

Plans de la HEIG-VD interactifs

Travail de Bachelor

Département TIC

Filière Informatique et systèmes de communication

Orientation Informatique logicielle

Kylian Bourcoud

5 août 2022

Supervisé par :
Prof. Y. Chevallier (HEIG-VD)

Préambule

Ce travail de Bachelor (ci-après **TB**) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'École.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD
Le Chef du Département

Yverdon-les-Bains, le 5 août 2022

Authentification

Je soussigné, Kylian Bourcoud, atteste par la présente avoir réalisé seul ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Kylian Bourcoud

A handwritten signature in black ink, appearing to read "Kylian Bourcoud".

Yverdon-les-Bains, le 5 août 2022

Résumé

Ce rapport décrit le processus de création d'une application web de plans interactifs pour la HEIG-VD. Ce processus est séparé en cinq étapes : une introduction au contexte, une analyse des besoins des utilisateurs et des systèmes déjà existants, la conception d'une solution, la réalisation de celle-ci, et finalement une critique du travail accompli.



This report describes the creation process for a web application of interactive plans for the HEIG-VD school. This process is separated in five steps : an introduction of the context, an analysis of the users' needs and the existing solutions, the conception of a solution, its development and finally a review of the accomplished work.

Table des matières

| | |
|--|------------|
| Préambule | i |
| Authentification | iii |
| Résumé | v |
| 1 Introduction | 1 |
| 1.1 Contexte | 1 |
| 1.2 Description du problème | 2 |
| 2 Analyse | 3 |
| 2.1 Analyse fonctionnelle du système | 3 |
| 2.2 Etat de l'art WebSIG | 4 |
| 2.3 Données existantes à disposition | 9 |
| 3 Conception de la solution | 11 |
| 3.1 Exigences de la solution | 11 |
| 3.2 Solution envisagée | 11 |
| 3.3 Explications de certains principes | 11 |
| 3.4 Infrastructure | 12 |
| 3.5 Pipeline CI/CD | 13 |
| 3.6 Technologies utilisées | 14 |
| 3.7 Conception de la base de données | 16 |
| 3.8 Planification de la mise en place de la solution | 18 |
| 3.9 Design de la solution | 19 |
| 4 Réalisation de la solution | 23 |
| 4.1 Création des données géographiques | 23 |
| 4.2 Backend Serveur-api | 26 |
| 4.3 Frontend | 28 |
| 4.4 Déploiement | 34 |
| 5 Conclusion | 37 |
| 5.1 Critique de la solution | 37 |
| 5.2 Critique de la méthode de travail | 38 |
| 5.3 Implémentations futures | 40 |
| 5.4 Conclusion personnelle | 40 |
| Appendices | 43 |

TABLE DES MATIÈRES

| | |
|-----------------------------|-----------|
| A Planning | 43 |
| B Cahier des charges | 45 |

Table des figures

| | | |
|------|---|----|
| 1.1 | Plans actuels de la HEIG-VD | 1 |
| 2.1 | Schéma des cas d'utilisation | 4 |
| 2.2 | Plans interactifs de l'EPFL | 5 |
| 2.3 | Zoom sur les plans de l'EPFL | 6 |
| 2.4 | Géoportail de L'EPFL | 6 |
| 2.5 | Carte du campus du MIT | 7 |
| 2.6 | Plans du SITN | 7 |
| 2.7 | Plans du campus de Stanford | 8 |
| 2.8 | Plans de l'aéroport de Zurich | 8 |
| 3.1 | Architecture de l'infrastructure | 13 |
| 3.2 | Pipeline CI/CD | 14 |
| 3.3 | Schéma conceptuel de la base de données | 16 |
| 3.4 | Design global | 20 |
| 3.5 | Design de l'outil de changement d'étage | 20 |
| 3.6 | Design de la fenêtre d'informations | 21 |
| 3.7 | Design du menu de filtrage | 22 |
| 3.8 | Design de l'outil de recherche | 22 |
| 4.1 | Résultat final du géoréférencement d'un plan | 24 |
| 4.2 | Résultat final de la création d'informations de l'étage E | 24 |
| 4.3 | Résultat final de la création d'informations des ressources | 25 |
| 4.4 | Arborescence de fichiers du serveur-api | 26 |
| 4.5 | Diagramme de séquence représentant les communications pour le serveur-api | 27 |
| 4.6 | Affichage global de l'application | 28 |
| 4.7 | Organisation du code source | 28 |
| 4.8 | Schéma de la stratégie de récupération des données | 29 |
| 4.9 | Ecran de chargement | 30 |
| 4.10 | Outil de changement de bâtiments | 30 |
| 4.11 | Outil de changement d'étages | 31 |
| 4.12 | Outil de filtrage des ressources | 31 |
| 4.13 | Outil de changement d'affichage et effet sur les plans | 32 |
| 4.14 | Outil de recherche avec menu des suggestions | 32 |
| 4.15 | Fenêtre d'informations | 33 |
| 4.16 | Affichage sur smartphone en mode portrait | 34 |
| 4.17 | Affichage sur smartphone en mode paysage | 34 |
| 5.1 | Répartition du temps | 39 |

Liste des tableaux

| | | |
|-----|---|----|
| 2.1 | Liste des Besoins | 4 |
| 2.2 | Liste des Fonctionnalités | 5 |
| 2.3 | Tableau de comparaison entre Leaflet et Openlayers | 9 |
| 5.1 | Etat d'avancement des fonctionnalités du cahier des charges | 37 |
| 5.2 | Etat d'avancement des fonctionnalités supplémentaires | 37 |
| 5.3 | Tableau de comparaison entre planification et réalisation | 39 |

Chapitre 1

Introduction

1.1 Contexte

Les plans d'architecte ont été un support fondamental pour l'orientation des personnes au sein d'un bâtiment. Avec l'arrivée du web, l'usage d'applications orientées "plan" remplace peu à peu les plans physiques, offrant de nouvelles fonctionnalités plus interactives.

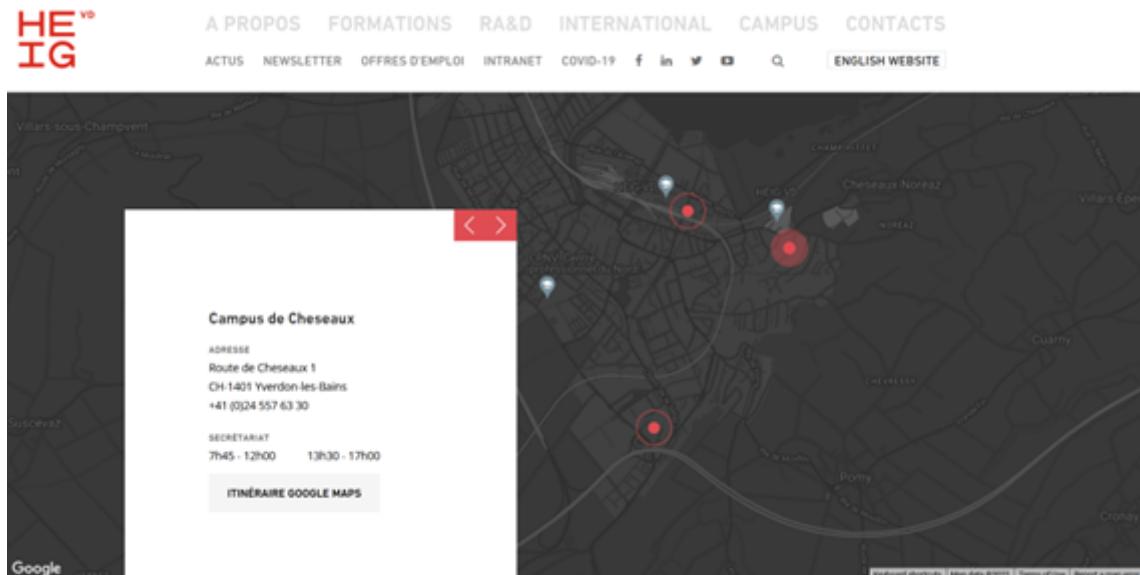


Figure 1.1 – Plans actuels de la HEIG-VD

La HEIG-VD est une haute école dispensant des formations dans différents domaines de l'ingénierie, ainsi que de la gestion d'entreprise. Elle se situe sur trois sites dispersés dans la ville d'Yverdon-Les-Bains, Vaud : Cheseaux, en périphérie de la ville, St-Roch proche de la gare et le Y-Parc, dans la zone industrielle.

Cette école fournit sur son site web un plan [HEI10] (voir Figure 1.1) indiquant l'emplacement des trois bâtiments principaux. On peut aussi y obtenir les informations sur les différents moyens d'accès à ces sites afin d'aider ses utilisateurs et utilisatrices à s'orienter. Cependant, elle ne fournit ni sur son site, ni dans le guide de l'étudiant, une interface permettant de s'orienter vers une salle ou une ressource précise.

1.2 Description du problème

M. Chevallier souhaite mettre à disposition des utilisateurs et utilisatrices une interface interactive pour permettre la visualisation des salles et des plans de la HEIG-VD pour les trois bâtiments du campus.

Chapitre 2

Analyse

2.1 Analyse fonctionnelle du système

L'analyse fonctionnelle est un processus qui permet de déterminer les fonctionnalités d'un système à partir d'une analyse des besoins utilisateurs.

2.1.1 Cas d'utilisation

La première étape est de déterminer les cas d'utilisation du futur système. Ceux-ci ont été établis lors de la conception du schéma des cas d'utilisation, Figure 2.1. Celui-ci est en notation UML, un des standards utilisés dans la modélisation d'application logiciel, et a été réalisé à l'aide de l'application web diagrams.net [dia05].

Ce schéma présente les cas d'utilisation suivants :

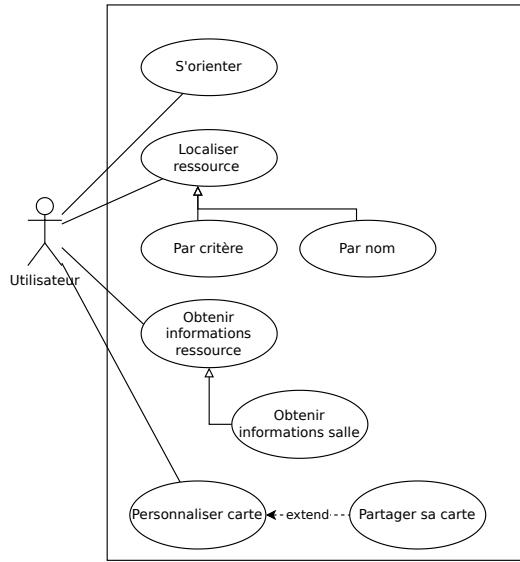
- Un utilisateur utilise le système pour s'orienter sur les différents sites de la HEIG-VD.
- Un utilisateur opère une recherche sur le système afin de localiser une ressource par des critères ou par son nom (une ressource est un terme générique pouvant symboliser une salle, l'emplacement d'un bureau d'un collaborateur, l'emplacement de matériel, etc.).
- Un utilisateur obtient des informations sur une ressource.
- Un utilisateur personnalise la carte et l'exporte pour d'autres usages.

2.1.2 Analyse des besoins

La deuxième étape de l'analyse fonctionnelle est de déterminer les besoins des utilisateurs à partir des cas d'utilisation du système. Pour ce système, les besoins ont été déterminés dans la Table 2.1. Ils ont été numérotés de N1 à N8 pour pouvoir s'y référer plus facilement par la suite.

2.1.3 Fonctionnalités du système

La troisième étape est de déterminer les fonctionnalités du système à partir des besoins des utilisateurs. Ils ont été listés dans la Table 2.2. De plus, les besoins auxquels les fonctionnalités répondent dans la Table 2.1 ont été précisés dans la colonne Besoins.

**Figure 2.1 – Schéma des cas d'utilisation**

| Besoin | |
|--------|--|
| N1 | S'orienter facilement à travers les sites de la HEIG-VD |
| N2 | Localiser une ressource à l'aide de son nom |
| N3 | Localiser une ressource à l'aide de critères |
| N4 | S'informer efficacement sur une ressource |
| N5 | Être capable de personnaliser une carte |
| N6 | Être capable de partager sa carte personnalisée |
| N7 | S'informer efficacement des noms des noms des locaux, de leur surface, et de leur type |
| N8 | Localiser facilement un collaborateur sur un plan des sites |

Table 2.1 – Liste des Besoins

2.2 Etat de l'art WebSIG

Dans cette section, différents systèmes d'interface web de visualisation de données géographiques (webSIG) vont être analysés, ainsi que les technologies associées à ce type d'application.

2.2.1 WebSIG existants

Plan EPFL

Les plans interactifs du campus de l'EPFL [EPF18b] affichent les bâtiments du campus en s'adaptant selon le zoom et l'étage sélectionné. Suivant le zoom, on peut visualiser le contour du site, puis le contour des bâtiments et enfin les salles des bâtiments (voir 2.3).

L'utilisateur a la possibilité de filtrer les points d'intérêts à afficher sur la carte à l'aide d'un menu sur la gauche du site. Il y a aussi la possibilité de rechercher différentes ressources en fonction de leur nom, comme les bâtiments, les salles, les personnes, les restaurants, les magasins, ou encore les espaces culturels. D'autres outils sont fournis, comme la recherche du plus court itinéraire entre deux ressources, un outil pour l'impression, ou un outil pour changer l'affichage pour une vue aérienne.

2.2. ETAT DE L'ART

| | Fonctionnalités | Besoins |
|-----|---|------------|
| F1 | Afficher un plan afin d'aider à l'orientation | N1, N7 |
| F2 | Utilisable facilement et de façon ergonomique | N1 |
| F3 | Fournir une orientation rapidement | N1 |
| F4 | Facilement accessible | N1 |
| F5 | Afficher les ressources désirées | N1, N3, N5 |
| F6 | Fournir un outil de tracé du plus court itinéraire pour l'orientation | N1 |
| F7 | Offrir un outil de localisation de ressource par nom | N2, N8 |
| F8 | Fournir un outil de localisation de ressource par critère | N3 |
| F9 | Fournir des informations sur les ressources | N4, N7 |
| F10 | Fournir un outil de dessin sur carte | N5 |
| F11 | Fournir un outil d'exportation | N5 |
| F12 | Fournir un outil d'impression de carte | N6 |
| F13 | Fournir un outil de partage de carte | N6 |
| F14 | Fournir un outil de sauvegarde de carte | N6 |

Table 2.2 – Liste des Fonctionnalités



Figure 2.2 – Plans interactifs de l'EPFL

Finalement, un lien permet d'accéder au Géoportail de l'EPFL.

La principale technologie utilisée pour le frontend est Ngeo (combine Angular js et openlayers, plus de détails dans la section technologie).

Géoportail EPFL

Le Géoportail de l'EPFL [EPF18a] est très similaire au plan du campus, mais il offre la possibilité de dessiner des formes vectorielles sur la carte. Il permet aussi d'afficher des ressources plus précises comme les réseaux wifi ou les prises électriques.

MIT campus map

Les plans interactifs du MIT [MIT13] affiche le tracé des bâtiments ainsi que le nom de ceux-ci. Seules les légendes s'adaptent en fonction du zoom.

Un utilisateur peut rechercher des bâtiments ou des points d'intérêts liés à l'université (ex : le world wide web consortium W3C).

Il peut aussi appliquer quelques filtres pour afficher des repères comme les restaurants. Cliquer sur un bâtiment permet d'obtenir des informations sur celui-ci.

CHAPITRE 2. ANALYSE



Figure 2.3 – Zoom sur les plans de l'EPFL

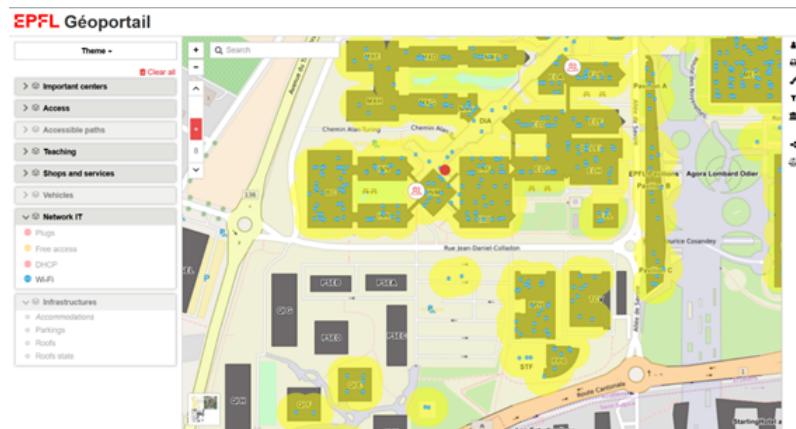


Figure 2.4 – Géoportail de L'EPFL

D'autres outils sont fournis comme un outil de partage ou d'impression.

