS RENCONTRÉES

### COMPTE UTILISATEUR

d. CE QUI A ÉTÉ RÉALISÉ

e. CE QUI N'A PAS ÉTÉ RÉALISÉ **ET POURQUOI**

f. LES DIFFICULTÉS RENCONTRÉES

# DÉVELOPPEMENT DES PAGES DU SITE

## PAGE ACCUEIL

a. LE LIEN DE LA PAGE

La page est à la racine du site internet c'est à dire « / »

b. CE QUI A ÉTÉ RÉALISÉ

La page d'accueil comporte plusieurs éléments : le header, le corps et le footer.

Le header et le footer ont été développés comme des components distincts pour pouvoir être réutilisés partout dans le site internet et pas seulement sur la page d'accueil.

La page accueil est divisé est quatre parties : le slider, les concerts à venir, les articles, et la privatisation. Un élément que l'on retrouve plusieurs fois sont les cartes. Donc, pour utiliser la force de React, j'ai découpé les cards en un component qui est réutilisé à plusieurs endroits, ainsi que dans plusieurs pages. Ce component affiche des cartes différentes en fonction du besoin. Les cartes concerts et actualités ne sont par exemple pas exactement les mêmes et il faut les différencier dans le component pour retourner le bin format de carte avec les bonnes infos.

c. CE QUI N'A PAS ÉTÉ RÉALISÉ **ET POURQUOI**

Le squelette est existant, mais le style n'est pas le même que sur la maquette. Le slider n'est pas connecté à l'api pour récupérer des informations. J'ai passé trop de temps sur la connexion à l'API et à essayer de faire fonctionner les éléments React Bootstrap, j'ai donc préféré rattraper ce temps perdu en avançant le plus possible les autres pages plutôt que passer du temps à peaufiner le design.

d. LES DIFFICULTÉS RENCONTRÉES

La difficulté majeure a été de faire communiquer le back et le front entre la page d'accueil et le component card, pour faire passer les props d'un component à l'autre (les difficultés classiques en débutant React).

## PAGE PROGRAMMATION

e. LE LIEN DE LA PAGE

/scheduled

f. CE QUI A ÉTÉ RÉALISÉ

Le début de la page seulement a été réalisé pour afficher les filtres du lieu et de la catégorie de musique.

g. CE QUI N'A PAS ÉTÉ RÉALISÉ **ET POURQUOI**

Le reste de la page. Ce sont les mêmes raisons de la page d'accueil. Beaucoup trop de temps passé pour l'API, notamment sur les requêtes multiples (plus de 3 séances) pour partir de ce que l'on avait vu en cours, mais en l'adaptant pour plusieurs requêtes.

h. LES DIFFICULTÉS RENCONTRÉES

Découverte d'axios, donc beaucoup de recherche dessus. J'étais parti au début sur une technique n'utilisant pas les nouveaux hooks de React, j'ai donc dû modifier le component, puis j'ai changé de méthode pour prendre celle de la cour qui est avec les hooks. J'ai donc essayé de mieux comprend, et j'ai au final réussi à faire fonctionner mon appel API avec les useEffect et les useState.

## PAGE CONCERT

a. LE LIEN DE LA PAGE

b. CE QUI A ÉTÉ RÉALISÉ

c. CE QUI N'A PAS ÉTÉ RÉALISÉ **ET POURQUOI**

d. LES DIFFICULTÉS RENCONTRÉES

## PAGE RÉSERVATION – ETAPE 1 – RÉSERVATION

a. LE LIEN DE LA PAGE

/reservation

b. CE QUI A ÉTÉ RÉALISÉ

Affichage du contenu, présentation du concert, le plan du concert en visuel mais pas en fonctionnel.

c. CE QUI N'A PAS ÉTÉ RÉALISÉ **ET POURQUOI**

La gestion dynamique des prix, par catégorie, ainsi que la sélection des places sur le plan. J'ai rencontré beaucoup de difficulté sur comment je pouvais mettre en place le plan de salle et j'ai demandé l'aide de mes coéquipiers concernant l'affichage qui se devait d'être cohérent pour la gestion et l'affichage du prix. De fait, l'affichage est présent mais il n'y a aucune gestion du state derrière, ni de fonction handleClick qui aurait permis de prendre en considération les places choisies par l'utilistateur.

d. LES DIFFICULTÉS RENCONTRÉES

La création du plan de salle et la gestion de ce composant a posé problème, à cause de mes difficultés à le conceptualiser et par conséquent un avancement lent, ce qui freine le développement global de la page puisque le reste des éléments présents sur la page, repose sur la bonne gestion du plan de salle.

L'étape 1 de la page de réservation étant incomplète, je n'ai pas pu poursuivre le développement des étapes suivantes, puisqu'elles étaient dépendantes de cette première étape. Ainsi, les pages :

- Page réservation – étape 2 – Panier d'achat
- Page réservation – étape 3 – Coordonnées
- Page réservation – étape 3 – Situations 1 et 2
- Page réservation – étape 4 – Paiement
- Page réservation – étape 5 - Confirmation

Les pages listées ci-dessous, n'intégrant pas le MVP, n'ont, par conséquent, pas été réalisées par notre équipe. Cette absence s'explique notamment par un retard accumulé au lancement du projet, qui ne nous permettait pas d'aller jusqu'au bout de la réalisation de la maquette.

