

Master of Science HES-SO in Engineering  
Av. de Provence 6  
CH-1007 Lausanne

# Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication

## Développement d'une Application Web pour la Visualisation et la Recherche de Données Médicales

Fait par

**Kewin Dousse**

Sous la direction de  
Prof. Sandy Ingram  
à la HEIA-FR

Dr. Alexandre Terrier (EPFL), Prof. Alain Farron (CHUV), Dr. Fabio Becce (CHUV)

Lausanne, HES-SO//Master, 2017

## Résumé

Le but de ce projet est de concevoir et développer une application Web multi-plateforme pour la recherche et la visualisation de données médicales.

**Keywords.** Web, Visualisation

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Contexte . . . . .	6
1.2	Objectifs . . . . .	6
1.3	Contraintes . . . . .	7
1.4	Méthodologie . . . . .	7
<b>2</b>	<b>Analyse</b>	<b>8</b>
2.1	Analyse des besoins . . . . .	8
2.1.1	Enquête . . . . .	8
2.1.2	Besoins cliniques . . . . .	9
2.1.3	Besoins scientifiques . . . . .	9
2.2	Analyse technologique . . . . .	10
2.2.1	Base de données . . . . .	10
2.2.2	Technologies . . . . .	10
<b>3</b>	<b>Prototype 1</b>	<b>19</b>
3.1	Fonctionnalités . . . . .	19
3.2	Conception . . . . .	19
3.2.1	Architecture générale de l'application . . . . .	19
3.2.2	Structure de la navigation . . . . .	20
3.2.3	Maquettes . . . . .	21
3.3	Implémentation frontend . . . . .	22
3.3.1	Disposition générale des éléments sur la page . . . . .	24
3.3.2	Routage . . . . .	25
3.3.3	Pages "Patients" et "Cases" . . . . .	26
3.3.4	Communication des données . . . . .	27
3.4	Implémentation backend . . . . .	28
3.4.1	Structure du backend . . . . .	28
3.4.2	Express.js . . . . .	30
3.4.3	Base de données . . . . .	30
3.5	Evaluation . . . . .	31

3.5.1	Heuristique et adaptations . . . . .	31
3.5.2	Validation par les clients . . . . .	31
<b>4</b>	<b>Prototype 2</b>	<b>32</b>
4.1	Fonctionnalités . . . . .	32
4.2	Conception . . . . .	32
4.3	Pages d'édition . . . . .	32
4.4	Recherche . . . . .	33
4.5	Implémentation . . . . .	33
4.5.1	Pages d'édition . . . . .	33
4.5.2	Recherche . . . . .	35
4.6	Evaluation heuristique et adaptations . . . . .	36
4.7	Résultat . . . . .	36
<b>5</b>	<b>Conclusion</b>	<b>39</b>
5.1	Conclusion du projet . . . . .	39
5.1.1	Délivrables . . . . .	39
5.1.2	Conclusion générale . . . . .	39
5.1.3	Perspectives . . . . .	40
5.2	Conclusion personnelle . . . . .	40
<b>A</b>	<b>Historique des versions</b>	<b>46</b>
<b>B</b>	<b>Cahier des charges</b>	<b>47</b>
B.1	Activités . . . . .	47
B.2	Planification . . . . .	48
B.3	Diagramme de Gantt . . . . .	49
<b>C</b>	<b>Documentation</b>	<b>50</b>
C.1	Localisation . . . . .	50
C.2	Contenu . . . . .	50
C.2.1	Forge . . . . .	50
<b>D</b>	<b>Procès-verbaux</b>	<b>51</b>

# Table des figures

2.1	Le schéma de la base de données reçue . . . . .	11
2.2	Le Virtual DOM est une étape intermédiaire entre React et le DOM[2]	12
2.3	Fichier <code>package.json</code> . . . . .	13
2.4	Code JSX . . . . .	14
2.5	Equivalent compilé en JavaScript . . . . .	14
2.6	Code d'un composant " <code>ShoppingList</code> ", tiré de [11] . . . . .	15
2.7	Code modifié en utilisant un Composant . . . . .	16
2.8	Code d'un fichier HTML affichant un Composant React. . . . .	16
2.9	L'architecture que propose Flux est unidirectionnelle.[13] . . . . .	17
3.1	La structure de la hiérarchie et navigation entre les vues . . . . .	21
3.2	Wireframe sur papier de l'interface de la liste des cas . . . . .	22
3.3	Maquette sur papier de l'interface d'un cas particulier . . . . .	23
3.4	Sketch sur papier des pages "Stats" et "Admin" . . . . .	23
3.5	Premier prototype fonctionnel de l'application . . . . .	24
3.6	Utilisation des composants <code>Router</code> et <code>Route</code> . . . . .	25
3.7	Premier prototype de la page "Patients" . . . . .	27
3.8	Flux de données de l'application . . . . .	28
3.9	Schéma de l'architecture réseau de l'EPFL utilisée par VisMed . . . . .	29
4.1	Prototype de la page de modification d'un cas . . . . .	33
4.2	Prototype de la liste des cas avec recherche de mot clé . . . . .	34
4.3	Mise en évidence du champ modifié . . . . .	35
4.4	Page "Cases" mise à jour, montrant la recherche de l'information 'P24' . . . . .	37
4.5	Page finale d'édition d'un cas . . . . .	37
4.6	Page "Patients" mise à jour, montrant la recherche de l'information '1980' . . . . .	38
4.7	Page finale d'édition d'un patient . . . . .	38

# Chapitre 1

## Introduction

### 1.1 Contexte

L'hôpital orthopédique du CHUV utilise actuellement un fichier Excel partagé pour conserver les données médicales des patients. Bien que fonctionnelle, cette méthode est fastidieuse à la fois pour l'entrée de nouvelles données que pour la recherche parmi celles-ci, et ne possède pas de réel point positif autre que la facilité initiale de la mise en place.