L'affichage de la carte s'effectue à l'aide de l'API Google Maps, permettant un accès à Google Street View.

SITN - Géoportail du système d'informations du territoire neuchâtelois

Les plans du SITN [SIT10] affichent les différentes informations géographiques du canton de Neuchâtel. Le niveau de détail de la carte s'adapte en fonction du zoom. Par exemple les numéros des maisons selon le cadastre de chaque commune s'affichent lors d'un zoom à une échelle 1 :1000.

Il y a la possibilité d'appliquer plusieurs filtres afin d'obtenir les informations recherchées comme le tracé des communes ou les points d'intérêts.

L'outil offre aussi des outils de dessin vectoriel, d'impression, la possibilité de changer le fond du plan, un accès à Google Street View, et un accès au géoportail LIDAR.

Le site utilise GeoMapFish pour l'affichage des cartes ainsi que l'API Google Maps pour Google Street View.

Stanford Campus Map

La carte du campus de Stanford [Sta10] affiche le tracé des bâtiments ainsi qu'une légende précisant le nom de ceux-ci. Seul l'affichage de la légende varie selon lors d'un zoom rapproché.

2.2. ETAT DE L'ART

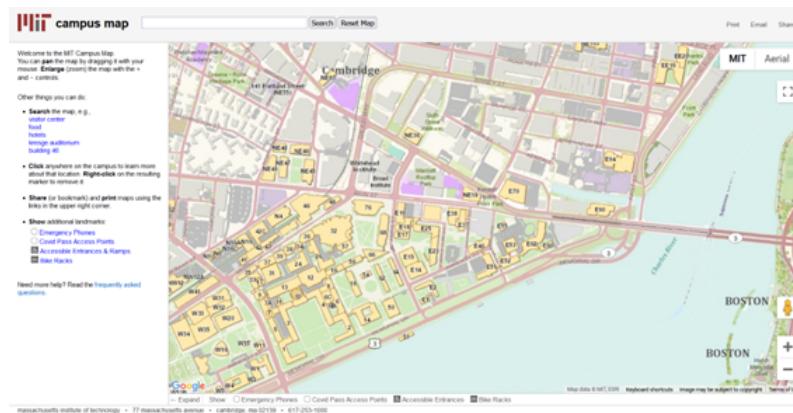


Figure 2.5 – Carte du campus du MIT

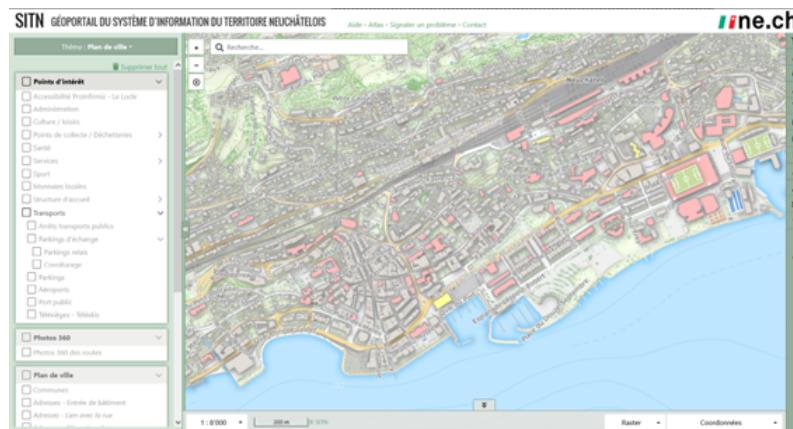


Figure 2.6 – Plans du SITN

Le site offre un outil de recherche de ressources aux utilisateurs, un accès à Google Street View, un outil d'impression et un outil de partage.

A noter que le menu en bas à droite est un mélange de plusieurs fonctionnalités, ce qui peut rendre l'utilisation confuse.

L'application web utilise l'API Google Maps pour l'affichage de la carte.

Aéroport de Zurich

L'aéroport de Zurich [aér10] offre un plan non géoréférencé en 3D isométrique. Les différents points d'intérêts affichés varient en fonction du zoom. L'application offre un outil de recherche avec des suggestions par thème. On peut aussi changer le fond du plan en fonction de l'étage.

Les plans ne servent qu'à l'orientation des voyageurs et sont donc pauvre en fonctionnalités annexes.

Le site utilise le framework réactif React js, ainsi que le module bundler webpack. Les plans sont des images bitmap (pixellisées).

2.2.2 Conclusions de l'analyse des systèmes existants

On peut distinguer deux types d'applications : des plans interactifs aidant les utilisateurs à s'orienter, ainsi que les Géoportails qui fournissent des informations géogra-

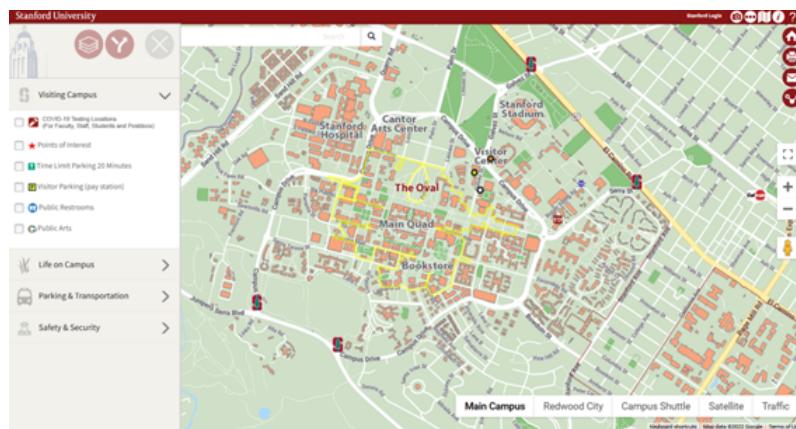


Figure 2.7 – Plans du campus de Stanford

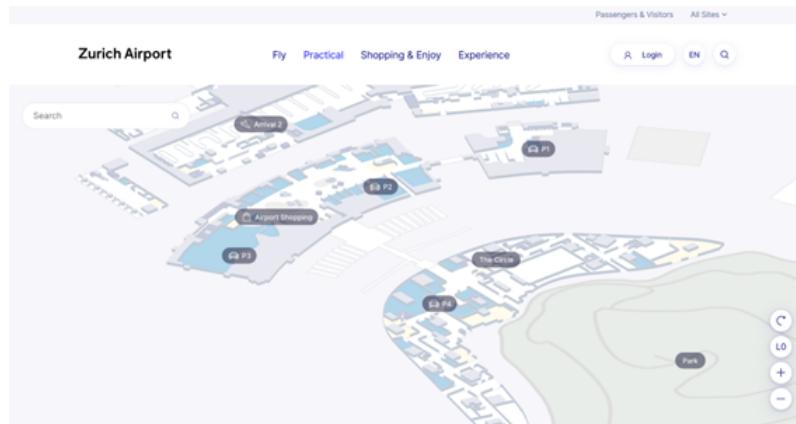


Figure 2.8 – Plans de l'aéroport de Zurich

phiques et permettent de construire de nouvelles données à l'aide d'outils de dessin vectoriel.

Cette distinction a été faite par l'EPFL qui distingue les deux types d'application dans deux sous-domaines.

Les technologies utilisées sont aussi différentes : les sites les plus complets utilisent souvent GeoMapFish alors que les sites les plus simples utilisent plutôt l'API Google Maps.

2.2.3 Technologies pour les webSIG

Un webSIG est une application web qui présente des données géographiques sur des plans. Cette section présente plusieurs technologies qui facilitent la mise en place de tels systèmes.

Openlayers

Openlayers [Ope13] est une librairie JavaScript open source qui aide à la construction d'applications webSIG. Elle permet de rajouter des calques contenant des données géographiques au-dessus d'un fond de carte.

2.3. DONNÉES EXISTANTES À DISPOSITION

Leaflet

Leaflet [Lea11] est une librairie open source concurrente à Openlayers. Elle est simple d'utilisation et légère. On peut étendre la librairie à l'aide de plugins.

2.2.4 Comparaison entre Leaflet et Openlayers

| Point de comparaison | Leaflet | Openlayers |
|-----------------------------|----------------|---------------------------------|
| Simplicité d'utilisation | simple | moyen |
| Poids | léger | lourd |
| Flexibilité | peu flexible | très flexible |
| Nombre de formats supportés | un seul format | plusieurs formats et protocoles |

Table 2.3 – Tableau de comparaison entre Leaflet et Openlayers

On peut déduire que Leaflet est plus adapté à de petites applications ou applications mobiles webSIG alors que openlayers est plus adapté à des applications plus complexes.

Google Maps API

L'API de Google Maps [Goo05] n'est pas open source et peut être payante. Elle n'offre pas la possibilité de rajouter des calques au fond de carte, mais uniquement des points d'intérêts. Elle n'est donc pas adaptée à la construction d'applications webSIG.

GeoMapFish

GeoMapFish est une technologie open source développée par l'entreprise campto-camp facilitant la construction de webSIG. Elle est composée de Ngeo pour le frontend [cam14] et de c2cgeoportal [cam12] pour le backend.

Ngeo est une librairie JavaScript basée sur le Framework réactif Angular Js et l'API Openlayers.

C2cgeoportal est la partie serveur construite à partir d'une image Docker. Il faut avoir des connaissances en Python pour l'utiliser.

Cette technologie est malheureusement obsolète car elle utilise une vieille version d'Angular qui est déconseillée pour le développement de nouveaux projets.

PostGreSQL et PostGis

PostGreSQL est une base de données relationnelle open source. Elle permet de stocker des données géographiques avec l'extension PostGis [Pos05]. Cette solution est utilisée par c2cgeoportal et dans de nombreuses autres applications webSIG.

2.3 Données existantes à disposition

2.3.1 Plans

Des plans des étages des sites de Cheseaux et St-Roch non géoréférencés ont été fournis au format DGW, format propriétaire de AutoCAD, un logiciel de dessin technique.

2.3.2 Serveur LDAP

Un serveur LDAP est un service d'annuaire numérique. La HEIG-VD possède un tel annuaire qui pourrait être utilisé pour acquérir des informations sur certaines ressources.

Chapitre 3

Conception de la solution

3.1 Exigences de la solution

Suite à l'analyse, nous pouvons établir qu'il faudra mettre en place une interface interactive de visualisation des plans comprenant uniquement le bâtiment de Chezeaux. Celle-ci devra afficher toutes les salles du site avec leurs noms, un qualificatif (secrétariat, salle de cours, etc.) et leur surface en mètres carrés. Un outil de changement d'étage permettra de parcourir les différents étages du site. L'interface sera disponible tant sur de grand (télévision) que sur de petit écran (téléphone mobile).

Cette application sera hébergée sur une machine virtuelle fournie par l'école. Elle comportera une base de données qui s'occupera de stocker les données utilisées. Un serveur API récupérera les informations et les enverra à l'utilisateur.

Elle positionnera aussi certaines ressources comme les collaborateurs sur le site de la HEIG-VD.

Si le temps le permet, un outil de filtrage des ressources à afficher et/ou un outil de recherche sera aussi développé.

La solution a pour but de démontrer les possibilités qu'elle offrirait à l'orientation des collaborateurs sur les différents sites de la HEIG-VD. Elle ne sera pas une solution utilisable par l'école.

3.2 Solution envisagée

Afin de répondre aux exigences, il a été défini que l'application sera un site web, car c'est le moyen le plus accessible pour fournir cette interface, autant sur grand que petit écran, et sans obliger l'utilisateur à télécharger un logiciel au préalable.

Elle sera contenue sur une seule page web (single-page application) et offrira les différentes fonctionnalités à travers différents menus. L'affichage se modifiera en fonction de son utilisation.

3.3 Explications de certains principes

Cette section explique les principes du web, d'un reverse-proxy, des containers et de la base de données. Vous pouvez ignorer les sous-sections si vous avez des connaissances dans ces concepts.

3.3.1 Web

Il ne faut pas confondre internet, le standard qui régit la communication entre les ordinateurs et le web qui se base sur internet et est utilisé par les navigateurs pour afficher des applications. Le second utilise les protocoles de communications HTTP et HTTPS afin d'envoyer et récupérer des données.

Lorsqu'un utilisateur entre une URL, comme www.google.ch, dans un navigateur web, il va demander à un serveur de lui envoyer plusieurs fichiers dans des formats différents afin de permettre l'affichage de l'application web.

Sont envoyés au navigateur des fichiers HTML qui décrivent le contenu de l'application, des fichiers CSS qui changent l'aspect de ce contenu, des fichiers JavaScript, un langage de programmation qui modifie le contenu selon les interactions de l'utilisateur avec l'application, et les images à afficher.

En fonction de l'utilisation de l'application, celle-ci devra demander des données supplémentaires sur une ressource pour son fonctionnement. Celles-ci sont envoyées dans des formats standardisés comme JSON ou XML.

3.3.2 Reverse-proxy

Un Reverse-proxy est un serveur qui permet d'accéder à d'autres serveurs en utilisant la même url. Cette machine va trier les requêtes des utilisateurs et envoyer ces dernières sur le serveur correspondant le mieux. Elle peut aussi refuser une requête si aucun serveur ne peut la traiter. Utiliser un Reverse-proxy amène aussi plus de sécurité car on n'expose pas les autres serveurs à l'extérieur.

3.3.3 Container

Un container est une entité qui contient une application ainsi que les autres programmes nécessaires à son fonctionnement. Il est indépendant du système d'exploitation et permet d'installer les applications sur n'importe quelle machine en évitant les conflits provoqués par le système d'exploitation. Pour exécuter un container, il faut auparavant installer un logiciel gérant ceux-ci sur le système d'exploitation.

On peut aussi créer des images de container. Celles-ci sont des fichiers regroupant les instructions qui permettent de construire les containers. Elles peuvent être facilement dupliquées et partagées sur le web à travers des hébergeurs appelés container registry.

3.3.4 Base de données

Une base de données est un programme qui va organiser le stockage de données sur une machine. Elle va simplifier l'obtention des données, l'ajout de nouvelles données, la modification et/ou la suppression de données existantes.

3.4 Infrastructure

3.4.1 Architecture prévue

L'infrastructure (voir Figure 3.1) tourne sur une machine virtuelle de la HEIG-VD et est composée de quatre containers :

- Le reverse-proxy qui renvoie les requêtes vers le serveur ou le serveur-api.
- Le serveur qui envoie les fichiers HTML, CSS, et JavaScript au client.
- Le serveur-api qui récolte les données depuis la base de données et les renvoie au client.

3.5. PIPELINE CI/CD

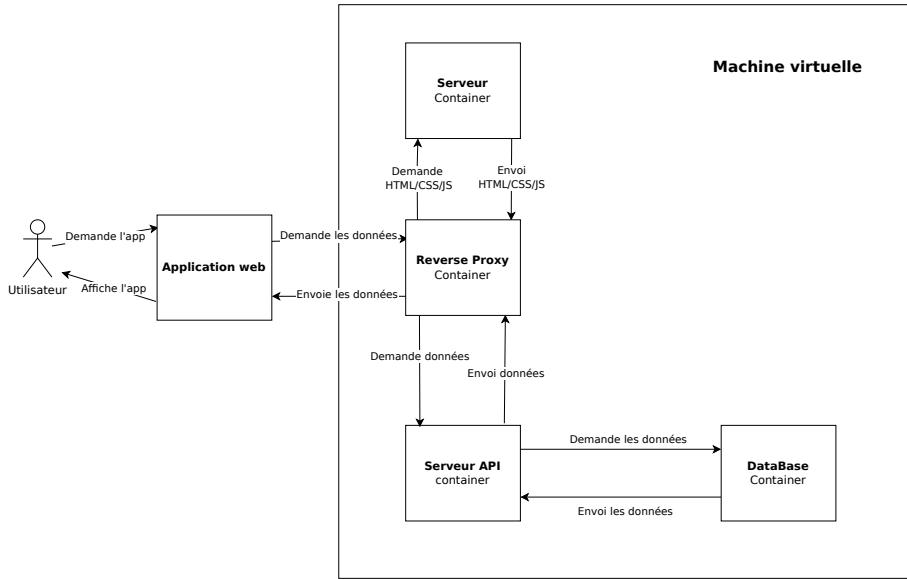


Figure 3.1 – Architecture de l'infrastructure

- La base de données qui stocke les informations.

Lorsqu'un utilisateur va demander l'accès à l'application en utilisant l'url "https://tb22-bourcoud.einet.ch/", il va accéder au reverse-proxy. Celui-ci va renvoyer la requête vers le serveur. Ce dernier sera chargé d'envoyer les fichier HTML, CSS, JavaScript ainsi que les images au client en repassant par le reverse-proxy.

