Alexandre MALFREYT - Julien ROSSE

*Polytech Paris-Saclay*
*APP3 IM - TC-C*

# RAPPORT DE PROJET WEB

Projet *Amphi* Bien

Cours de Programmation Web

# Sommaire

# Le projet

## L'idée

➢ Un "Tripadvisor" des amphis

Permettre aux étudiants de partager des avis, évaluations, et conseils pour leur permettre de rechercher des espaces adaptés à leurs besoins. Les données seraient créées de manière communautaire, fortement inspirée du fonctionnement de Google Maps.

## Le cahier des charges

On a commencé le projet avec un brainstorming pour trouver le plus d'idées possibles pour le projet. Voici un extrait :

- Liste d'amphis
    - Différentes vues (Liste, Cards, Map)
    - Filtres de recherche avancés
- Page d'un amphi
    - Infos de base
      (*Nom, Photos, Localisation GPS (carte)*)
    - Caractéristiques
      *(Nombre de places, présence de prises, matériel à disposition, projecteur, micro, haut-parleurs, fenêtres, accessibilité en fauteuil, …)*
    - À proximité
      *(autres amphis, machines à café, toilettes, hébergements, RUs, cafets, salles de classes, bibliothèques et espaces de travail, points d'eau)*
- Commentaires
    - Note /20
    - Plusieurs facteurs *(qualité des sièges, température, luminosité, …)*
    - Possibilité de signaler des problèmes *(gravité, marquer comme réparé)*

- Page “Tendances”
    - Les amphis les mieux notés sur une période X dans un rayon Y
- Page utilisateur
    - Photo de profil, Nom, Pseudo, Badges *(par exemple “Contributeur - Niveau 5”)*

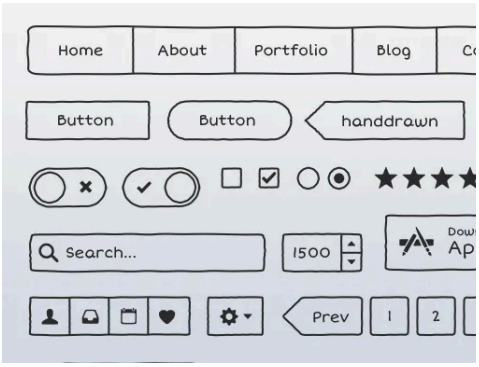
Nous n’avons pas eu le temps de tout implémenter

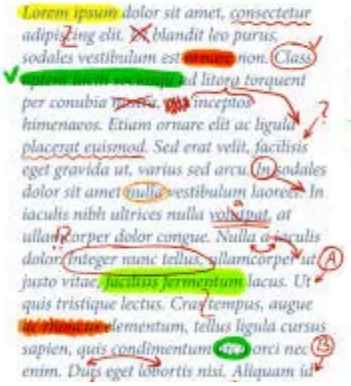
## La charte graphique

Pour la charte graphique, nous avons souhaité nous inspirer d’une ambiance qui rappelle les études, avec un style graphique inspiré de brouillons à crayon sur des feuilles de papier à carreaux.

### Inspirations

Nous avons cherché des images d’inspiration avant de commencer à développer pour partir sur la même idée lors du développement :