Le but de ce projet est de concevoir et développer une application Web multiplateforme pour la recherche et la visualisation de ces données médicales. Le projet comporte deux challenges techniques centraux :

1. L'architecture désirée nécessitera donc un développement sur toutes les couches du système, autrement dit « full stack ».
2. L'aspect multiplateforme nécessitera l'utilisation d'outils et de librairies récentes.

Ce projet est mené dans un but double : Il aura à la fois une utilité scientifique en tant que base de données pour des futures études, et une utilité clinique par l'utilisation de son interface par le personnel.

### 1.2 Objectifs

À la fin du projet, nous serons en possession d'une application web multiplateforme permettant de :

- Rechercher et visualiser des données médicales
- Rechercher, visualiser et modifier des données de patients
- Insérer des données médicales

Un objectif est qualifié de secondaire, et sera réalisé en fonction de l'avancement du projet :

- Produire et montrer des statistiques sur les patients et les cas

## 1.3 Contraintes

Le développement du projet va se faire autour de quelques contraintes définies :

- L'application doit converser avec une base de données MySQL déjà existante.
- L'application doit être adaptée à une utilisation à la fois sur un appareil desktop, et sur un appareil mobile, ainsi que convenir à la plupart des navigateurs récents.

## 1.4 Méthodologie

Le développement se fera de manière itérative, afin de produire plusieurs maquettes et prototypes, avec des degrés de fidélité augmentant au fil du projet. Le but est d'utiliser une méthodologie agile afin de pouvoir rapidement proposer des prototypes, et avoir des retours de clients assez tôt. On désire s'assurer que l'on produit une solution adaptée aux besoins du client.

# Chapitre 2

## Analyse

### 2.1 Analyse des besoins

#### 2.1.1 Enquête

Afin de comprendre correctement les demandes des clients, il a été nécessaire de mener une enquête sur le public cible en procédant à une interview de quelques personnes concernées parmi le public cible. Une rencontre a donc eu lieu avec Mme. Sandy Ingram, M. Alain Farron, M. Alexandre Terrier et M. Fabio Becce. Cette réunion a donné lieu à plusieurs questions et des réponses de la part de toutes les personnes, qui ont aidé à définir quels étaient les besoins du public ciblé.

La solution sera donc utilisée par deux catégories de personnes qui vont vouloir en tirer des informations différentes :

- Des médecins vont vouloir utiliser l'application pour visualiser et modifier les données des patients directement.
- Des scientifiques vont vouloir, dans un deuxième temps, extraire des informations statistiques sur les données elles-mêmes, afin de pouvoir par exemple mener des études approfondies sur les nombres.

En plus des informations du public cible, nous avons appris plusieurs informations essentielles sur le fonctionnement demandé de plusieurs des pages :

- Un cas peut avoir plusieurs CT scans associés à lui.
- Pour un patient, on aimerait voir la liste de ses cas, et rapidement voir son cas "primaire" (le plus récent).
- Pour un patient, on aimerait voir principalement son diagnostique, et son traitement.
- Pour un cas, on aimerait voir tous les scans, ainsi que toutes les mesures de ces scans.

Ces critères ont été pris en compte pour la suite du développement de l'application. Certains points sont restés flous, mais non bloquants pour l'avancement de

l'application ; est-ce que l'application doit permettre l'ajout de mesures ? Est-ce que l'application doit permettre l'ajout de patients ? Mais ces mêmes questions ont été éclaircies à l'avenir par des échanges de mails, et également une réunion par Skype.

En conclusion de cette réunion, nous pouvons tout de même regrouper les besoins du client en deux catégories.

### 2.1.2 Besoins cliniques

Des médecins et du personnel soignant vont être encouragés à utiliser l'application, et ceux-ci veulent pouvoir accéder à une vue par patient, ce qui est le plus naturel pour eux. Leur but sera d'obtenir, de visualiser et de modifier des données reliées à un patient. Il sera plus important de pouvoir accéder aux données complètes d'une personne en particulier, ainsi que tous les cas associés à cette personne. En effet, une personne peut être enregistrée dans la base de données pour plusieurs raisons : Un "cas" correspond à une anomalie à traiter. Une personne peut par exemple en accumuler plusieurs sur le temps. Il sera dans ce cas probablement intéressant de regarder l'évolution d'une personne sur le temps, regroupant tous les cas liés.