Par la suite, l'application a besoin d'obtenir les données des plans à afficher. Elle réac-cède au reverse-proxy avec une adresse commençant par "https://tb22-bourcoud.einet.ch/api/", et la requête est renvoyée au serveur-api. Ce dernier récolte les données demandées depuis la base de données et les renvoie au client en repassant par le reverse-proxy.

3.4.2 Architecture idéale

L'architecture mentionnée dans la sous-section précédente est valide pour ce travail de Bachelor. Cependant, si l'école voulait mettre en place une solution plus aboutie, il faudrait séparer chaque container dans des machines virtuelles différentes et permettre aux serveur et serveur-api de se répliquer automatiquement ou d'effacer des réPLICATION en fonction du nombre d'accès à l'application (scalabilité horizontale élastique). Cela permettrait aussi de faire face à la panne d'un des containers, car d'autres maintiendraient alors le service.

3.5 Pipeline CI/CD

Un pipeline CI/CD est une automatisation des opérations à exécuter sur les applications pour qu'elles soient mises à la disposition des utilisateurs.

Le pipeline mis en place (voir Figure 3.2) va créer les images des containers du serveur et du serveur-api. Les deux autres containers, vus à la section précédente, sont créés à partir d'images déjà disponibles.

Ce pipeline va d'abord tester les deux applications afin d'éviter que des problèmes surviennent. Ensuite, il va construire les images et les publier sur un container registry.

CHAPITRE 3. CONCEPTION DE LA SOLUTION

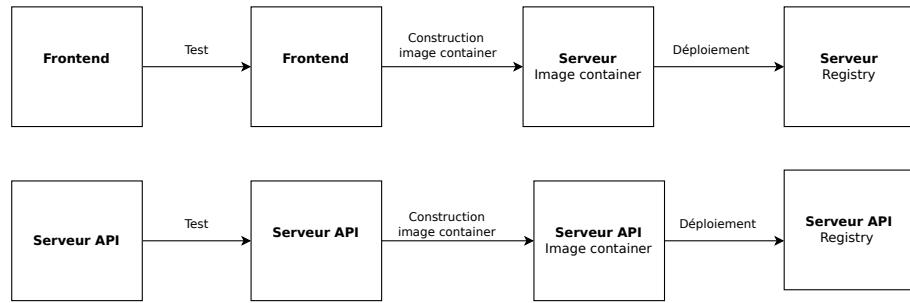


Figure 3.2 – Pipeline CI/CD

3.6 Technologies utilisées

3.6.1 Node et NPM

Node est un programme qui permet d'utiliser le langage de programmation JavaScript en dehors des navigateurs web.

Celui-ci est très utilisé chez les développeurs et possède de nombreuses librairies.

NPM est un programme qui gère les différentes librairies de Node. Il permet d'ajouter, de mettre à jour, et de supprimer des librairies.

3.6.2 TypeScript

TypeScript est un langage de programmation reprenant comme base JavaScript et lui ajoutant de nouvelles fonctionnalités utiles pour le développement. Il se transforme (transpile) par la suite en JavaScript lors de la construction de l'application web.

3.6.3 Vue 3 composition

Vue 3 est un framework JavaScript permettant de simplifier l'écriture des interactions avec le contenu de l'application web. Il est aussi chargé de créer du code HTML et CSS susceptible d'être facilement modifié. Il offre deux styles d'écriture de code : composition ou option. Ce projet a été écrit en utilisant le style "composition".

La librairie utilise une abstraction appelée "component" (mot anglais de composant) afin de créer les différents éléments de l'interface de l'application.

C'est un des frameworks les plus utilisés dans le développement d'applications web. Il a l'avantage d'être plus complet et compréhensible que React, et nécessite aussi moins de code qu'Angular pour arriver à un résultat similaire.

3.6.4 Pinia

Pinia est une librairie associée à Vue 3 permettant de simplifier la communication de données entre les différents composants. Elle est recommandée par les développeurs de la librairie Vue 3 pour cette tâche.

3.6.5 Openlayers

Openlayers est une librairie JavaScript permettant d'implémenter des webSIG.

3.6. TECHNOLOGIES UTILISÉES

3.6.6 Vite

Vite est un environnement de développement qui permet de simuler un serveur pour le développement en local, et de construire (compiler) l'application tout en optimisant la place en mémoire qu'elle prendra.

Cet environnement est très vite installé et simplifie la compilation du programme. Il permet aussi de créer un projet avec Vue 3 déjà intégré.

3.6.7 Vitest

Vitest est la librairie de test unitaire, conseillé par l'équipe de Vue 3 (permet de tester des parties du code indépendamment du reste). Elle est optimisée pour fonctionner avec Vite mais peut être utilisée dans d'autres projets.

3.6.8 Vue test Utils v2

Vue test Utils v2 est la librairie de test officielle de Vue 3 pour les composants.

3.6.9 Nginx

Nginx est le serveur qui hébergera les fichiers HTML, CSS et JavaScript de l'application.

3.6.10 Express

Express est une librairie simplifiant la mise en place d'un serveur sur Node. Il est utilisé pour créer le serveur-api.

3.6.11 Traefik

Traefik est le programme servant de Reverse-proxy pour l'infrastructure du projet. Il est plus simple à mettre en place par rapport à ses concurrents.

3.6.12 PostgreSQL et PostGis

PostgreSQL est la base de données utilisée pour le projet. Elle est open source et peut être étendue par PostGis. Cette extension rajoute la possibilité de stocker des données géographiques.

3.6.13 Docker

Docker est le logiciel qui gérera les containers sur la machine virtuelle.

3.6.14 Git et GitLab

Git est un programme permettant de facilement créer des versions du code source, revenir à une version précédente si besoin, et héberger son code sur des sites dédiés.

GitLab est un hébergeur de code source utilisant l'utilitaire Git. Il permet aussi la collaboration entre plusieurs développeurs sur une même application.

3.6.15 GitLab CI/CD

GitLab CI/CD est la fonctionnalité de l'hébergeur GitLab qui permet de lancer un pipeline CI/CD qui construira les deux images de container du projet.

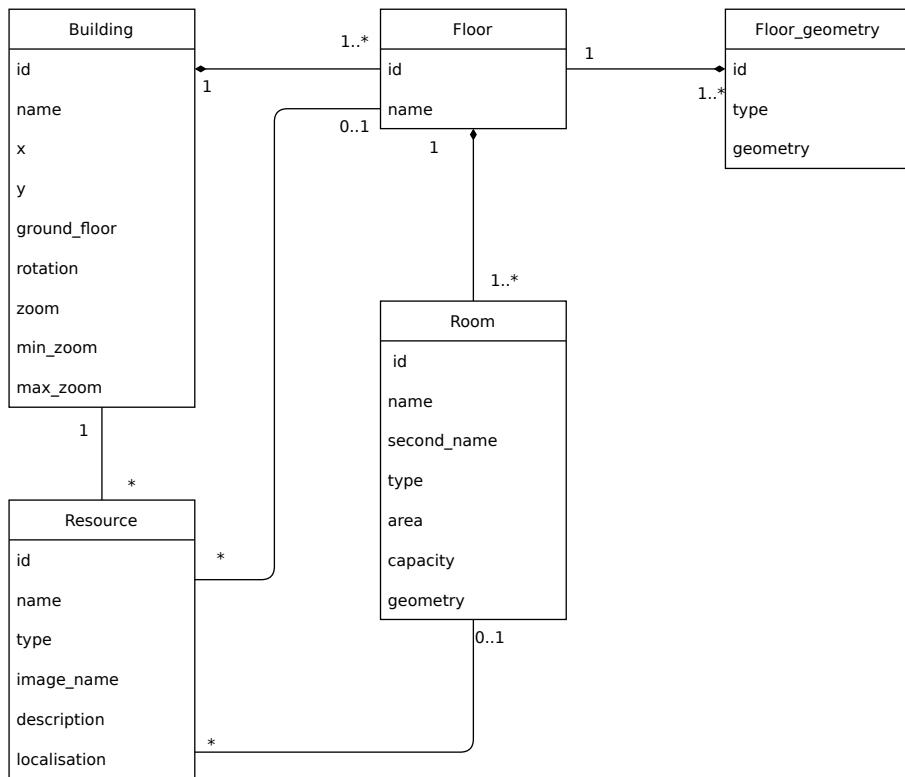


Figure 3.3 – Schéma conceptuel de la base de données

3.6.16 GitLab container registry

GitLab container registry est l'hébergeur des deux images de container.

3.7 Conception de la base de données

3.7.1 Base de données relationnelle

Une base de données relationnelle est un type de base de données. Au lieu de regrouper les données en une seule table, elle permet d'en créer plusieurs et de former des liens entre elles. Cela permet d'éviter la duplication de données.

A titre d'exemple, si on veut stocker tous les numéros de salle d'un bâtiment, tout en précisant à quel étage la salle appartient, si on utilise une seule table la mention du nom de l'étage apparaitra sur plusieurs lignes. Par contre, si on utilise une base de données relationnelle, on peut créer une table pour les salles, une table pour les étages et former un lien entre les deux.

Cela permet aussi d'éviter les erreurs lors d'insertions de données.

3.7.2 Schéma et choix

La conception de la base de données est représentée dans la Figure 3.3. Ce schéma a été conçu en notation UML. On y aperçoit les cinq tables qui seront décrites ci-après. Si on étudie bien chaque table, on peut remarquer que certains champs auraient pu être séparés sur d'autres tables afin d'éviter des duplications. Par exemple, le champ type dans room.

Il a été choisi de laisser ces duplications car cela permet de simplifier le transfert

3.7. CONCEPTION DE LA BASE DE DONNÉES

de données géographiques sous format geoJson (standard de fichier JSON pour le stockage de données géographiques).

3.7.3 Tables et relations

Building

La table Building représente un bâtiment et est composée des champs suivants :

- id : l'identifiant unique du bâtiment.
- name : le nom du bâtiment.
- x : la coordonnée x pour centrer le bâtiment sur une projection ESPG : 3857.
- y : la coordonnée y pour centrer le bâtiment sur une projection ESPG : 3857.
- ground_floor : le nom de l'étage du rez-de-chaussée.
- rotation : la rotation par rapport au nord pour une meilleure visualisation du bâtiment.
- zoom : le zoom pour Openlayers pour une meilleure visualisation du bâtiment.
- min_zoom : le zoom minimum pour Openlayers que pourra effectuer l'utilisateur.
- max_zoom : le zoom maximum pour Openlayers que pourra effectuer l'utilisateur.

La table possède les relations suivantes :

- Floor : le bâtiment possède plusieurs étages.
- Resource : le bâtiment peut posséder plusieurs ressources.

Floor

La table Floor représente un étage d'un bâtiment et est composée des champs suivants :

- id : l'identifiant unique de l'étage.
- name : le nom de l'étage.

La table possède les relations suivantes :

- Building : l'étage appartient à un bâtiment.
- Resource : l'étage peut posséder plusieurs ressources.
- Floor_geometry : l'étage est composé de plusieurs données géométriques

Floor_geometry

La table Floor_geometry représente des données géométriques associées à un étage et est composée des champs suivants :

- id : l'identifiant unique de la donnée géométrique.
- type : le type de géométrie.
- geometry : la donnée géométrique.

La table possède la relation suivante :

- Floor : les données géométriques appartiennent à un étage.

Room

La table Room représente une salle d'un bâtiment et est composée des champs suivants :

- id : l'identifiant unique de la salle.
- name : le nom de la salle (ex : E01).

CHAPITRE 3. CONCEPTION DE LA SOLUTION

- second_name : le second nom de la salle s'il y en a un (ex : marketing). Il peut ne pas être renseigné.
- type : le type de salle (ex : salle de cours). Il peut ne pas être renseigné.
- area : la surface de la salle. Elle peut ne pas être renseignée.
- capacity : le nombre de places d'une salle. Il peut ne pas être renseigné.
- geometry : les données géométriques de la salle.

La table possède les relations suivantes :

- Floor : la salle appartient à un étage.
- Resource : la salle peut posséder plusieurs ressources.

Resource

La table Resource représente tous types de ressources comme les extincteurs, les toilettes ou les ascenseurs que l'on peut trouver dans un bâtiment, un étage, une salle. Elle est composée des champs suivants :

- id : l'identifiant unique de la ressource.
- name : le nom de la ressource.
- type : le type de la ressource.
- image_name : le nom de l'image associé à la ressource.
- description : la description de la ressource.
- localisation : la localisation de la ressource dans une projection ESPG : 3857.

La table possède les relations suivantes :

- Building : la ressource est associée à un bâtiment.
- Floor : la ressource peut être associée à un étage.
- Room : la ressource peut être associée à une salle.

3.8 Planification de la mise en place de la solution

Ce chapitre présente la planification du projet et les principales étapes.

3.8.1 Echéances et étapes

Les principales échéances de ce travail sont le rendu d'un cahier des charges le jeudi 14 avril 2022, le rendu d'un rapport intermédiaire le lundi 16 mai 2022, le rendu final du projet initialement prévu le vendredi 29 juillet 2022 mais repoussé au vendredi 5 août 2022, et finalement une défense qui aura lieu entre le 22 août et le 16 septembre 2022.

Il y a deux grandes étapes prévues : une première version servant de proof of concept achevée le dimanche 29 mai 2022, et la version finale du projet prévue pour le 29 juillet 2022. Ces versions sont décrites dans les sous-sections suivantes.

D'autres versions pourront être mises en place par la suite afin de rajouter des fonctionnalités, mais cela demanderait plus de temps que ce qui a été planifié pour le travail de Bachelor.

3.8.2 Elaboration du projet

La première étape consistait en l'analyse d'informations sur les projets webSIG. Pour cela, 16 heures ont été planifiées. Ensuite, le travail consistait en la rédaction d'un cahier des charges et d'un planning. Pour ce faire, 25 heures ont été planifiées.

3.9. DESIGN DE LA SOLUTION

3.8.3 Première version du projet

La première étape était la prise en main des différentes technologies afin de vérifier la faisabilité du projet. Le but était de mettre en place une application webSIG minimalistique n'affichant qu'un étage avec ses salles et n'offrant aucune fonctionnalité supplémentaire.

La réalisation de cette version nécessiterait de :

- Traiter un des plans et créer les données géographiques (24 heures).
- Concevoir le design de l'application (8 heures).
- Mettre en place un pipeline CI/CD (4 heures).
- Création du serveur-api et de la base de données (16 heures).
- Création du frontend (24 heures).

3.8.4 Deuxième version du projet

La seconde étape mène à la mise en place de la solution finale.

La réalisation de cette version nécessiterait de :

- Traiter les plans du site de Cheseaux et créer les données géographiques (54 heures).
- Mettre en place le serveur-api et de la base de données (30 heures).
- Mettre en place le frontend (70 heures).
- Déploiement de l'application sur la machine virtuelle (30 heures).

3.8.5 Documentation du travail

La documentation du travail étant importante, il a fallu planifier celle-ci :

- Documentation du code (20 heures).
- Rapport intermédiaire (10 heures).
- Rapport final, (40 heures).
- Présentation (5 heures).

3.9 Design de la solution

Cette section décrit les designs imaginés pour la solution. Ceux-ci ne sont pas précis et sont susceptibles de changer lors de l'implémentation. De plus, des outils comme le menu de filtrage ou la recherche risquent de ne pas être implémentés.

3.9.1 Design global

Le design global (voir Figure 3.4) tend à respecter la charte graphique du site heig-vd.ch/ en utilisant son logo, et son code couleur. Le design a été conçu pour un écran au format 16/9, mais l'application sera aussi disponible pour d'autres types d'écrans. L'application sera sur une seule page sans barre de défilement principal (des barres de défilement secondaires seront possibles pour des fonctionnalités comme la fenêtre d'informations).

Cette page permet d'accéder à plusieurs outils :

- Un menu pour filtrer les informations à afficher en cliquant sur le bouton en haut à gauche.
- Un outil de recherche à droite de l'en-tête.
- Un outil de changement d'étage à droite de la fenêtre.

CHAPITRE 3. CONCEPTION DE LA SOLUTION

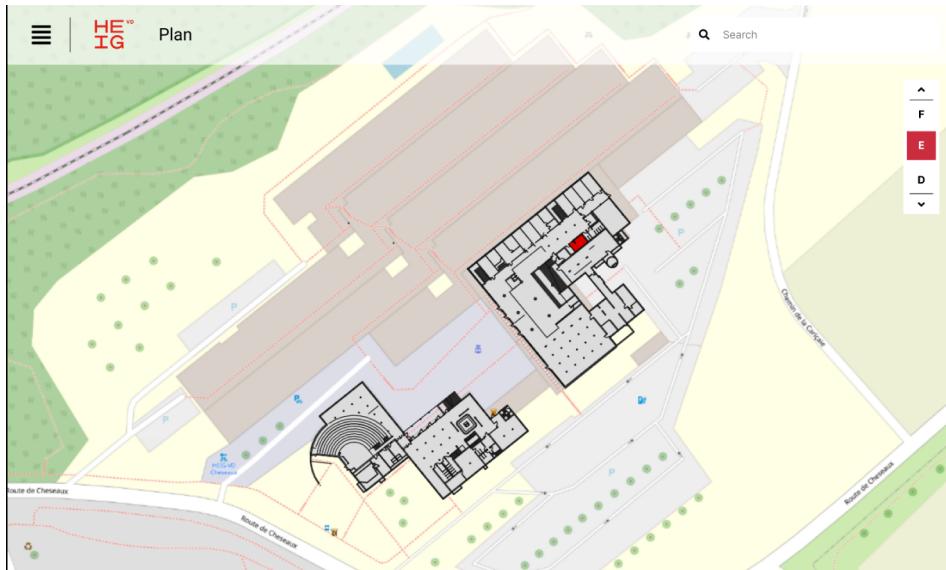


Figure 3.4 – Design global

- Une fenêtre d’informations en cliquant sur une salle ou en ayant effectué une recherche.