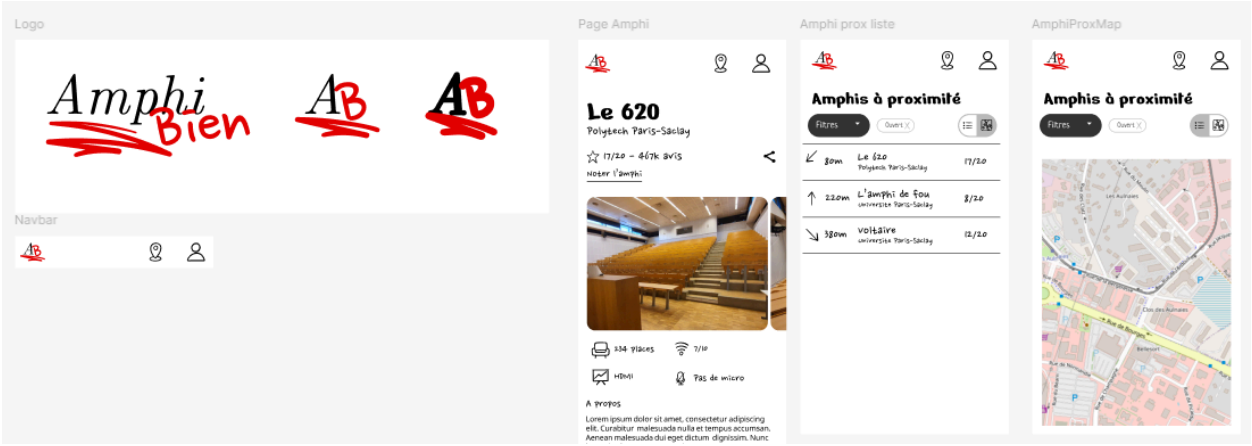



### Bibliothèques et polices d’écriture utilisées

- CSS Handcraft : https://bootswatch.com/sketchy/
- Surligner (CSS) : https://codepen.io/inegoita/pen/abdpaez
- Fonts : Cabin Sketch (Titres) & Neucha (Textes)

## Prototypage

Avant de commencer à coder le frontend, nous avons commencé par designer ce à quoi nous voulions que notre site ressemble dans les grandes lignes avec un prototype d'interface sur Figma :




## Résultat final

# La stack utilisée

## Gestion de versions (Git)

Le projet est géré via Git pour assurer une collaboration efficace (travail en parallèle sur des fonctionnalités différentes) et suivre les modifications apportées au code.

Lien : https://github.com/AlexZeGamer/amphi-bien

## Framework RedwoodJS

- **Langage** : TypeScript (JavaScript)
- **Front-end** : React.js
- **API** : GraphQL
- **Back-end** : Node.js
- **Base de données** : PostgreSQL avec l'ORM Prisma

L'utilisation d'une seule technologie (JavaScript/TypeScript) sur l'ensemble du projet permet une meilleure cohérence et facilite la maintenance. De plus, RedwoodJS dispose d'une base de projet avec une architecture de fichiers toute faite, nous permettant ainsi de nous concentrer sur le développement des pages et des fonctionnalités plutôt que de passer du temps à mettre en place le projet.

# L'Hébergement (bonus)

En plus du développement du site, nous avons souhaité apprendre à mettre en ligne le site avec un hébergement en ligne pour rendre le site accessible 24h/24 de manière professionnelle.

## Google Cloud Platform

Nous avons mis en place une infrastructure "serverless", afin de limiter les coûts de l'application lorsqu'elle est peu utilisée.