- Page restauration – présentation
- Page restauration – réservation
- Page parking – présentation
- Page parking – réservation
- Page privatisation – présentation
- Page privatisation – réservation
- Page actualités
- Les pages infos pratiques – Comment venir ? et FAQ
- Page contact
- Les pages CGU et Mentions Légales

## IV. **CONCLUSION**

Cloé :

La réalisation de cette maquette m'a permis de découvrir la librairie React et d'en faire l'expérience à travers un projet concret, plutôt qu'à travers des tutos qui manque parfois de réalisme. La maquette s'est imposée comme un bon exercice, qui ne peut cependant pas réussir à tout le monde en vue des différentes capacités et performances des différentes équipes. D'autant plus que nous

avons rapidement été confrontés à des difficultés, notamment d'apprentissage et de formation qui nous ont fait prendre un certain retard.

Je pense que cela peut aussi être dû à un certain manque d'organisation au sein de l'équipe qui est malgré tout rapidement fonctionnelle lorsqu'elle est en marche. De plus, j'ai été confrontée à des difficultés de conceptualisation, notamment sur la partie : plan de salle, je ne savais pas vraiment par où commencer, ni même comment le gérer. D'une manière très générale, je peux aisément affirmer avoir gagné en compétence, plus en React qu'en Symfony je pense, mais n'ayant jamais vu ces Frameworks auparavant, cela m'a permis un gain de compétences. Mais je boucle ce projet sur une note assez négative, avec beaucoup de déception. J'aurais aimé en faire plus, pour moi, mais aussi pour l'équipe.

Dorian :

La réalisation de cette maquette a été très intéressante d'un point de vue projet car on est parti avec des clients, leur projet, leurs spécifications demandées et il fallait les comprendre pour correctement analyser et faire évoluer ce projet.

Ce que j'ai aimé de ce projet était le contexte (gestion de concert), qui permettait et impliquait de voir chaque facette de la partie que l'on travaillait (back ou frontend). Ce que j'ai le moins apprécié était le fait que notre emploi du temps et notre avancé ne nous a pas permis, une fois encore, d'aller au bout, mais le MVP était du coup bien meilleurs car il gardait le principal en enlevant le superflu (mais qui restait encore bien assez complet).

Mes difficultés ont principalement été des points de blocages futiles qui m'ont arrêté quelque heure mais que j'ai fini par régler au fil du temps (exemple : une date qui ne passait pas dans ma requête POST mais qui ne me retournait pas d'erreur, juste... disparut. J'ai compris plus tard qu'Api Platform n'écrivait pas en Snake Case mais en Camel Case).

Ce projet m'a permis de poursuivre mon évolution sur Symfony, malheureusement je n'ai pas pris le risque d'aller à l'inconnue (le frontend sur React) sachant que l'on avait un projet de groupe et qu'il fallait perdre le moins de temps possible. De même pour API Plateform, dans lequel je n'ai pas eu le temps d'aller explorer toutes les fonctionnalités qu'il proposait.

Jérémy :

J'ai utilisé pour la première fois React avec ce projet, c'est le deuxième Framework JS que je touche, le premier étant Angular. J'ai largement préféré React à Angular, déjà pour son installation plus simple et sa flexibilité. Découvrir React pendant un projet n'a évidemment pas été facile, mais je trouve que même si j'ai l'impression de ne pas avoir fait grand-chose, en tout pas autant que ce que je comptais faire, j'ai beaucoup progressé en React et cela m'a donné envie de continuer sur React et voir ce que je peux faire. J'ai aimé découvrir un Framework en même temps que le projet, et ainsi voir une amélioration au fil des jours, ainsi que la satisfaction de faire quelque chose qui fonctionne. En plus de React, j'ai aidé l'équipe pour les « déploiements » sur git, et l'utilisation du versionning dans un projet ou c'était obligatoire pour avancer rapidement. Le projet en lui-même a été très intéressant car très complet et directif dans le style « il n'y a plus qu'à coder ». Le MVP aussi était une bonne chose car comme nous étions beaucoup en retard, nous avons dû préparer un sprint

pour réaliser le plus efficacement possible le MVP. Je regrette d'avoir passé trop de temps sur l'initialisation du projet, sur des exercices React que j'ai fait.

Guillaume :

Grâce à ce projet j'ai pu monter en compétence en React et apprendre à consommer une API locale avec Axios (et toutes les difficultés que cela apporte). J'ai beaucoup aimé ce Framework même si je trouve qu'il est extrêmement permissif et nous laisse donc une possibilité d'erreur bien plus importante. Malheureusement, je n'ai pas su réaliser tous les composants qui m'étaient attribué et je le regrette. Je pense continuer la réalisation du MVP après la date de rendu afin de monter en compétence et me challenger. La réelle difficulté sur ce projet pour moi était la gestion du temps et le manque de compétences que j'ai dû acquérir au cours de ce projet. Fort heuresement ce projet m'a tout de même permis de mieux cerner les concepts de React ainsi que son fonctionnement. Je suis peut-être également devenu un meilleur intégrateur :)