3.9.2 *Outil de changement d’étage*

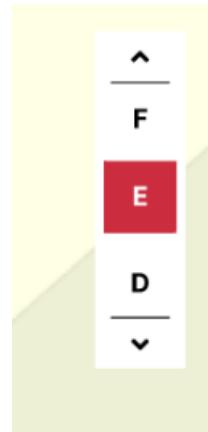


Figure 3.5 – Design de l’outil de changement d’étage

L’outil de changement d’étage (voir Figure 3.5) permet de modifier l’étage affiché sur le plan. Il indique, écrit en blanc sur fond rouge, l’étage actuel auquel se trouve l’utilisateur. On pourra modifier celui-ci en cliquant sur les flèches. L’outil indique également les étages adjacents afin d’éviter une mauvaise utilisation de l’outil.

3.9.3 *Fenêtre d’informations*

La fenêtre d’informations (voir Figure 3.6) s’affichera sur le bas, lors d’un clic sur une salle ou après avoir exécuté une recherche. Elle présentera les informations concernant la salle. Pour la fermer, l’utilisateur pourra cliquer en dehors de la fenêtre.

3.9. DESIGN DE LA SOLUTION



Figure 3.6 – Design de la fenêtre d’informations

3.9.4 Menu de filtrage

Le menu de filtrage (voir Figure 3.7) s'affichera sur la gauche de l'écran lorsque l'utilisateur cliquera sur le bouton en haut à gauche. Il permettra d'ajouter ou de retirer des éléments à afficher sur la carte. Pour sortir du menu, l'utilisateur pourra cliquer en dehors de celui-ci, ou à nouveau sur le bouton.

3.9.5 Outil de recherche

Lorsque l'utilisateur commencera à entrer des caractères dans le formulaire de recherche (voir Figure 3.8), celui-ci proposera un menu de suggestions. Si l'utilisateur clique sur la touche entrée ou sur une des suggestions, l'affichage du plan se modifiera pour afficher la ressource demandée.

CHAPITRE 3. CONCEPTION DE LA SOLUTION

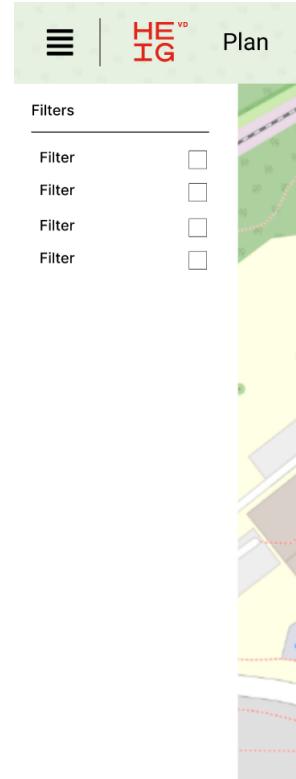


Figure 3.7 – Design du menu de filtrage

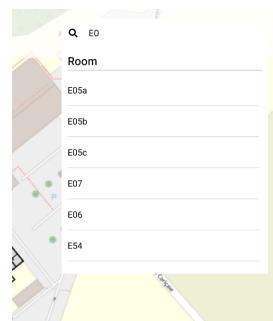


Figure 3.8 – Design de l'outil de recherche

Chapitre 4

Réalisation de la solution

4.1 Cration des donnees gographiques

Les donnees gographiques doivent tre traites, ou crees. Cette section explique le processus de travail afin d'obtenir ces informations, ainsi que leur stockage dans la base de donnees.

La solution proposee dans cette section fera l'objet d'une critique dans le chapitre "Conclusion".

4.1.1 QGis

QGis est un logiciel de systeme d'informations gographiques qui permet de visualiser, traiter et diffuser ces informations. Il a te utilise dans ce projet afin de traiter les plans, et crer les autres donnees gographiques.

Une initiation  l'utilisation du logiciel a te ncessaire afin de mettre en oeuvre une solution pour le projet.

4.1.2 Les plans

Les plans mis  disposition sont stockes dans des fichiers DWG, un format proprietaire du logiciel de dessin technique AutoCAD utilise pour crer les plans d'architectes. Ce format represente les informations gographiques en images vectorielles. Les donnees sont separes sur plusieurs calques.

Les plans ne sont pas encore gorefencres, c'est-a-dire qu'ils se trouvent dans un systeme local de coordonnees et qu'il faut les traiter afin de les placer dans un systeme de coordonnees exploitable par l'application.

4.1.3 Gorefencrement des plans dans QGis

Avant meme d'ouvrir le logiciel QGis, il faut choisir la projection (systeme de coordonnees) dans laquelle les plans seront gorefencres. Pour ce projet, la norme "ESPG : 3857 - WGS 84 / Pseudo-Mercator" a te choisie, car c'est celle utilisee par Open Street Map.

En premier lieu, il a fallu selectionner les calques contenant les informations utiles au projet, puis importer ceux-ci grace  l'outil integre dans QGis "Import Layers from DWG/DXF".

CHAPITRE 4. RÉALISATION DE LA SOLUTION



Figure 4.1 – Résultat final du géoréférencement d'un plan

En second lieu, il a fallu effectuer le géoréférencement. Pour ce faire, le fond de carte Open Street Map a été intégré et servira de référence pour le placement des plans. Le plugin "Vector Bender" a été utilisé pour effectuer une transformation affine sur les plans afin de les déplacer vers la bonne référence. Finalement, les nouvelles données ont été enregistrées au format "ESRI shapefile", plus adapté au manipulations dans ce logiciel.

Le résultat final peut être observé à la Figure 4.1.

4.1.4 Crédit des informations géographiques



Figure 4.2 – Résultat final de la création d'informations de l'étage E

Une fois le géoréférencement fini, seul le tracé de chaque étage était visible. Il a fallu ensuite créer les données suivantes :

- Le contour des étages sous forme de polygones au lieu de lignes.
- Le contour des salles sous forme de polygones.
- La localisation des ressources sous forme de points.

La Figure 4.2 illustre la création du contour de l'étage E et de ses salles. La Figure 4.3 illustre la création de localisations des ressources.

4.1. CRÉATION DES DONNÉES GÉOGRAPHIQUES

Le travail de création des données géographiques pour le site de Cheseaux a été chronophage.

Lors de la création de ces données, il y a la possibilité de rajouter des métadonnées à chaque information géographique (données supplémentaires). Cependant, ce travail a été principalement effectué sans avoir dans son optique cette possibilité. Une rectification nécessiterait un temps considérable.

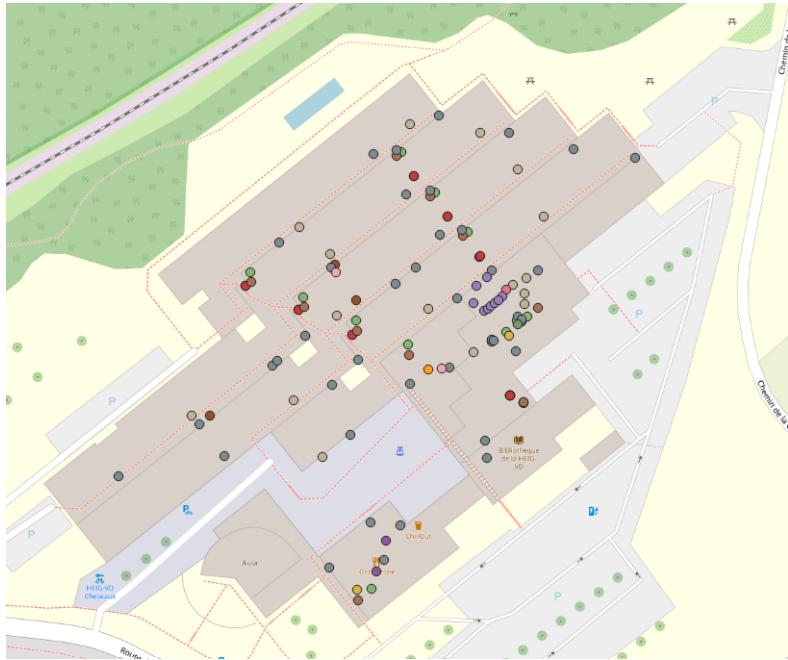


Figure 4.3 – Résultat final de la création d’informations des ressources

4.1.5 Problèmes rencontrés avec QGis

Le premier problème rencontré lors de l’utilisation du logiciel concernait les recherches de solutions pour le géoréférencement, certains outils ne faisant pas de transformation affine correcte ou émettant des erreurs.

Le second concernait le plugin "Vector Bender", nécessitant alors que tous les calques utilisés pour la transformation affine utilisent la même projection.

4.1.6 Exportation et arborescence de fichiers

Afin de pouvoir traiter les données et les placer dans la base de données, il fallait exporter celles-ci depuis le logiciel dans un format facilement lisible par un programme informatique. Le choix s'est porté sur le format GeoJson.

Celles-ci ont été exportées dans l’arborescence de fichiers suivante.

Le dossier racine contient des dossiers portant le nom des différents bâtiments. Ces derniers contiennent des dossiers portant le nom de chaque étage ainsi qu'un dossier contenant les fichiers GeoJson des ressources liées au bâtiment. Les dossiers des étages contiennent les fichiers GeoJson des informations géographiques relatifs à l'étage, ainsi qu'un dossier portant le nom rooms. Ce dernier contient les fichiers GeoJson relatifs à chaque salle.

Cette arborescence sera utile pour la création des scripts SQL.

4.1.7 Cration des scripts SQL

SQL est un langage permettant la gestion de bases de donnees relationnelles comme PostgreSQL. Celui-ci permet de crer la base de donnees ainsi que les tables, de recuperer les donnees, d'en inserer de nouvelles, d'en modifier ou d'en supprimer.

Pour ce projet, un programme a ete cre afin de lire les fichiers dans l'arborescence decrite dans la sous-section precedente et crer trois fichiers SQL : un pour la creation de la base de donnees ; un pour la creation des tables ; un pour l'insertion des donnees geographiques.

Ils seront utilises afin de mettre en place la base de donnees lors du deploiement de l'application.

4.2 Backend / Serveur-api

Le serveur-api s'occupe de recuperer les donnees depuis la base de donnees, et de les envoyer a l'application web.

4.2.1 Organisation du code source

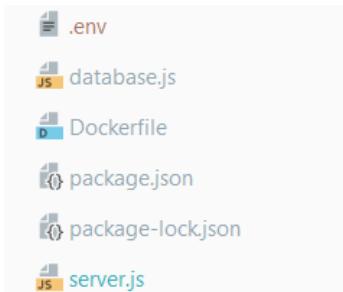


Figure 4.4 – Arborescence de fichiers du serveur-api

La Figure 4.4 contient deux fichiers importants : server.js qui implemente le serveur Express et les differentes routes, et database.js qui contient le code de connection a la base de donnees et les requetes SQL pour les acheminer.

4.2.2 Communications

Le diagramme de la Figure 4.5 presente les communications entre le client web, le serveur-api, et la base de donnees. Le client web va d'abord faire une requete HTTP au serveur-api. Ce dernier va envoyer une requete SQL a la base de donnees afin de recuperer les informations utiles.

Lorsque qu'il reoit les informations, il va verifier si celles-ci sont existantes. Le cas echant, il va les traiter puis les envoyer en format JSON au client web, avec le code de statut "200 OK", propre au protocole HTTP. Dans le cas contraire, il va retourner un message d'erreur avec le code de statut "400 Bad request".

4.2.3 Routing

L'application web a besoin d'acheminer differentes ressources. Pour ce faire, elle accede au serveur-api en utilisant differentes URLs liees aux donnees desirees. Ces URLs sont appelees routes.

4.2. BACKEND / SERVEUR-API

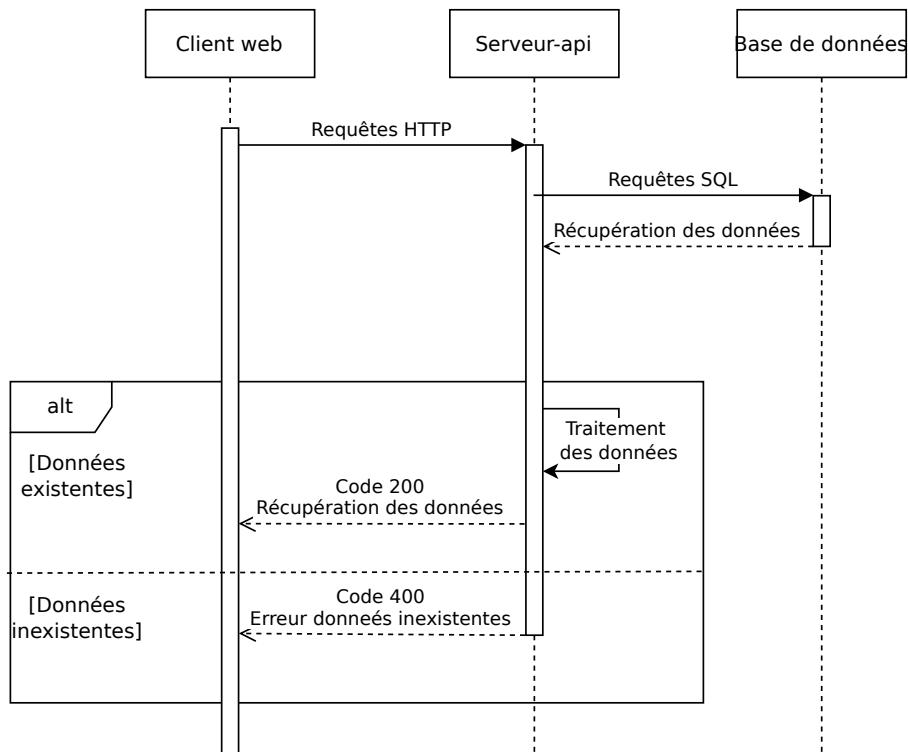


Figure 4.5 – Diagramme de séquence représentant les communications pour le serveur-api

Les routes seront présentées avec la deuxième partie de l'URL. Par exemple, le site factice `www.site.web/element1/ :element2` sera présenté seulement avec `/element1/ :element2`.

De plus, si un des éléments est précédé par deux points, cela veut dire qu'il peut prendre n'importe quelle valeur. Par exemple, `/element1/ :element2` peut être écrit `/element1/356` ou `/element1/bonjour`. Cette valeur sera utilisée lors de la requête SQL et récupérera les informations correspondant aux entrées de la base de données.

Ci-dessous la liste de chaque route et des données récupérées :

- `'/api/buildings'` récupère les informations sur tous les bâtiments.
- `'/api/buildings/ :buildingId/floors'` récupère les informations sur les étages d'un bâtiment.
- `'/api/buildings/ :buildingId/resources'` récupère les informations sur les ressources liés à un bâtiment mais pas à un étage.
- `'/api/floors/ :floorId/features'` récupère les informations liées à un étage.
- `'/api/rooms/ :roomId/resources'` récupère les ressources liées à une salle.
- `'/api/rooms/search/ :search'` récupère les noms des salles selon une recherche.
- `'/api/rooms/ :name'` récupère les informations d'une salle selon leur nom.

4.2.4 ORM

Ce projet n'utilise pas d'ORM, une technique de programmation qui facilite la récupération de données prêtes à l'emploi, pour les raisons suivantes :

- Les données sont récupérées au format JSON, utilisable par le langage de programmation JavaScript sans trop de transformation.
- Certaines requêtes sont spécifiques à l'extension PostGis et auraient demandé d'étendre les fonctionnalités de l'ORM.

CHAPITRE 4. RÉALISATION DE LA SOLUTION

- Le serveur-api ne fait que peu de traitement de données avant de les renvoyer au client web.

4.3 Frontend

Le frontend est un projet qui sera transformé en fichier HTML, CSS et JavaScript, avant d'être délivré par le serveur.