Cette catégorie de personne veut pouvoir afficher ou cacher certaines catégories d'informations. Par exemple, il arrivera qu'un médecin veuille consulter rapidement une fois les données des scans du patient, et une autre fois les données concernant son traitement. La solution est donc de ne pas tout afficher directement, mais de laisser à l'utilisateur la possibilité de filtrer les résultats. Certaines informations seront affichées par défaut car très souvent consultées, comme par exemple le traitement actuel du patient ou son diagnostic. Mais d'autres catégories d'informations telles que les mesures des scans ne seront pas visible d'office et pourront être activées lors de la recherche.

### 2.1.3 Besoins scientifiques

L'application va cibler une catégorie d'utilisateurs qui est intéressé à obtenir des informations brutes sur les données de la base elle-même afin de les intégrer par exemple au sein d'études scientifiques, et dans un deuxième temps de produire des statistiques. Pour eux, les données spécifiques à un patient sont moins importantes, mais il est capital de pouvoir naviguer parmi les informations de la manière la plus structurée possible. Le but sera de regrouper les informations par cas, et non pas par personne. Il a été défini selon les besoins que plusieurs scans peuvent être liés à un seul cas.

Cette catégorie de personne va vouloir accéder aux données "brutes" plus rapidement : Mesures des scans, images des scans par exemple. Comme pour la

vue des patients, la vue des cas affichera certaines qu'une partie des informations par défaut pour des raisons de lisibilité. Ici à nouveau, il sera possible d'activer ou de désactiver l'affichage d'autres catégories d'informations qui sont ici moins importantes, comme par exemple le diagnostic du patient.

## 2.2 Analyse technologique

Le projet comprend un challenge technologique : Le but est d'utiliser une ou plusieurs technologies de pointes dans le web. Mais bien qu'il s'agisse du challenge principal, plusieurs aspects surviennent lors de l'analyse des différentes couches de l'application.

### 2.2.1 Base de données

Le client possède actuellement les données des patients dans un fichier Excel. Afin de migrer ces données vers une base de données, celui-ci nous a envoyé un schéma montrant la structure qu'il a mise en place. La figure 2.1 présente ce fichier qui, après une analyse, semble correspondre aux besoins qu'il demande ainsi qu'à la structure actuelle des données du fichier Excel.

Nous allons donc prendre ce schéma comme référence pour la suite de l'analyse et du développement du projet.

### 2.2.2 Technologies

En plus de l'approche orientée utilisateur du projet, le but est également de découvrir et d'apprendre à utiliser certaines des plus récentes librairies du web. Pour ce projet, le choix logique a été fait de se concentrer une technologie servant à créer des interfaces utilisateur : React.

Pour l'apprentissage de cette technologies et des librairies qui lui sont proches, j'ai suivi un tutoriel en plusieurs vidéo. Cette liste de vidéo est disponible sur YouTube[14].

#### React

**Approche** Développée par Facebook le milieu de l'année 2013, React est "Une librairie JavaScript pour la création d'interfaces utilisateur"[1], traduit littéralement depuis le site officiel. React s'occupe donc strictement de la vue de l'application. Ce qui démarque principalement React des autres librairies similaires comme AngularJS est sa performance.