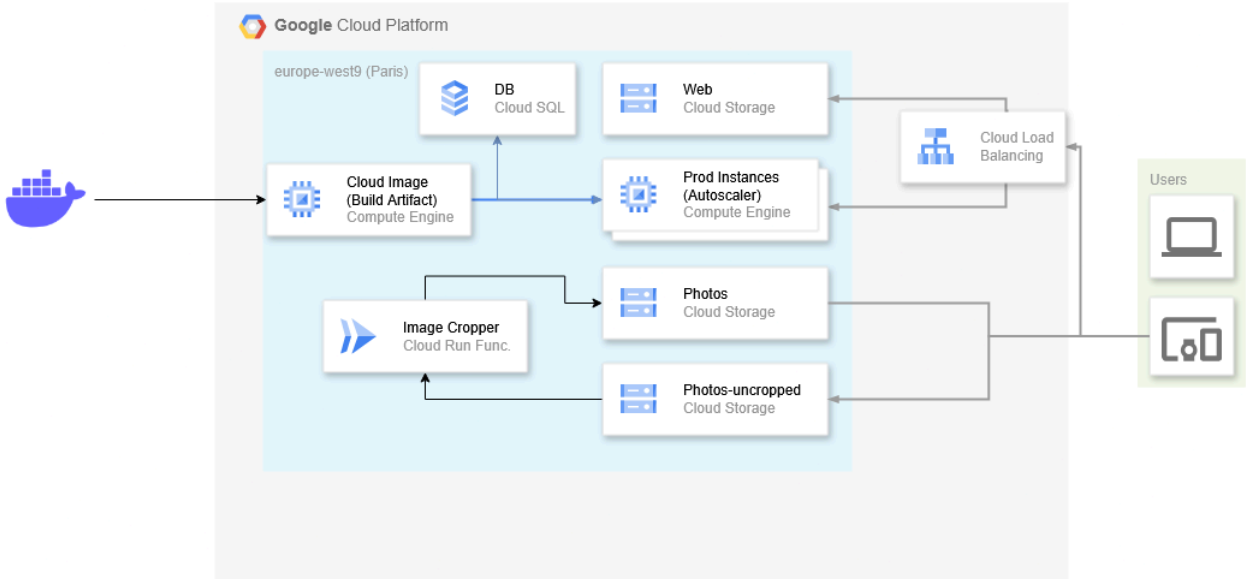



*Schéma d'infrastructure Google Cloud Platform*

### L'infrastructure serverless

L'infrastructure serverless permet de réduire les coûts d'exploitation en adaptant automatiquement les ressources utilisées à la charge réelle du site. Le code du backend est hébergé dans un container **Docker** qui est actif uniquement lorsque les utilisateurs font des requêtes sur le site.

Ainsi, lorsque le trafic est faible, les ressources allouées diminuent automatiquement, réduisant les frais. À l'inverse, en cas de forte affluence, l'infrastructure est capable de monter en puissance rapidement pour maintenir une expérience utilisateur optimale.

### La base de donnée

La base de données utilisée est une base de données **PostgreSQL** hébergée sur le cloud de Google, qui communique avec notre backend avec l'ORM **Prisma** intégré à RedwoodJS. L'utilisation de Prisma permet une gestion simplifiée des données notamment grâce à une sécurité accrue *(permet d'éviter les injections SQL par exemple)*, mais aussi car elle fonctionne de manière indépendante du SGBD utilisé *(ici PostgreSQL).*

### Les fichiers des utilisateurs (images)

Les fichiers mis en ligne par les utilisateurs d'AmphiBien sont uploadés sur Google Cloud dans un premier **bucket** qui stocke les fichiers bruts envoyés par les utilisateurs, puis un **script Python** les rogne à la taille maximale définie au préalable pour les rendre moins volumineux, afin d'économiser de l'espace de stockage ainsi que de charge le site plus rapidement pour les autres utilisateurs au chargement des pages.

Ces fichiers sont ensuite distribués via un système similaire à un **CDN (Content Delivery Network)**, permettant de servir rapidement les images aux utilisateurs où qu'ils se trouvent dans le monde. Cela permet d'économiser de l'espace de stockage et d'améliorer considérablement les temps de chargement pour les utilisateurs.

## L'authentification

L'authentification est gérée par le service **Firebase** de Google qui offre plusieurs avantages : il permet non seulement d'éviter d'avoir à gérer tous les processus d'authentification et de gestion des comptes utilisateurs, nous permet d'intégrer facilement l'autentification avec des services externes comme Google, mais permet aussi une sécurité accrue en nous évitant d'avoir des failles de sécurité sur ce système critique (notamment la gestion de mots de passe utilisateurs, en cas de piratage de notre base de données).

## Le nom de domaine

Le nom de domaine amphi-bien.fr a été acheté sur OVH, et lié à l'hébergement en configurant la **Zone DNS** avec des entrées A permettant de lier ce nom de domaine vers le **load balancer** de Google Cloud Platform.

## Pour aller plus loin : CI/CD

Pour automatiser la mise en ligne après chaque mise à jour du site, on aurait pu utiliser du **déploiement continu (CI/CD)** afin de lier le dépôt GitHub à l'infrastructure Google Cloud avec **GitHub Actions**, permettant d'automatiser les mises à jour et tout en réduisant le risque d'erreur humaine lors du déploiement.

# Les limites du projet

## Manque de temps

Au vu des contraintes temporelles et au temps alloué à ce projet, certaines des fonctionnalités souhaitées n'ont pas pu être implémentées.

Voici certains points qui n'ont pas pu être traités :

- Notations des amphis
- Implémentations des points d'intérêts externes (restauration, bibliothèque)
- Ajout des fonctionnalités d'amphi (projecteur, wifi...) dans la page d'édition d'amphi.
- Pages utilisateurs et aspect social

Le perfectionnisme de l'équipe a également ralenti la progression, ne permettant pas de finaliser toutes les fonctionnalités prévues initialement.

## Besoin d'un temps de formation

L'équipe a rencontré une difficulté d'adaptation technique initiale. Julien avait déjà une expérience préalable avec RedwoodJS, mais Alexandre découvrait totalement les frameworks web, nécessitant ainsi une période d'apprentissage supplémentaire.

Si c'était à refaire, nous aurions mieux réparti les tâches et commencé le développement plus tôt en parallèle de notre formation aux différentes technologies utilisées.