Figure 4.6 – Affichage global de l'application

4.3.1 Organisation du code source

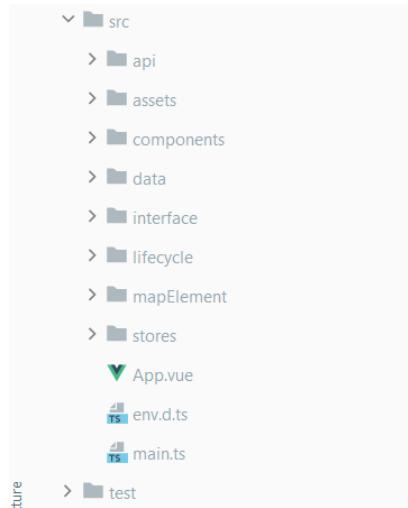


Figure 4.7 – Organisation du code source

Le code de l'application (voir Figure 4.7) se trouve dans le dossier nommé "src" (abréviation de source), et les tests dans le dossier test. A la racine du dossier "src" se trouve le fichier "main.ts", porte d'entrée de l'application, ainsi que le fichier "app.vue", component Vue principal qui englobe tous les autres components.

Le reste du dossier est composé des dossiers suivants :

- api : contient le code permettant de communiquer avec le serveur-api.
- assets : contient les images de l'application.

4.3. FRONTEND

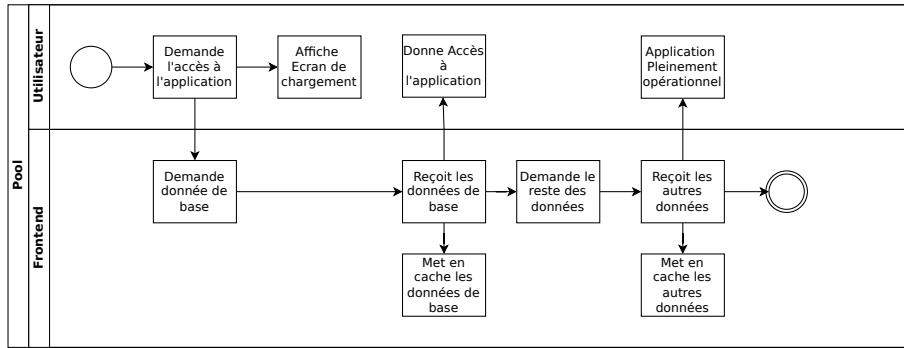


Figure 4.8 – Schéma de la stratégie de récupération des données

- component : contient les composants Vue, organisés dans des sous-dossiers par type de composant.
- interface : contient les interfaces TypeScript qui définissent des types d'objet.
- lifecycle : contient le code gérant la stratégie de récolte des données.
- mapElement : contient les éléments concernant la librairie Openlayers.
- stores : contient les fichiers permettant de communiquer des informations entre plusieurs composant Vue.

4.3.2 Stratégie de récupération des données

La récolte des données nécessaires au bon fonctionnement de l'application se fait en deux temps(voir Figure 4.8). Ceci permet à l'utilisateur d'accéder plus rapidement à l'application sans devoir attendre que celle-ci soit complètement prête. Dans un premier temps, l'application va récupérer les données nécessaires via le serveur-api afin d'afficher le fond de carte, l'étage E de Cheseaux et les ressources associées à cet étage. L'utilisateur pourra alors accéder aux plans sans trop d'attente.

Dans un deuxième temps, le serveur-api va récupérer les informations géographiques de tous les bâtiments contenues dans la base de données, ainsi que leur étage.

Toutes les données récupérées lors de ces deux étapes sont mises en cache (stockées temporairement sur la machine de l'utilisateur), afin d'éviter de multiplier les requêtes, et permettre ainsi une navigation fluide sur la carte sans temps de latence.

Lors de l'utilisation de l'outil de recherche, des requêtes sont encore effectuées pour récupérer les données de salle.

4.3.3 Openlayers calques

Openlayers permet de séparer les données géographiques sur plusieurs calques (layers) superposés. Afin d'afficher les plans, les données géographiques ont été séparées sur les calques suivants :

- osmLayer : calque affichant le fond de carte fourni par Open Street Map.
- backgroundLayer : calque affichant les contours du bâtiment.
- polygonLayer : calque affichant le fond de l'étage sélectionné.
- lineLayer : calque affichant les lignes provenant du plan de l'étage selectionné.
- labelsLayer : calque affichant les noms des salles.
- resourceLayer : calque affichant les icônes des ressources.

Quand un utilisateur veut changer d'étage ou de bâtiment, les sources de données des calques sont remplacées par d'autres sources.

CHAPITRE 4. RÉALISATION DE LA SOLUTION

4.3.4 Fonctionnalités

Ecran de chargement

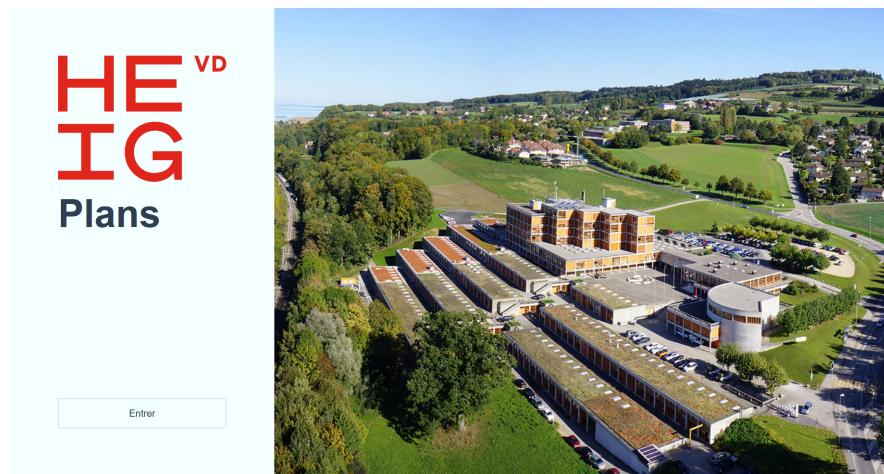


Figure 4.9 – Ecran de chargement

L'écran de chargement (voir Figure 4.9) permet de faire patienter l'utilisateur jusqu'au chargement des informations essentielles à l'affichage des plans (1ère étape de la récupération des données). Il permet aussi de donner un aperçu du bâtiment de Cheseaux et ainsi offrir une meilleure compréhension des plans par la suite.

L'accès aux plans se fait au moyen d'un bouton.

Outil de changement de bâtiment



Figure 4.10 – Outil de changement de bâtiments

L'outil de changement de bâtiment (voir Figure 4.10) permet de naviguer d'un bâtiment à l'autre. Le bâtiment actuellement sélectionné est écrit en blanc sur fond rouge. Les autres bâtiments sont des boutons. Tous les textes sont des abréviations des noms des bâtiments, et lorsque l'utilisateur passe le curseur au dessus de l'outil, le nom entier s'affiche à la place des abréviations.

Lors d'un clic sur un des ces boutons, le plan change d'affichage et se centre sur le bâtiment nouvellement sélectionné. Les sources des calques sont modifiées en conséquence. L'étage qui est sélectionné est le rez-de-chaussée du bâtiment, mentionné dans la base de données. L'outil de changement d'étage adapte aussi la liste des étages. Cependant, ceci ne se vérifie pas dans la version finale car les données des étages de St-Roch ne sont pas implémentées, le comportement normal de l'application lors de données manquantes étant de ne pas modifier la liste.

Lors de la conception du design, cet outil n'avait pas été planifié.

4.3. FRONTEND



Figure 4.11 – Outil de changement d'étages

Outil de changement d'étages

L'outil de changement d'étages (voir Figure 4.11) permet de naviguer à travers les étages d'un bâtiment. Il présente l'étage sélectionné, ainsi que les étages adjacents.

Le design de l'outil correspond à ce qui était prévu. Il a cependant été associé à l'outil de changement de bâtiment dans une barre d'outil, et a été placé à gauche.

Outil de filtrage des ressources

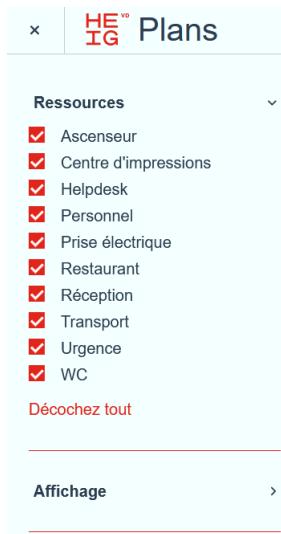


Figure 4.12 – Outil de filtrage des ressources

L'outil de filtrage des ressources (voir Figure 4.12) se trouve dans le menu accessible à partir du bouton à gauche du header. Il permet de sélectionner les types de ressources à afficher. Si ces dernières sont sélectionnées, les icônes des ressources correspondantes s'affichent sur le plan. Dans le cas contraire, les icônes sont masquées.

Le design de l'outil est plus abouti que la conception initiale : les cases à cocher se trouvent à gauche, et il se trouve dans un menu déroulant.

Outil de changement d'affichage

L'outil de changement d'affichage (voir Figure 4.13) permet de modifier la présentation des plans. Actuellement, deux affichages différents sont disponibles : celui par défaut, et un affichage qui colorise les salles en fonction de leur type.

CHAPITRE 4. RÉALISATION DE LA SOLUTION



Figure 4.13 – Outil de changement d'affichage et effet sur les plans

Lors de la conception du design, cet outil n'avait pas été planifié.

Interaction avec les plans

L'utilisateur peut interagir avec les plans en cliquant sur des salles ou des icônes de ressources. Si c'est le cas, celles-ci se mettent en mode sélectionné et l'application affiche la fenêtre d'informations.

Cela est possible car Openlayers fournit des événements lors de l'interaction avec les plans.

Grâce à Openlayers, il est possible de sélectionner plusieurs salles ou ressources en maintenant la touche shift. Ceci est géré par la fenêtre d'informations en affichant les infos de chaque salle ou ressource.

Outil de recherche



Figure 4.14 – Outil de recherche avec menu des suggestions

L'outil de recherche (voir Figure 4.14) permet de rechercher une salle par son numéro ou par son nom. Il permet aussi de trouver la salle où se trouve un collaborateur de l'école en écrivant son nom.

4.3. FRONTEND

A partir de deux caractères écrits, l'outil propose des suggestions de salles et/ou de collaborateurs. L'utilisateur peut cliquer sur une des suggestions. S'il le fait, les plans change pour afficher la salle concernée en rouge et les informations de celle-ci dans la fenêtre d'informations.

Si l'outil n'a aucune suggestion à proposer, il affiche une erreur. Si l'utilisateur clique sur un collaborateur lié à une salle non implémenté dans la base de données, un message d'alerte le précise à l'utilisateur.

Lorsque l'utilisateur clique en dehors de la barre de recherche, le menu de suggestions disparaît, et reapparaît lorsqu'il reclique à l'intérieur.

Afin de limiter les requêtes au serveur-api, les données des salles sont mises en cache à partir de deux caractères. Ce cache est réinitialisé quand l'utilisateur clique sur une suggestion, ou qu'il efface sa saisie en dessous de deux caractères.

Le design de l'outil n'est pas très différent de la conception initiale : l'icone de loupe n'apparaît plus dans la barre de recherche. Les suggestions de salle proposent en plus du numéro le nom de la salle.

Fenêtre d'informations



Figure 4.15 – Fenêtre d'informations

La fenêtre d'informations (voir Figure 4.15) peut apparaître soit après une recherche, soit lors d'une interaction avec les plans. Celle-ci fait apparaître toutes les informations disponibles au sujet de la salle ou de la ressource. Dans le cas d'une salle, il peut afficher dans une liste déroulante le personnel et/ou les ressources associées à celle-ci.

Pour quitter la fenêtre d'informations, il faut cliquer directement sur les plans.

4.3.5 Responsivité

Afin de garantir l'accessibilité des fonctionnalités lors de l'utilisation de l'application, l'affichage s'adapte en fonction de la taille de l'écran.

Si la largeur de l'écran est trop petite, la barre de recherche devient un bouton avec une icône de loupe. Si l'utilisateur clique dessus, la barre de recherche s'affiche à la place du logo et du titre de l'application. Celui-ci redévient normal lorsque l'utilisateur reclique sur le bouton ou finalise sa recherche.

La fenêtre d'informations s'adapte aussi si la largeur est trop petite. Elle s'affiche au bas de l'écran au lieu de la droite.

CHAPITRE 4. RÉALISATION DE LA SOLUTION

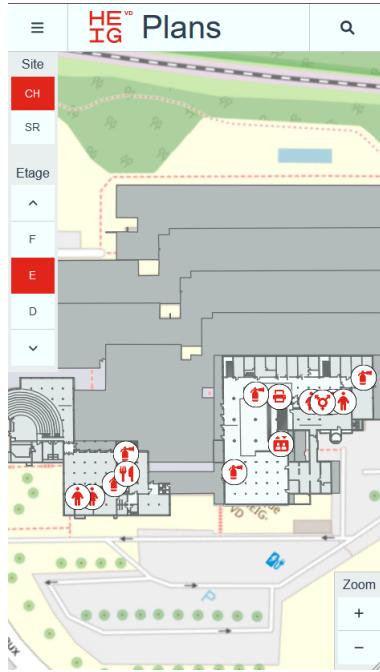


Figure 4.16 – Affichage sur smartphone en mode portrait

Si la hauteur de l'écran est trop petite, le menu comportant les outils de changement de bâtiment et d'étages se place sur deux colonnes.

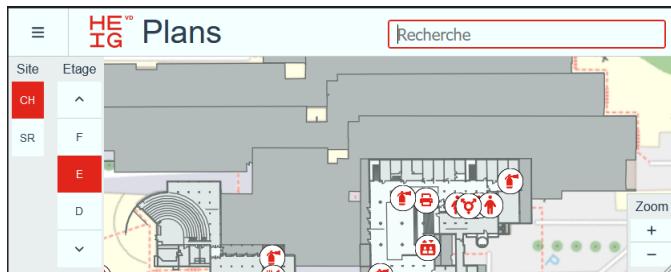


Figure 4.17 – Affichage sur smartphone en mode paysage

4.4 Déploiement

4.4.1 Docker compose

Docker compose est une méthode proposée par Docker qui permet de démarrer tous les containers de l'infrastructure en une seule commande à partir d'un fichier de configuration. Ce fichier permet aussi de paramétriser chaque container et les communications entre eux.

4.4.2 Déploiement sur la machine virtuelle

Afin d'accueillir les différents containers, il a fallu préparer la machine virtuelle en téléchargeant le logiciel Docker. Ensuite il a fallu transférer le fichier de configuration utilisé par Docker compose, ainsi que les scripts SQL. Il suffit ensuite de démarrer la construction des images.

Docker va télécharger les quatre images nécessaires à la création des containers, soit

4.4. DÉPLOIEMENT

Traefik, PostGis et les images du serveur et du serveur-api préalablement créées et hébergées sur GitLab container registry. Par la suite, il créera les containers, les paramétrera et créera les liens pour la communication entre eux.

Une fois fini, le site est mis à disposition de tous.

Paramétrage PostGis

Lors de la construction du container PostGis, les paramètres d'accès à la base de données sont définis. La base de données va être construite, avec ses tables, et les données remplissant ces tables en utilisant les fichiers SQL fournis.

Paramétrage Traefik

Lors de la construction du container Traefik, on va définir les points d'entrées par lesquels les clients accèdent aux serveurs, ainsi que les règles permettant de rediriger vers le bon serveur.

Paramétrage serveur-api

Lors de la construction du container serveur-api, on précise le nom du container PostGis, ainsi que les paramètres pour y accéder.

Chapitre 5

Conclusion

5.1 Critique de la solution

5.1.1 Comparaison avec le cahier des charges

Cette section compare le travail planifié par le cahier des charges avec le travail accompli. Cette comparaison est résumée à la Table 5.1.

| Nom des tâches | Est nécessaire | Etat d'avancement |
|------------------------------------|----------------|-------------------|
| Visualisation plans Cheseaux | Oui | Complet |
| Affichage noms des salles | Oui | Complet |
| Affichage qualificatifs des salles | Oui | Complet |
| Outil de changement d'étages | Oui | Complet |
| Surface des salles | Oui | Complet |
| Responsivité | Oui | Complet |
| Hébergé sur une machine virtuelle | Oui | Complet |
| Comporte une base de données | Oui | Complet |
| Localisation des ressources | Oui | Complet |
| Outil de filtrage | Non | Complet |
| Outil de recherche | Non | Partiel |

Table 5.1 – Etat d'avancement des fonctionnalités du cahier des charges

Les différentes fonctionnalités nécessaires ont été implémentées au niveau du code et complétées dans ce projet. Seul l'outil de recherche a une implémentation partielle car la recherche par des critères, autre que par le nom, est inexistante. Il n'est pas exclu d'améliorer chaque fonctionnalité lors de futures itérations du projet.