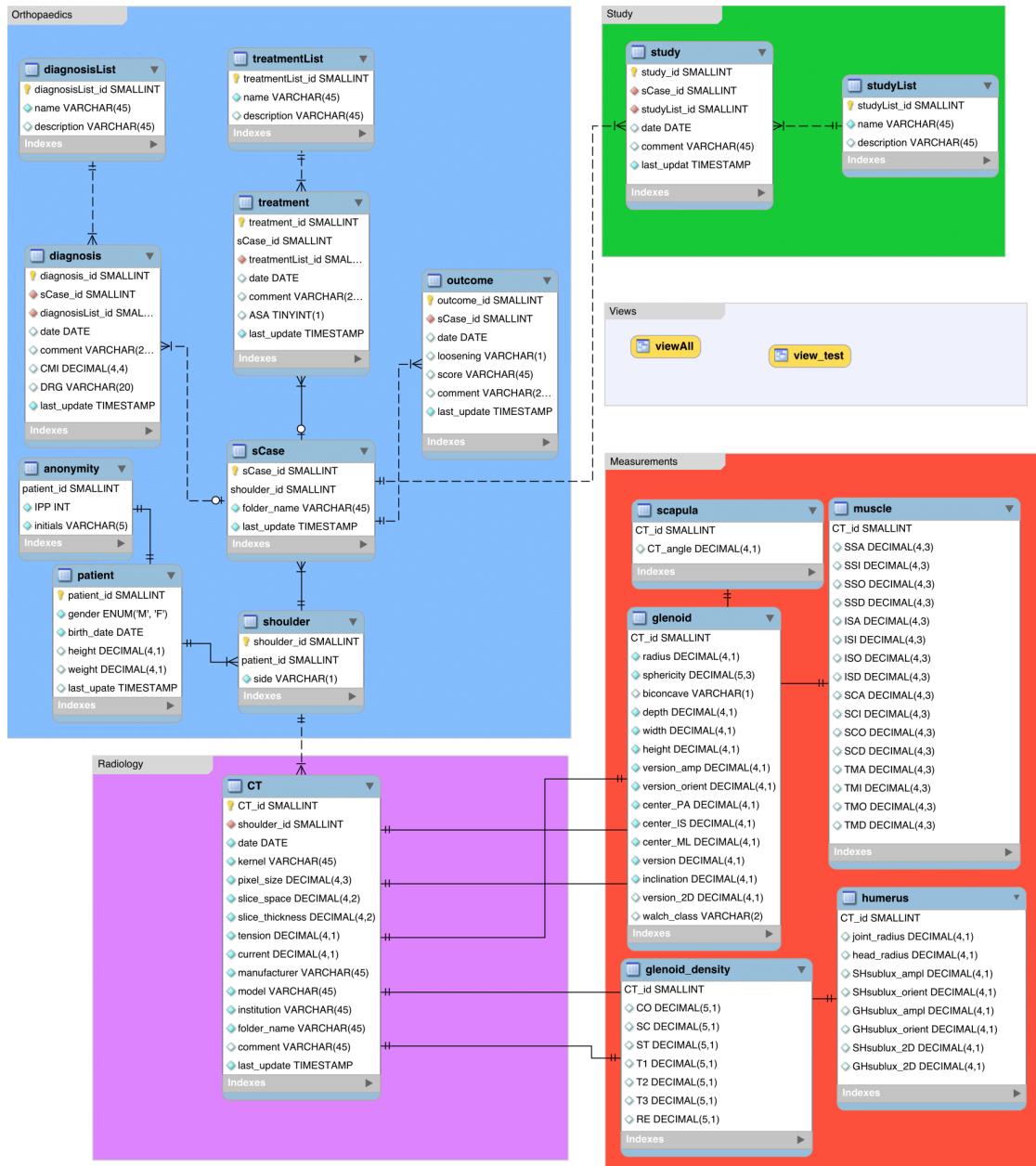


FIGURE 2.1 – Le schéma de la base de données reçue.

La principale raison des performances élevées de React est dans le fait de conserver, en plus de la page affichée actuellement à l'écran (le DOM), un "DOM virtuel". Ce DOM virtuel est en pratique une copie en mémoire des éléments affichés à l'écran. La gestion habile de ce DOM virtuel permet à React de n'actualiser que les strictes parties de la page qui ont besoin de l'être, au moment où React

le décide. Cela lui permet d'éviter beaucoup de rafraîchissements inutiles, que les autres librairies se devraient d'effectuer.

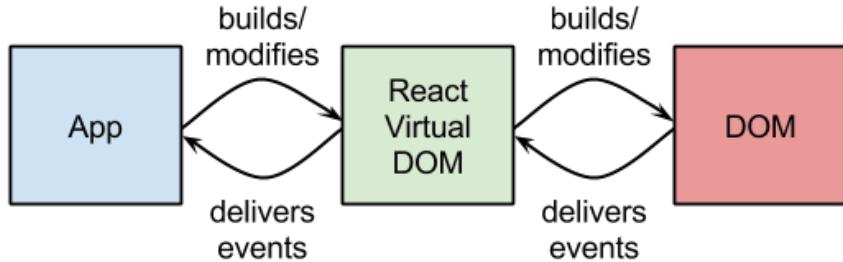


FIGURE 2.2 – Le Virtual DOM est une étape intermédiaire entre React et le DOM[2]

Le fait que le changement du DOM soit le principal élément lent dans presque tout code JavaScript permet donc à React de se démarquer.

Une autre des caractéristiques de React est l'encouragement à écrire des composants réutilisables. De plus, bien que React ait été designé pour créer des interfaces web, il est aujourd'hui possible d'utiliser React ainsi que ses composants pour créer bien d'autres interfaces que du HTML. Par exemple, Facebook développe également depuis 2015 "React Native", qui permet d'utiliser des composants React pour créer des applications smartphone natives. Depuis, encore d'autres librairies ont constaté la puissance de React et tentent d'en tirer parti en dehors du navigateur.

**Tutoriel** Bien que React soit une librairie JavaScript, il est très recommandé de se construire un environnement de développement avec des outils adaptés afin de pouvoir développer correctement. Avec l'arrivée des gestionnaires de paquets tels que npm pour JavaScript, il serait une perte de temps de s'encombrer manuellement de toutes les étapes de recherche de dépendances de librairies, de leur assemblage, etc. Nous n'allons pas nous étendre sur tous les fichiers de configurations, mais allons lister les outils utiles pour React et allons expliquer la logique de fonctionnement principale de la librairie.

**Node.js et npm** Nous allons partir du principe que la version 6.10.1 LTS de Node.js[3] (ainsi que npm[4], qui est installé avec) sont installés sur la machine.

Afin d'utiliser npm, nous allons créer un fichier de package, nommé 'package.json'. Ce fichier représente notre package d'application, et liste les librairies nécessaires

pour le projet.

**Dépendances** React lui-même est très puissant, mais il ne s'agit que du cœur logique du framework. D'autres librairies sont utilisées en cohésion avec React afin de par exemple générer du html. La figure 2.3 'montre' le contenu d'un fichier 'package.json' d'exemple, exposant des librairies quasi indispensables pour l'utilisation de React.

```

1  {
2    "name": "tutoriel",
3    "version": "0.0.1",
4    "description": "",
5    "main": "webpack.config.js",
6    "dependencies": {
7      "react": "^15.4.2",
8      "react-dom": "^15.4.2",
9      "react-router": "^4.0.0",
10     "react-router-dom": "^4.0.0-beta.8",
11     "react-tap-event-plugin": "^2.0.1",
12     "webpack": "^2.2.1",
13     "webpack-dev-server": "^2.4.2"
14   },
15   "devDependencies": {},
16   "scripts": {
17     "dev": "webpack-dev-server --content-base src --inline --hot",
18   },
19   "author": "",
20   "license": "ISC"
21 }
```

FIGURE 2.3 – Fichier package.json

On peut y voir les dépendances suivantes :

- **react** : Coeur logique de React.
- **react-dom** : Permet à React d'interagir avec le DOM.
- **react-router** : Permet à React d'interpréter des liens.
- **react-router-dom** : Fait le lien entre **react-router** et **react-dom**.
- **react-tap-event-plugin** : Corrige un bug pour que les interactions fonctionnent sur smartphone.
- **webpack** : Permet de "compiler" plusieurs scripts et librairies en un seul fichier JavaScript.
- **webpack-dev-server** : Permet de simplifier l'utilisation de **webpack** lors du développement.

**webpack et JSX** **webpack**[5] est ici utile pour n'obtenir qu'un seul fichier JavaScript, très simple à utiliser avec une page HTML. Il est très recommandé de

l'utiliser en cohésion avec React, mais son utilisation est optionnelle et sa configuration ne sera pas discutée ici. Les morceaux de code qui apparaîtront utiliseront tout de même la syntaxe JSX[10], qui est un sucre syntaxique pour JavaScript. Il permet principalement d'écrire du code HTML dans du JavaScript. webpack, ainsi que d'autres technologies similaires, permettent de compiler cette syntaxe en JavaScript classique. Il est aussi tout à fait possible de ne pas utiliser JSX et d'écrire du JavaScript conventionnel simplement. La figure 2.4 montre un code JSX, et la figure 2.5 montre son équivalent JavaScript.

```

1 <MyButton color="blue" shadowSize={2}>
2   Click Me
3 </MyButton>
```

FIGURE 2.4 – Code JSX

```

1 React.createElement(
2   MyButton,
3   {color: 'blue', shadowSize: 2},
4   'Click Me'
5 )
```

FIGURE 2.5 – Equivalent compilé en JavaScript

**Versions des librairies** Il est également à noter que React et d'autres librairies qu'il utilisent sont encore à ce jour énormément en développement, et il n'est pas rare de voir l'API changer, parfois excessivement souvent (j'ai pu voir des séries de changements conséquents à moins de 6 mois d'écart pour certaines librairies).

Il est donc important de remarquer les versions utilisées des librairies, sur lesquelles se baseront à la fois le code du projet ainsi que ce guide d'utilisation. À noter également que les versions notées ici peuvent être utilisées ensemble, mais que cela n'est pas toujours le cas : Très souvent, une version d'une librairie va demander une version minimum d'une autre.

**Composants** Le développement d'une application utilisant React repose sur la déclaration et l'utilisation de Composants. Un composant peut être vu comme un élément HTML possédant sa propre intelligence, indépendant des autres parties de la page. La figure 2.6 montre un exemple de code d'un composant simple.

Voici les éléments qui constituent ici ce Composant :

```

1 class ShoppingList extends React.Component {
2   render() {
3     return (
4       <div className="shopping-list">
5         <h1>Shopping List for {this.props.name}</h1>
6         <ul>
7           <li>Instagram</li>
8           <li>WhatsApp</li>
9           <li>Oculus</li>
10        </ul>
11      </div>
12    );
13  }
14}
15 // Example usage: <ShoppingList name="Mark" />

```

FIGURE 2.6 – Code d'un composant "ShoppingList", tiré de [11]

- La méthode `render()` doit retourner une description de l'élément à afficher, soit ici un élément HTML (décrit avec la syntaxe JSX) en contenant plusieurs autres. Il s'agit là de la représentation qu'aura ce Composant lorsqu'il sera affiché dans la page HTML.
- Le paramètre `this.props.name`. Il s'agit là de code JavaScript : `this` se réfère à cette classe React, et `props` est un objet. `props` signifie 'propriétés', et c'est cet objet qui va généralement contenir les propriétés d'un élément React. Ici, une propriété est le nom `name`.

**Composition de Composants** Ici, le composant se constitue uniquement d'éléments HTML. Mais l'intérêt vient de la possibilité d'utiliser d'autres Composants à l'intérieur de la définition d'un de ces Composants. Par exemple, si nous avons décrit un Composant `Liste` dans notre code, nous pourrions l'utiliser ici à la place de l'élément HTML `ul` par exemple. Ainsi, la figure 2.7 montre cette modification apportée à notre décret précédemment.

**Propriétés** Nous avons ici remplacé l'élément HTML par un composant fictif `<List>`. Celui-ci prend en paramètre une liste de valeurs. Dans notre cas, celles-ci seront ensuite passées à l'intérieur de son objet `props`, qu'il pourra faire usage pour afficher notre liste correctement.