D'autres fonctionnalités non prévues initialement par le cahier des charges ont été implémentées. Elles sont présentées à la Table 5.2

| Nom des tâches | Etat d'avancement |
|----------------------------------|-------------------|
| Page de chargement | Complet |
| Outil de changement de bâtiments | Complet |
| Outil de changement d'affichage | Partiel |
| Interaction avec les plans | Complet |

Table 5.2 – Etat d'avancement des fonctionnalités supplémentaires

CHAPITRE 5. CONCLUSION

L'outil de changement d'affichage n'est que partiellement implémenté, car il ne propose que deux types. D'autres types d'affichages pourront être développés, comme la possibilité de préciser la disponibilité des salles de cours à un horaire fixe.

5.1.2 *Traitement et création des données géographiques*

Le traitement des données est suffisant pour une démonstration des capacités de l'application web, mais pas assez précis pour une version définitive : le géoréférencement des plans des étages est différent et souffre d'un léger décalage entre chaque étage. Il faudrait utiliser un meilleur outil que l'extension Vector Bender.

Les autres données géographiques sont dépendantes de ce premier géoréférencement et devront être refaites. De plus, de nombreuses données n'ont pas été créées, comme celle concernant les deux autres sites de l'école.

Ces erreurs sont dues à un manque d'expérience avec les logiciels et les techniques dans ce domaine.

5.1.3 *Design de l'application*

Le design de l'application est satisfaisant car le projet apparaît achevé. Des bonnes pratiques concernant l'expérience utilisateur ont été implémentées afin d'offrir une prise en main aisée.

Cependant, ces points pourraient être améliorés à l'avenir.

5.1.4 *Backend*

Le backend est simple et suffisant pour acheminer les données. C'est pourquoi il a été développé avec des technologies de base. S'il fallait le complexifier, il faudrait alors envisager d'autres technologies comme le framework Nest JS.

5.1.5 *Frontend*

Le frontend est la partie la plus aboutie du projet. Les fonctionnalités nécessaires et optionnelles ont été implémentées. Le code source a été séparé en plusieurs fichiers afin d'offrir une compréhension du projet. Les différentes fonctionnalités ont été testées.

Cependant, il contient sûrement des erreurs dues à un manque d'expérience et de connaissances dans les bonnes pratiques à utiliser.

Les tests end-to-end servant à éprouver l'application dans sa globalité n'ont pas été implémentés. Il faudrait aussi soumettre l'application à un plus grand nombre d'utilisateurs afin d'observer leur utilisation et détecter d'autres bugs.

5.2 *Critique de la méthode de travail*

5.2.1 *Critique du temps accrédité au projet*

Ce projet atteint 424.5 heures de travail au lieu des 450 heures requises au travail de Bachelor. Il accuse donc un manque de 25.5 heures. Cela est dû à plusieurs facteurs :

- Démarrage lent du projet lors de la récupération d'informations (8 heures).
- La semaine spéciale prévue par l'école (Crunch) n'a pas pu être mise à contribution (12 heures).
- Quelques jours ont été perdus à cause de maladie (8 heures).

5.2. CRITIQUE DE LA MÉTHODE DE TRAVAIL

- D'autres petits projets ont pris le pas sur celui-ci en fin de semestre (12 heures).
- 6.5 jours d'indisponibilité à la fin du projet (52 heures).

Une semaine d'extension a été accordée à l'accomplissement du projet, permettant la récupération de 40 heures.

Au total, environ 52 heures ont été perdues. 26.5 heures ont pu être rattrapées.

Certains éléments, comme la maladie, pourraient être comptabilisés comme imprévus et feraient partie des 90 heures planifiées.

Malgré les retards, toutes les fonctionnalités prévues, ainsi que quelque autres ont été implémentées. Le temps manquant aurait alors été consacré à l'optimisation de ces dernières, à l'implémentation de tests end-to-end et à la correction de bugs.

5.2.2 Comparaison entre planification et réalisation

La Table 5.3 compare la planification et le temps pris lors de la réalisation.

| Etapes | Planifié | Réalisé |
|-----------------------|----------|---------|
| Analyse et conception | 49 | 48 |
| Traitement données | 95 | 110.5 |
| Développement | 137 | 162 |
| Déploiement | 30 | 16 |
| Documentation | 75 | 79.5 |
| Total | 386 | 416 |

Table 5.3 – Tableau de comparaison entre planification et réalisation

A noter que le total de la planification ne prend pas en compte les 90 heures d'imprévus, ce qui porterait le total à 476 heures. La différence entre le total de la réalisation et celui mentionné dans la sous-section précédente s'explique par les réunions administratives ayant pris un total de 8.5 heures.

Les étapes de traitement de données, de développement et de documentation ont pris plus de temps que planifiées. L'étape de déploiement a été, quant à elle, plus rapide.

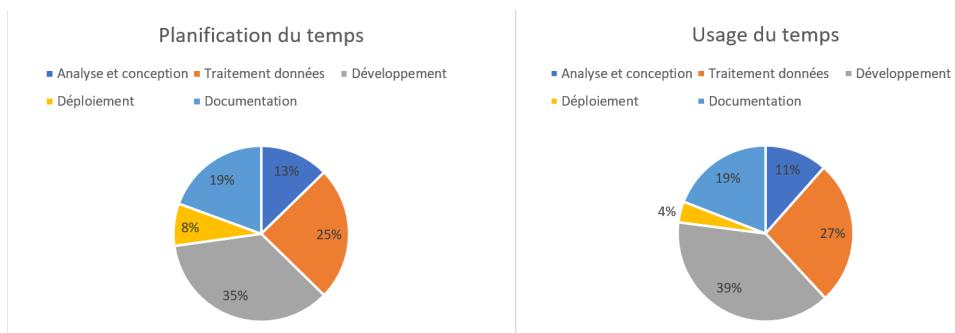


Figure 5.1 – Répartition du temps

A la Figure 5.1, on observe que le temps de travail a été plus concentré sur le traitement des données et le développement et moins sur le déploiement.

5.3 *Implémentations futures*

Ce projet peut être étendu de plusieurs manières. D'abord en améliorant les différentes fonctionnalités présentes dans cette solution, ensuite en implémentant les fonctionnalités suivantes :

- Recherche par critère.
- Affichage de la disponibilité des salles de cours.
- Outil d'exportation et d'impression des plans.
- Outil du plus court itinéraire vers une ressource ou une salle.
- Outil de dessin vectoriel sur les plans.
- Outil de rajout de ressources localisées pour les administrateurs de l'application.
- Localisation de l'utilisateur sur le plan.
- Technologie pour surveiller le trafic de l'application.
- Optimiser la détection par les moteurs de recherches (SEO).

Cette liste n'est pas exhaustive.

5.4 *Conclusion personnelle*

Le travail accompli est, à mon avis, bon. Il est perfectible sur de nombreux points. Pour autant, il s'agit du travail d'une seule personne, ne connaissant initialement que peu de choses dans le domaine des webSIG.

Un tel projet, pour être finalisé, devrait soit être réalisé par une équipe de différents corps de métier, et être planifié sur une période plus longue, permettant un apprentissage plus approfondi des différents domaines.

Kylian Bourcoud



Annexes

Annexe A

Planning

| Tâche | Description | Effort | Semaine Jours | | | | | | | | | | | | | | | | | | | | | | Effort | Effort | Différence | | | |
|----------|--|--------|---------------|-----|-----|-----|-----|------|------|-----|------|-----|------|------|------|------|-----|-----|-----|-----|------|------|-----|-----|--------|----------|------------|-----|------|-------|
| | | | Prévu | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | Consommé | Restant | | | |
| 10 | Elaboration et vérification du projet | 41 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10.10 | <i>Recherche informative</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10.10.10 | Recherche de systèmes existants | 8 | 6 | 2 | 2 | 2 | | | | | | | | | | | | | | | | | | | | | | 9 | 0 | -1.0 |
| 10.10.20 | Recherche technologique | 8 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | 7 | 0 | 1.0 |
| 10.20 | <i>Définition du projet</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10.20.10 | Cahier des charges | 20 | 2 | 5 | 4 | 7 | 3 | 4 | | | | | | | | | | | | | | | | | | | | 21 | 0 | -1.0 |
| 10.20.20 | Planning | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | 4 | 0 | 1.0 |
| 20 | v0.1 - Proof of concept | 76 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20.10 | <i>Traitement d'un plan</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20.10.10 | Géoréférencement du plan | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | 23 | -3 | 0.0 |
| 20.10.20 | Conversion format GeoJson + shapefile | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | 2 | 0 | 2.0 |
| 20.20 | <i>Design du site</i> | 8 | | | | | | | | | | | | | | | | | | | | | | | | | | 7 | 0 | 1.0 |
| 20.30 | <i>Pipeline CI/CD</i> | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 4.0 |
| 20.40 | <i>Backend</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20.40.10 | Déploiement serveur | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | 3 | 0 | 1.0 |
| 20.40.20 | Intégration des données géographiques du plan à la BDD | 12 | | | | | | | | | | | | | | | | | | | | | | | | | | 16 | 0 | -4.0 |
| 20.50 | <i>Frontend</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20.50.10 | Création projet basique | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 3.0 |
| 20.50.20 | Affichage du plan | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | 12 | 0 | -2.0 |
| 20.60 | <i>Communication frontend/backend</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20.60.10 | Envoi des données géographiques | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 9.0 |
| 30 | V0.2 - Affichage des plans | 154 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30.10 | <i>Pipeline CI/CD</i> | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | 20 | 0 | -10.0 |
| 30.20 | Création test | | | | | | | | | | | | | | | | | | | | | | | | | | | 12 | 0 | -12.0 |
| 30.20 | <i>Traitement des plans</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30.20.10 | Géoréférencement des plans | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | 2 | 0 | 18.0 |
| 30.20.20 | Création des données de salles | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | 40 | 0 | -10.0 |
| 30.20.30 | Conversion format compatible | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | 3.0 |
| 30.20.40 | Ajout des ressources | | | | | | | | | | | | | | | | | | | | | | | | | | | 6.5 | 0 | -6.5 |
| 30.30 | <i>Backend</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30.30.10 | Intégration des données géographiques dans la BDD | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | 20 | 0 | -15.0 |
| 30.30.20 | Mise en place complète du serveur API | 25 | | | | | | | | | | | | | | | | | | | | | | | | | | 6 | 0 | 19.0 |
| 30.40 | <i>Frontend</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30.40.10 | Affichage des plans | 16 | | | | | | | | | | | | | | | | | | | | | | | | | | 30 | 0 | -14.0 |
| 30.40.20 | Outil changement d'étage | 12 | | | | | | | | | | | | | | | | | | | | | | | | | | 10 | 0 | 2.0 |
| 30.40.30 | Outil changement de bâtiments | 6 | | | | | | | | | | | | | | | | | | | | | | | | | | 7 | 0 | -1.0 |
| 30.40.40 | Adaptation selon zoom | 16 | | | | | | | | | | | | | | | | | | | | | | | | | | 4 | 0 | 12.0 |
| 30.40.50 | Outil de filtres | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | 2 | 0 | -15.0 |
| 30.40.60 | Fenêtre d'information | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | 10 | 0 | -10.0 |
| 30.40.70 | Refactoring | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | 9 | 0 | -9.0 |
| 30.40.80 | Outil de recherche | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | 16 | 0 | -16.0 |
| 30.40 | <i>Communication frontend/backend</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30.40.10 | Adaptation des communications | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 20.0 |
| 30.50 | <i>Déploiement</i> | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | 16 | 0 | 14.0 |
| 40 | Préparation des livrables | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 40.10 | Documentation | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | 6.5 | 0 | 13.5 |
| 40.20 | Rapport intermédiaire | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | 16 | 0 | -6.0 |
| 40.40 | Rapport final | 40 | | | | | | | | | | | | | | | | | | | | | | | | | | 57 | 0 | -17.0 |
| 40.60 | Présentation PowerPoint | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 5.0 |
| | Réunion | 0.5 | 0.5 | 0.5 | | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 8.5 | 0 | -8.5 | |
| 50 | Imprévis (20 %) | 90 | | | | | | | | | | | | | | | | | | | | | | | | | | 6 | 0 | 84.0 |
| | TOTAL | 476 | 12.5 | 7.5 | 4.5 | 0 | 18 | 12.5 | 13.5 | 7.5 | 10.5 | 9.5 | 13.5 | 17.5 | 19.5 | 12.5 | 16 | 0.5 | 48 | 44 | 43.5 | 40.5 | 30 | 43 | 424.5 | -3 | 54.5 | | | |

Annexe B

Cahier des charges



Route de Cheseaux, 1
CH-1401 Yverdon-les-Bains
+41 24 557 73 77

Plans de la HEIG-VD interactifs

Travail de Bachelor 2022 | Cahier des charges

Kylian Bourcoud

Avant-propos

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'École.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

Table des matières

| | | |
|-------|---|----|
| 1 | Introduction | 4 |
| 1.1 | Domaine d'application | 4 |
| 1.2 | Référence normative..... | 4 |
| 1.3 | Termes et définitions | 4 |
| 1.4 | Contexte | 5 |
| 1.5 | Description du problème | 5 |
| 2 | Analyse fonctionnelle..... | 6 |
| 2.1 | Cas d'utilisation | 6 |
| 2.2 | Analyse du besoin | 7 |
| 3 | Fonctions | 7 |
| 4 | Idéation et analyse de la solution | 8 |
| 4.1 | Systèmes existants d'interface interactive | 8 |
| 4.1.1 | Plan EPFL | 8 |
| 4.1.2 | EPFL Géoportail | 9 |
| 4.1.3 | MIT map | 10 |
| 4.1.4 | SITN – Géoportail du système d'information du territoire neuchâtelois | 11 |
| 4.1.5 | Stanford campus map | 12 |
| 4.1.6 | Aéroport de Zurich | 13 |
| 4.1.7 | Conclusion de l'analyse de système existant | 13 |
| 4.2 | Architecture webGIS | 14 |
| 4.3 | Technologies | 14 |
| 4.3.1 | Openlayers | 14 |
| 4.3.2 | Leaflet..... | 14 |
| 4.3.3 | Google Maps API | 14 |
| 4.3.4 | GeoMapFish | 14 |
| 4.3.5 | Serveur cartographique..... | 15 |
| 4.3.6 | PostGreSQL et PostGIS | 15 |
| 4.4 | Données existantes à disposition | 15 |
| 4.4.1 | Plans | 15 |
| 4.4.2 | Serveur LDAP | 15 |
| 4.5 | Solution | 15 |
| 5 | Exigences..... | 16 |

| | | |
|---|---------------------|----|
| 6 | Bibliographie | 17 |
|---|---------------------|----|

1 Introduction

1.1 Domaine d'application

Ce document analyse les besoins et spécifient les exigences pour le travail de bachelor “Plans de la HEIG-VD interactifs”.

1.2 Référence normative

Les documents suivants cités dans le texte constituent, pour tout ou partie de leur contenu, des exigences du présent document.

| Norme | Description |
|-----------------------------|---|
| OMG UML 2.5.1 | <i>Unified Modeling Language</i> pour l'ingénierie logicielle. |
| ISO/IEC Dir 2 : 2016 | Principes et règles de structure et de rédaction des documents ISO et IEC |

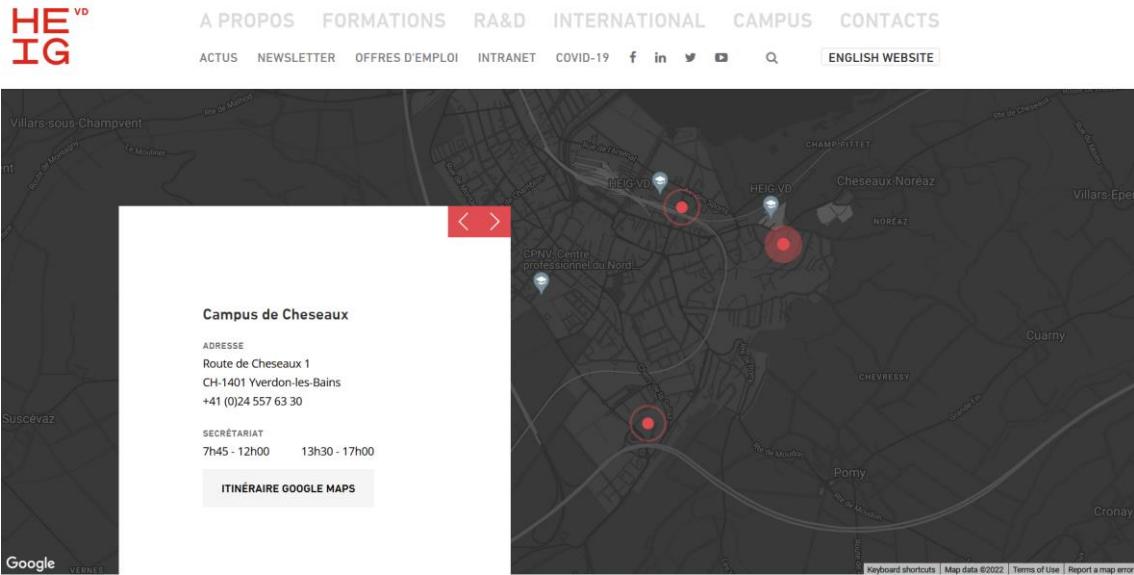
1.3 Termes et définitions

Cette section de termes et définitions précise les terminologies techniques et autres acronymes propres au projet.

| Terme | Définition |
|-----------|---|
| HEIG-VD | Haute École d'ingénierie et de gestion du canton de Vaud |
| Ressource | Terme générale pouvant symboliser plusieurs choses comme une salle, une personne ou encore de l'équipement (armoire, machine-outil) |
| webGIS | Système d'informations géographique pour le web |

1.4 Contexte

Les cartes qu'elle soit schématique ou précise ont été un outil très utile pour l'orientation des personnes au sein d'une région ou d'un bâtiment. Avec l'arrivée du web, l'usage des plateformes de cartes comme google maps ont peu à peu remplacer les cartes papier, amenant des fonctionnalités plus interactives.