Nous sommes donc ici en possession d'un Composant fonctionnel très simple. Il serait tout à fait possible d'ajouter d'autres fonctionnalités à notre Composant : Il se comporte comme une Classe, et nous avons donc tout loisir de lui implémenter d'autres fonctions que l'actuel `render()` afin de le rendre réellement intelligent. De plus, React propose d'autres mécanismes que `props` pour faire interagir les

```

1 class ShoppingList extends React.Component {
2   render() {
3     return (
4       <div className="shopping-list">
5         <h1>Shopping List for {this.props.name}</h1>
6         <List values={['Instagram', 'WhatsApp', 'Oculus']}>
7       </div>
8     );
9   }
10 }
11
12 // Example usage: <ShoppingList name="Mark" />

```

FIGURE 2.7 – Code modifié en utilisant un Composant

composants entre eux, mais nous en savons déjà suffisamment pour comprendre le fonctionnement de base de React ici.

```

1 <html>
2   <body>
3     <div id="app"></div>
4     <script src="script.js"></script>
5     <script>
6       ReactDOM.render(
7         <ShoppingList />,
8         document.getElementById('app')
9       );
10    </script>
11  </body>
12 </html>

```

FIGURE 2.8 – Code d'un fichier HTML affichant un Composant React.

**Affichage sur une page HTML** À présent, nous désirons afficher cet élément sur notre page HTML finale. En admettant que tout notre code (y compris les librairies `react` et `react-dom`) se trouve dans un fichier nommé `script.js` au même niveau qu'un fichier HTML, la figure 2.8 montre le code de la page nécessaire. Il suffit d'appeler la fonction `render` de `ReactDOM` en précisant en paramètres quel est l'élément principal à afficher, et où l'afficher sur la page.

Maintenant que nous connaissons la logique principale de React et des Composants, nous pourrons nous intéresser à comment les composer intelligemment, et à utiliser des Composants autres que les nôtres.

## Material-UI

Material-UI[6] est un ensemble de Composants React qui utilisent et respectent les guidelines de Google concernant Material Design[9]. L'utiliser va nous permettre de respecter facilement les guidelines de Material Design. Le but est de rendre le design la page cohérent dans son ensemble.

## Flux

Flux[12] est un outil très souvent utilisé avec React, car son modèle le complète bien. Flux n'est pas vraiment un framework, mais plutôt un design pattern. Flux définit une architecture d'application à suivre dans le code entier pour en tirer ses bénéfices, et propose quelques Classes pour implémenter son architecture proposée de manière rapide et facile.

Contrairement au modèle MVC très souvent utilisé, Flux repose sur l'utilisation d'un flux de données unidirectionnel.

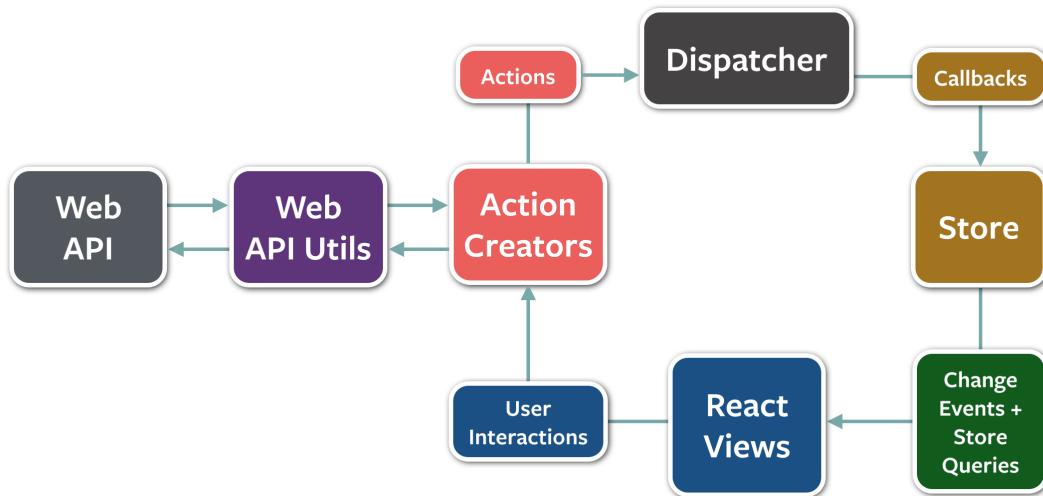


FIGURE 2.9 – L'architecture que propose Flux est unidirectionnelle.[13]

La figure 2.9 montre un schéma de l'architecture que propose Flux. En plus des Composants React décrits précédemment, Flux se décompose en trois éléments principaux :

**Stores** Un Store est un conteneur d'informations. Chaque Store va typiquement contenir un type d'information de la page web, et celui-ci devra être maintenu à jour. Il peut contenir par exemple le texte d'une conversation actuellement affichée. Les données contenues dans les stores vont souvent servir à être affichées sur la page web elle-même.

**Actions** Une Action est ce qui est généré principalement par les choix de l'utilisateur. Il s'agit par exemple d'un clic sur un bouton, ou de la saisie d'un texte dans un champ.

**Dispatcher** Le Dispatcher s'occupe de rassembler les Actions produites par les différents éléments de la page, et de les traiter afin d'envoyer des mises à jour aux Stores de la page pour que ceux-ci restent à jour.

## Backend

Plusieurs technologies seront utilisées du côté backend afin que l'application ne manque d'aucun composant.