La HEIG-VD fournit un plan (voir image ci-dessus) ainsi que des informations sur les différents moyens d'accès afin d'aider ses utilisateurs à s'orienter vers ses trois sites. Cependant elle ne fournit, ni sur son site, ni dans le guide de l'étudiant, une interface permettant de s'orienter sur les sites eux-mêmes.

1.5 Description du problème

M.Chevallier souhaite mettre à disposition aux utilisateurs une interface interactive pour visualiser les salles et les plans de la HEIG-VD sur ses trois sites.

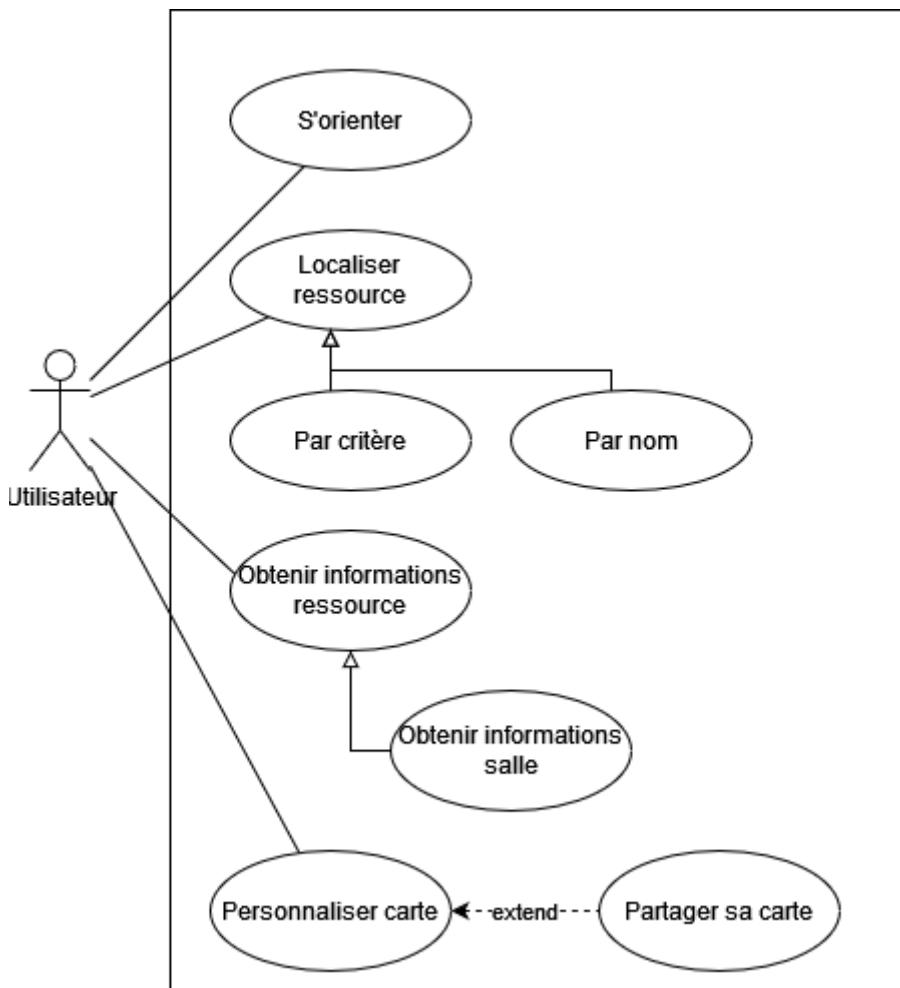
2 Analyse fonctionnelle

2.1 Cas d'utilisation

Le cas principal d'utilisation de l'interface est de pouvoir s'orienter à travers les différents sites de la HEIG-VD.

D'autres cas d'utilisation peuvent aussi survenir, comme localiser une ressource, obtenir des informations sur une ressource ou encore créer une carte personnalisée.

Le schéma ci-dessous en notation UML résume les différents cas d'utilisation



2.2 Analyse du besoin

Suivant les cas d'utilisations illustré ci-dessus, les besoins identifiés des utilisateurs sont exprimés dans le tableau 3

Tableau 1 Besoins des utilisateurs

| # | Besoin |
|----|--|
| N1 | S'orienter facilement à travers les sites de la HEIG-VD |
| N2 | Localiser une ressource à l'aide de son nom |
| N3 | Localiser une ressource à l'aide de critères |
| N4 | S'informer efficacement sur une ressource |
| N5 | Être capable de personnaliser une carte |
| N6 | Être capable de partager sa carte personnalisé |
| N7 | S'informer efficacement des noms des locaux, leurs surfaces, et leur types |
| N8 | Localiser facilement un collaborateur sur un plan des sites |

3 Fonctions

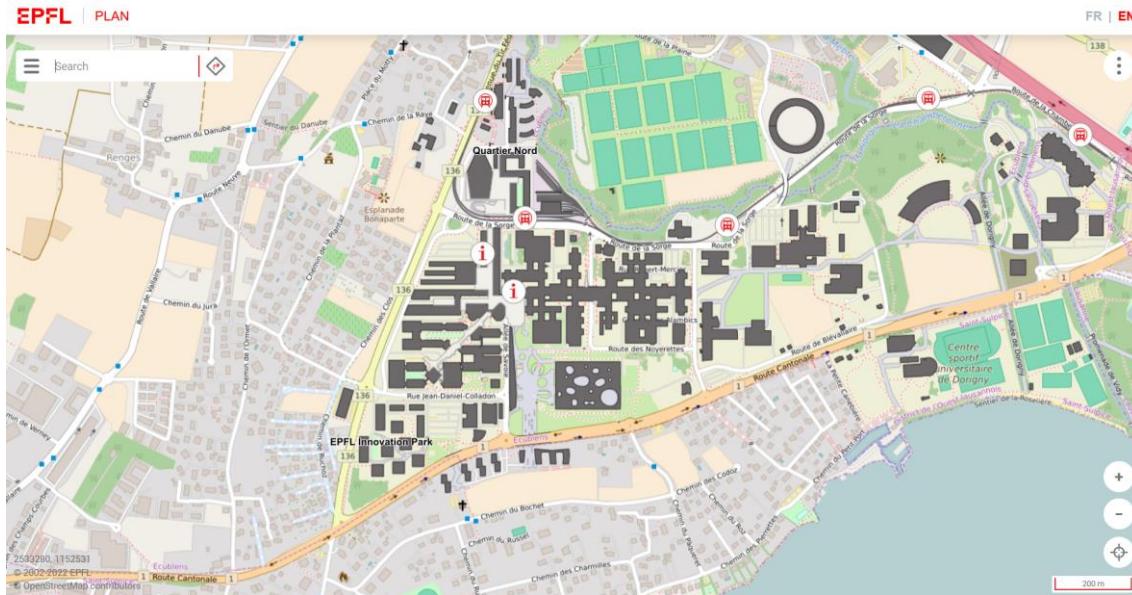
Tableau 5 Fonction du système à concevoir

| # | Fonctions système | Besoin |
|-----|--|----------------------|
| F1 | Afficher un plan afin d'aider l'orientation | N1.1 N1.7 |
| F2 | Utilisable facilement et de façon ergonomique | N1.1 |
| F3 | Fournir une orientation rapidement | N1.1 |
| F4 | Facilement accessible | N1.1 |
| F5 | Afficher les ressources désirées | N1.1 N1.3 N1.5 |
| F6 | Fournir un outil de tracé de plus cour itinéraire pour l'orientation | N1.1 |
| F7 | Offrir un outil de localisation de ressource par nom | N1.2 N1.8 |
| F8 | Fournir un outil de localisation de ressource par critères | N1.3 |
| F9 | Fournir des informations sur les ressources | N1.4 N1.7 |
| F10 | Fournir un outil de dessin sur carte | N1.5 |
| F11 | Fournir un outil d'exportation | N1.5 |
| F12 | Fournir un outil d'impression de carte | N1.6 |
| F13 | Fournir un outil de partage de carte | N1.6 |
| F14 | Fournir un outil de sauvegarde de carte | N1.6 |

4 Idéation et analyse de la solution

4.1 Systèmes existants d'interface interactive

4.1.1 Plan EPFL



Fonctionnalités

Les plans interactifs du campus de l'EPFL affichent les bâtiments du campus en s'adaptant selon le zoom et l'étage sélectionné. Suivant le zoom on peut visualiser le contour du site, puis les le contour des bâtiments et enfin les salles des bâtiments.



L'utilisateur a la possibilité de filtrer les points d'intérêts à afficher sur la carte à l'aide d'un menu sur la gauche du site.

Il y a aussi la possibilité de rechercher différentes ressources en fonction de leur nom comme les bâtiments, les salles, les personnes, les restaurants, les magasins, ou encore les espaces culturels.

D'autre outil sont fournis comme la recherche d'un plus court itinéraire entre deux ressources, un outil pour l'impression, changer l'affichage pour une vue aérienne.

Finalement un lien permet d'accéder au Géoportail de l'EPFL.

Technologies

La principale technologie utilisée pour le frontend est ngeo (combine Angular js et openlayers).

4.1.2 EPFL Géoportail

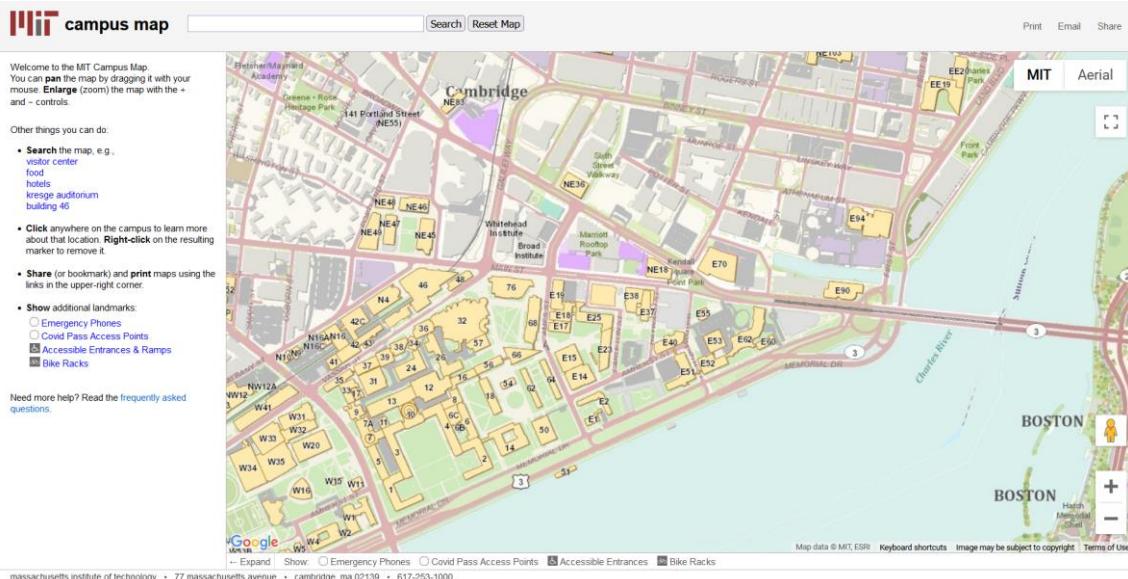
EPFL Géoportail



Le Géoportail de l'EPFL est très similaire au plan du campus, mais il offre la possibilité de dessiner des formes vectorielles sur la carte.

Il permet aussi d'afficher des ressources plus précises comme les réseaux wifi ou les prises électrique.

4.1.3 MIT map



Fonctionnalités

Le plan interactif du MIT affiche le tracé des bâtiments ainsi que le nom des de ceux-ci. Seul les légendes s'adaptent en fonction du zoom.

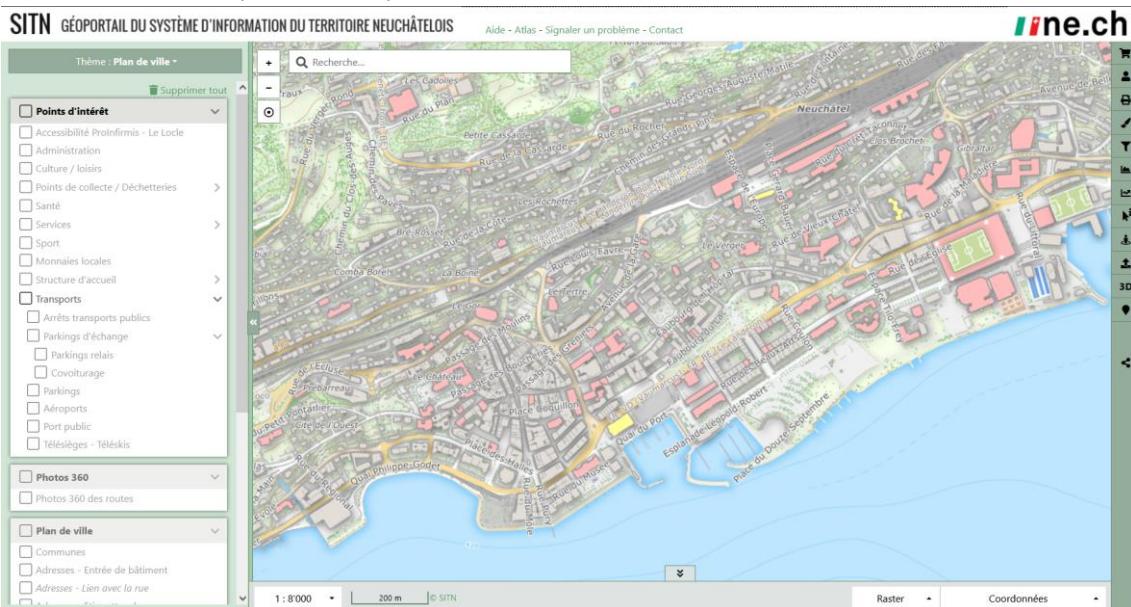
Un utilisateur peut rechercher des bâtiments ou des points d'intérêts lié à l'université (ex : le world wide web consortium W3C).

Il peut aussi appliquer quelque filtre pour afficher des repères comme les restaurants. Cliquer sur un bâtiment permet d'obtenir des informations sur celui-ci. D'autres outils sont fournis comme un outil de partage ou d'impressions

Technologies

L'affichage de la carte s'effectue à l'aide de l'api google maps. Ceci permet aussi un accès à google street view.

4.1.4 SITN – Géoportail du système d'information du territoire neuchâtelois



Le plan du SITN Affiche les différentes informations géographiques du canton de Neuchâtel. Le niveau de détail de la carte s'adapte en fonction du zoom. Par exemple les numéros des maisons selon le cadastre de chaque commune s'affichent lors d'un zoom à une échelle 1 :1000.

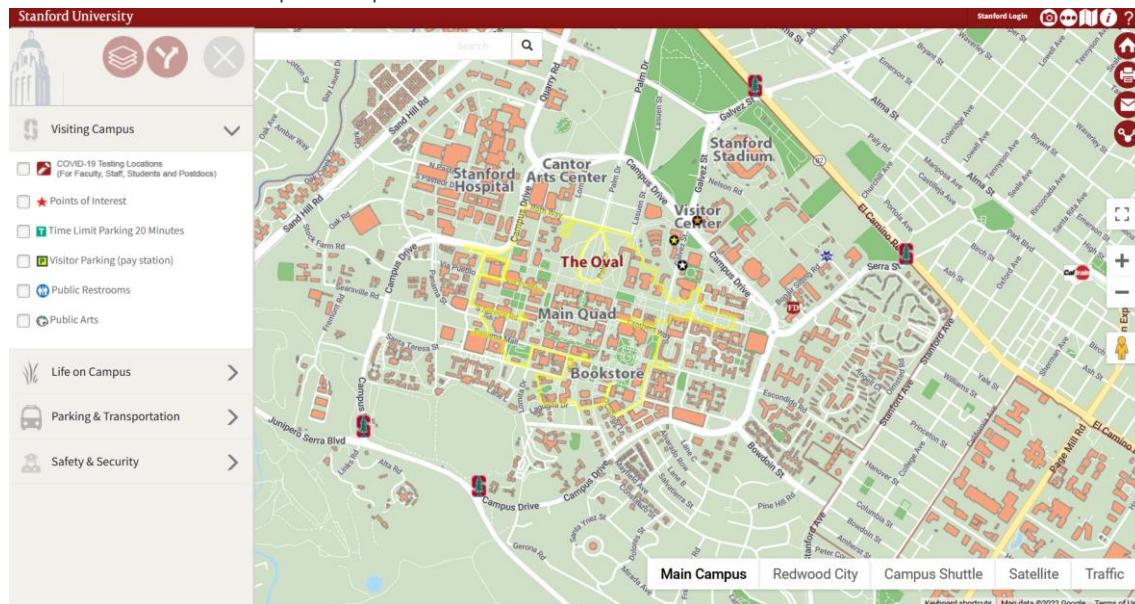
Il y a la possibilité d'appliquer plusieurs filtres afin d'obtenir les informations recherchées comme le tracé des communes ou les points d'intérêts.

L'outil offre aussi des outils de dessin vectoriel, d'impression, la possibilité de changer le fond du plan de changer un accès à google street map, et un accès au géoportail LIDAR

Technologies

Le site utilise GeoMapFish pour l'affichage des cartes ainsi que l'api google maps pour google street view.

4.1.5 Stanford campus map



Fonctionnalités

Le carte du campus de Stanford affiche le tracé des bâtiments ainsi qu'une légende précisant le nom de ceux-ci. Seul l'affichage de la légende varie selon le zoom : affiche seulement les légendes lors d'un affichage rapproché sur les bâtiments.

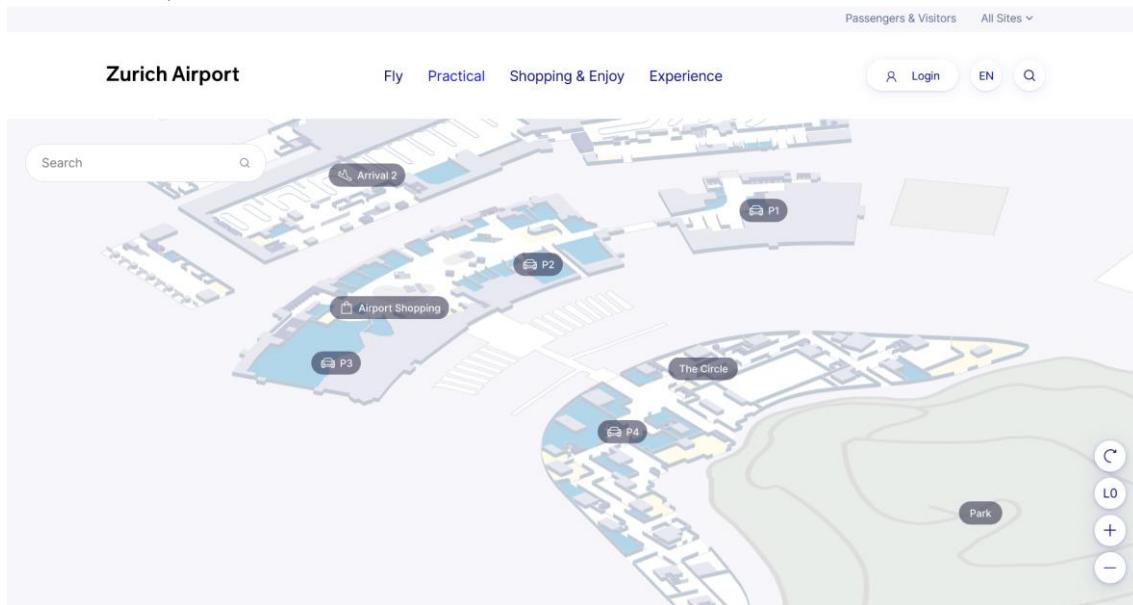
Le site offre un outil de recherche de ressources aux utilisateurs, un accès à open street view, un outil d'impression et un outil de partages.

A noter que le menu en bas à droite est un mélange de plusieurs fonctionnalités ce qui peut perdre un nouvel utilisateur : changer de site, accéder à une application offrant les informations sur les navettes et le changement de fond de plan.

Technologies

L'application web utilise l'api google maps pour l'affichage de la carte.

4.1.6 Aéroport de Zurich



Fonctionnalités

L'aéroport de Zurich offre un plan non géoréférencé en 3d isométrique. Les différents points d'intérêts affichés varient en fonction du zoom. L'application offre un outil de recherche avec des suggestions par thème. On peut aussi changer le fond du plan en fonction de l'étage.

Le plan ne sert que pour l'orientation des voyageurs et est donc pauvre en fonctionnalités annexes.

Technologies

Le site utilise le framework réactif React js, ainsi que le module bundler webpack. Les plans sont des images bitmap (pixellisé).

4.1.7 Conclusion de l'analyse de système existant

On peut distinguer deux types d'application : des plans interactifs qui aident les utilisateurs à s'orienter et les Géoportails qui fournissent des informations géographiques voire construire des nouvelles informations à l'aide d'outils de dessin vectorielle.

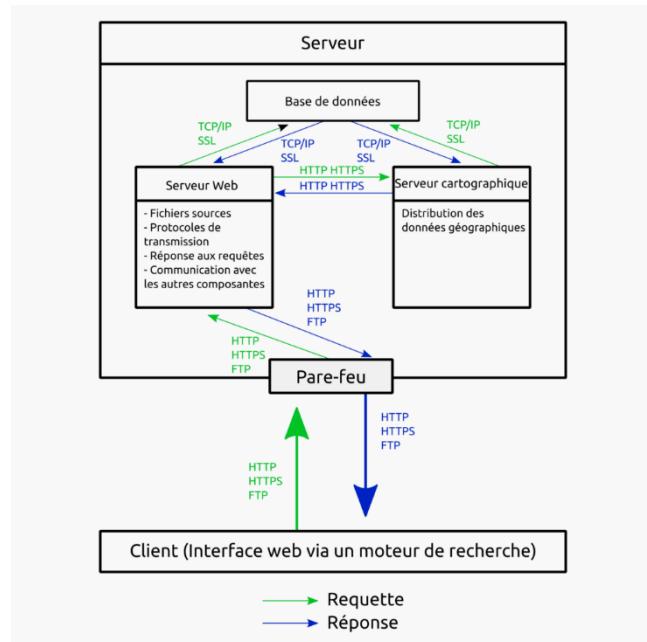
Cette distinction a été faite par l'EPFL qui sépare sur deux sous-domaines différents les deux types d'application.

Les technologies utilisées sont aussi différentes : les sites les plus complets utilisent souvent GeoMapFish alors que les sites les plus simples utilisent plutôt l'API Google Maps.

4.2 Architecture webGIS

Une application webGIS est composée côté serveur d'une base de données stockant les données géographiques, un serveur cartographique qui gère et distribue les données géographiques. La distribution se fait à l'aide des protocoles WMS/WMTS/WFS.

Finalement un serveur web envoie la page et les données géographique au client.



4.3 Technologies

4.3.1 Openlayers

Openlayer est une librairie JavaScript open source qui facilite la mise à disposition de carte dynamique. Elle permet de rajouter facilement des calques en dessus d'un fond de carte. Elle peut lire plusieurs formats de données comme du geojson ou du KML.

4.3.2 Leaflet

Leaflet est une librairie open source concurrente à openlayers. Elle est plus simple d'utilisation et légère mais moins complète que openlayers. On peut rajouter des plugins à la librairie.

4.3.3 Google Maps API

L'API de google maps n'est pas open source et peut être payant. Elle n'offre pas la possibilité de rajouter des calques au fond de carte.

4.3.4 GeoMapFish

GeoMapFish est une technologie open source développée par camptocamp et permet de faciliter la construction de système d'information géographique pour le web (webSIG). Il est composé de ngeo pour le frontend (<https://github.com/camptocamp/ngeo/>) et de c2cgeoportal (<https://github.com/camptocamp/c2cgeoportal/>) pour le backend.

Ngeo est une librairie JavaScript qui facilite le développement d'application basé sur le framework réactif Angular Js et l'api openLayers. Elle utilise aussi webpack comme module bundler.

C2cGeoportal est la partie serveur construit à partir d'une image Docker. Il faut avoir une connaissance en Python pour l'utiliser.

4.3.5 Serveur cartographique

Qgis server est un serveur cartographique open source gérant les protocoles WMS WFS et WCS écrit en c++.

ArcGIS server est un composant delà suite ArcGIS entreprise. Ce logiciel est propriétaire et payant.

Geoserver est un serveur cartographique open source écrit en java. Il intègre la librairie openlayers.

Mapserver est le premier serveur cartographique. Il supporte les protocoles WMS WFS et WCS

4.3.6 PostgreSQL et PostGIS

PostgreSQL est une base de données relationnelle open source, elle permet de stocker des données géographiques avec l'extension PostGIS. Cette solution est utilisée par c2cgeoportal.

4.4 Données existantes à disposition

4.4.1 Plans

Des plans des étages des sites de Cheseaux et St-roch non géoréférencé ont été fournis au format DGW, format propriétaire de AutoCAD. Il faudra les géoréférencé à l'aide d'un logiciel GIS dans une projection compatible avec les technologies choisis à l'aide d'une transformation de Helmert.

4.4.2 Serveur LDAP

Le serveur LDAP de la HEIG-VD peut fournir des informations sur certaines ressources.

4.5 Solution

La solution consiste en une application web qui mettra à disposition une visualisation des plans des sites de la HEIG-VD. Elle devra fournir rapidement les informations souhaitées à l'utilisateur.

Elle sera développée à l'aide des outils GeoMapFish : ngeo et c2cgeoportal.

Elle doit être accessible sur téléphone mobile, tablette et ordinateur (responsive), fournir un affichage des plans pour chaque étage des sites de Cheseau et de St-Roch, fournir des filtres pour modifier l'affichage de l'interface.

Elle peut fournir des informations sur des ressources, un outil de localisation de ressources par nom et/ou par critères, un outil de plus court itinéraire entre deux ressources, un outil de dessin et un outil d'impression.

5 Exigences

| # | Exigence | Min. | Nom | Max | Unité |
|----|--|------|-----|-----|----------|
| R1 | Temps de réponse affichage de la carte | | 1 | 3 | s |
| R2 | Temps de réponse action utilisateur | | 1 | 2 | s |
| R3 | Application utilisable sur écran de diagonale de | 4 | 5 | | Pouces |
| R4 | Coût de maintenance en production | | | 500 | CHF/mois |

6 Bibliographie

Plan epfl : <https://plan.epfl.ch>

Géoportail epfl : <https://geoportail.epfl.ch/>

SITN : <https://sitn.ne.ch>

MIT campus map : <https://whereis.mit.edu/>

Standford campus map : <https://campus-map.stanford.edu/>

Aéroport de Zurich :

<https://www.flughafen-zuerich.ch/en/passengers/practical/guidance/interactive-map>

Informations architecture webGIS :

<https://veillecarto2-0.fr/2020/04/29/reflechir-son-architecture-dapplication-cartographique-du-serveur-au-client/>

Openlayers : <https://openlayers.org/>

Leaflet : <https://leafletjs.com/> ou <https://leafletjs.com/SlavaUkraini/> (adresse temporaire due à la guerre en Ukraine)

Google maps api : <https://developers.google.com/maps>

Geomapfish : <https://geomapfish.org/fr/index.html>

C2cgeoportal documentation : <https://camptocamp.github.io/c2cgeoportal/master/>

QGIS documentation : <https://docs.qgis.org/3.22/en/docs/index.html>

Geoserver documentation : <https://docs.geoserver.org/>

Mapserver : <https://mapserver.org/>

ArcGIS server :

<https://enterprise.arcgis.com/en/server/latest/get-started/windows/what-is-arcgis-for-server-.htm>

PostGIS : <https://postgis.net/>

Glossaire

container Entité contenant l'application ainsi que les autres programmes pour son fonctionnement. 12

CSS Norme de fichier permettant de modifier l'aspect du contenu d'une page. 12, 13, 14, 28

diagrams.net Site web permettant de créer des diagrammes. 3

Docker Programme gérant les container. 15, 34

EPFL Ecole polytechnique fédéral de Lausanne. 4

Express Librairie simplifiant la mise en place d'un serveur sur Node. 15, 26

GeoJson Norme pour les fichiers Json permettant de stocker des données géographiques. 25

Git Programme de versionning. 15

GitLab Hébergeur de code source. 15

GitLab container registry Hébergeur d'image de container. 16, 35

GitLab CI/CD Outil permettant la mise en place d'un pipeline CI/CD. 15

HEIG-VD Haute École d'Ingénierie et de Gestion du canton de Vaud. 1, 2, 11, 12

HTML Norme de fichier décrivant le contenu d'une page web. 12, 13, 14, 28

JavaScript Language de programmation utilisé par les navigateurs web afin de modifier les contenus des applications web. 12, 13, 14, 27, 28

Nginx Serveur Http. 15

Node Programme permettant d'utiliser JavaScript en dehors des navigateurs web. 14, 15

NPM Programme utilisé pour gérer les librairies Node. 14

Openlayers Librairie JavaScript permettant de créer des webSig. 14, 17, 29, 32

Pinia Librairie associée à Vue 3 permettant de simplifier la communication de données entre les différents components. 14

PostGis Extension de Postgres rajoutant la possibilité de stocker des données géographiques. 15, 27, 35

PostgreSQL Base de données relationnelle. 15, 26

QGis logiciel de systèmes d'informations géographiques. 23, 25

Reverse-proxy Permet d'accéder à différents serveurs en utilisant le même URL. 12, 15

Traefik Programme servant de reverse-proxy. 15, 35

TypeScript Language de programmation. 14, 29

UML Notation pour la modélisation d'applications en ingénierie logiciel. 3, 16

Vite Un environnement de développement. 15

Vitest Librairie permettant de créer des tests unitaires. 15

Vue test Utils v2 Librairie permettant de tester les composants Vue. 15

Vue 3 Librairie JavaScript simplifiant les interactions avec le contenu. 14, 15

webSIG systèmes d'interface web de visualisation de données géographiques. vii, 4, 8, 9, 14, 18, 19, 40

Bibliographie

- [dia05] DIAGRAMS.NET. *Outil de création de diagrammes*. 2005. URL : <https://www.diagrams.net/> (cf. p. 3).
- [Goo05] GOOGLE. *Google maps api*. 2005. URL : <https://developers.google.com/maps> (cf. p. 9).
- [Pos05] POSTGIS. *PostGIS*. 2005. URL : <https://postgis.net/> (cf. p. 9).
- [aér10] Zurich AÉROPORT. *Plan du campus de Stanford*. 2010. URL : <https://campus-map.stanford.edu/> (cf. p. 7).
- [HEI10] HEIG-VD. *Plan de la HEIG-VD*. 2010. URL : <https://heig-vd.ch/campus/mobilite/venir-a-la-heig-vd> (cf. p. 1).
- [SIT10] SITN. *Géoportail du système d'informations du territoire neuchâtelois*. 2010. URL : <https://sitn.ne.ch/> (cf. p. 6).
- [Sta10] STANFORD. *Plan du campus de Stanford*. 2010. URL : <https://campus-map.stanford.edu/> (cf. p. 6).
- [Lea11] LEAFLET. *Leaflet*. 2011. URL : <https://leafletjs.com/> (cf. p. 9).
- [cam12] CAMPTOCAMP. *camptocamp/c2cgeoportal*. 2012. URL : <https://github.com/camptocamp/c2cgeoportal/> (cf. p. 9).
- [MIT13] MIT. *plan du campus du MIT*. 2013. URL : <https://whereis.mit.edu/> (cf. p. 5).
- [Ope13] OPENLAYERS. *Openlayers*. 2013. URL : <https://openlayers.org/> (cf. p. 8).
- [cam14] CAMPTOCAMP. *camptocamp/ngeo*. 2014. URL : <https://github.com/camptocamp/Ngeo/> (cf. p. 9).
- [EPF18a] EPFL. *Géoportail de l'EPFL*. 2018. URL : <https://geoportail.epfl.ch/> (cf. p. 5).
- [EPF18b] EPFL. *Plan de l'EPFL*. 2018. URL : <https://plan.epfl.ch> (cf. p. 4).