**NodeJS** Node.js[3] permet d'exécuter des applications écrites en JavaScript en dehors du navigateur, sur une machine locale. Il sera utilisé ici en tant que serveur web pour répondre aux requêtes HTTP, et pour communiquer avec la base de données.

**Express** Express[7] est une librairie JavaScript permettant d'écrire rapidement et facilement un serveur web.

**MySQL** MySQL[8] est un système de gestion de bases de données. Son but est de conserver efficacement et de servir un grand nombre de données.

# Chapitre 3

## Prototype 1

### 3.1 Fonctionnalités

La première itération de l'application vise à couvrir certaines des exigences fonctionnelles demandées :

- Fenêtre avec liens dirigeant vers les pages
- Page "Cases" affichant la liste des cas avec possibilité d'afficher/cacher des colonnes, et de filtrer des lignes
- Page "Patients" affichant la liste des patients avec possibilité d'afficher/cacher des colonnes, et de filtrer des lignes

### 3.2 Conception

Le premier prototype de l'application a défini beaucoup des comportements et des mécaniques d'interaction de l'application.

#### 3.2.1 Architecture générale de l'application

Tout d'abord, il était nécessaire de réfléchir à l'architecture générale de l'application avant de s'attaquer à chaque partie. Au vu du produit demandé, l'application sera réalisée suivra une architecture classique du schéma Client - Serveur. Du code se trouvera donc exécuté sur le client, et sur le serveur.

##### Frontend

La partie frontend est développée à l'aide de React. Il s'agit d'une one-page app : Cela signifie que l'utilisateur utilisera toujours la même adresse web pour

accéder à l'application : Celle-ci se chargera ensuite elle-même de se transformer en les diverses interfaces que l'utilisateur va demander.

Cette page est développée à l'aide de React et de librairies permettant de compléter React, et la structure du code suivra une architecture adaptée à l'utilisation de Flux, dont nous avons parlé précédemment. Il s'agit de l'architecture qu'il est recommandé d'utiliser avec React pour une application de taille moyenne, ce qui correspond à notre projet.

## Backend

La partie backend se compose d'une couche applicative ainsi que d'une base de données. Le client ayant déjà créé une base de données de type MySQL avant le début du projet, elle ne sera pas changée.

La partie applicative du backend est développée en node.js. Ce choix s'est fait pour des raisons de simplicité : Etant donné que la partie frontend du projet est développée à l'aide d'outils adaptés pour JavaScript, il est assez naturel de continuer le développement du backend avec des technologies similaires.

La partie base de données restera donc avec MySQL. Quelques changements à la structure de la base de données seront apportés afin que celle-ci soit totalement compatible avec l'application demandée.

### 3.2.2 Structure de la navigation

En plus de répondre aux demandes du client, le but est de concevoir une application qui soit facile et agréable d'utilisation. Il a donc été nécessaire de réfléchir à la structure de la navigation de l'application en elle-même, ainsi qu'aux interactions qu'elle propose à l'utilisateur.

L'analyse des différentes parties de l'application a mené à une compréhension des différentes pages à mettre en place afin de répondre aux besoins des différents clients.

**Cases** La page *Cases* va présenter une vue des différents Cas. Un cas est lié à une épaule d'un patient, et à un ou plusieurs scans. Cette page va présenter dans un tableau la liste de tous les cas. Cette liste pourra être ordonnée et triée selon plusieurs critères. Il s'agira de la page que va principalement utiliser de public de la catégorie "scientifique". Un clic sur un cas ouvrira une page avec toutes les informations détaillées sur ce cas, comprenant entre autres des visualisations de scans.

**Patients** La page *Patients* va présenter une vue de tous les patients. Cette page sera utilisée principalement par le public de la catégorie "clinique". Elle va présen-

ter une liste complète des patients. Cette liste pourra également être ordonnée et filtrée selon plusieurs critères. Un clic sur un patient ouvrira une page montrant de manière détaillée les informations du patient, ainsi que l'ensemble des cas qui lui appartiennent. De là, il sera possible de naviguer sur un cas en particulier.

**Studies** Cette page va montrer un ensemble d'études faites sur les données de la base.

**Stats** Cette page va montrer des statistiques sur les données médicales des patients.

**Informations** Cette page va présenter de manière statique des informations sur le projet d'une manière générale.

**Admin** Cette page offrira un panel d'administration de l'outil. Les fonctionnalités de celui-ci ne sont pas encore totalement définies.

La figure 3.1 montre l'ensemble des pages ainsi que leur hiérarchie.

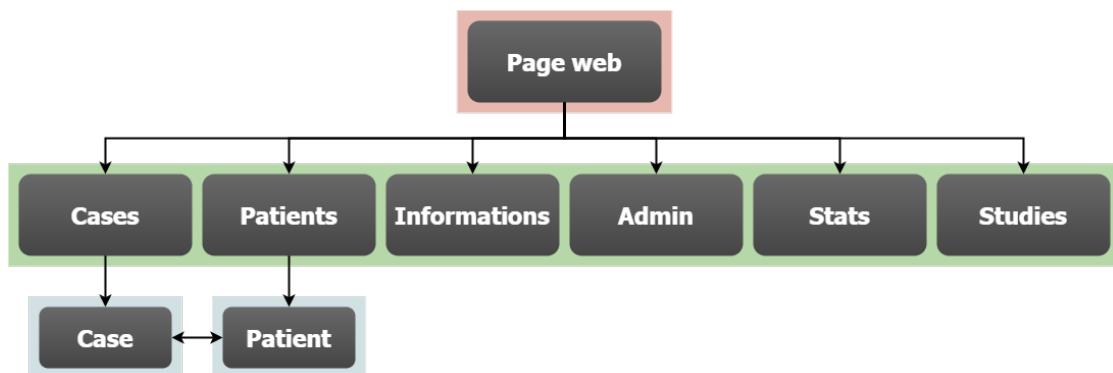


FIGURE 3.1 – La structure de la hiérarchie et navigation entre les vues

### 3.2.3 Maquettes

Des maquettes de l'interface ont été réalisées dès le début du projet. Celles-ci ne sont typiquement pas totalement fidèles à ce que ressemblera la solution finale ; le but est de montrer la structure de la navigation, ainsi que se donner les moyens de les améliorer sur le temps et de s'assurer que l'on comprenne les features demandées. La figure 3.2 montre par exemple la toute première ébauche de la page des cas, et la figure 3.3 montre la même version de la vue d'un cas particulier.

La figure 3.4 montre un sketch de deux pages secondaires au projet : une page d'administration ainsi qu'une page de statistiques.

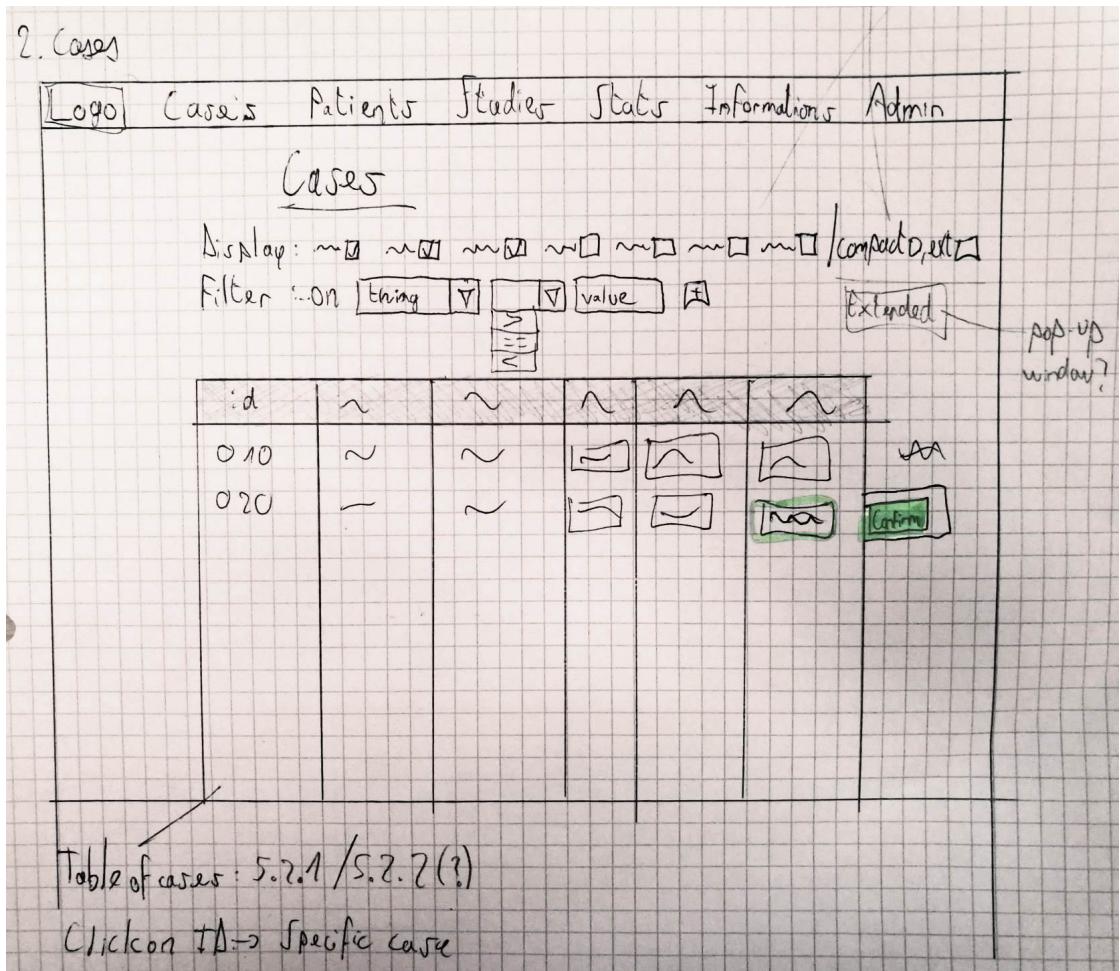


FIGURE 3.2 – Wireframe sur papier de l’interface de la liste des cas

### 3.3 Implémentation frontend

L’implémentation du frontend a commencé par le choix et l’installation de l’environnement et des outils nécessaires pour développer en React de manière efficace. Le chapitre 2.2.2 explique l’utilité de chacun des composants, mais nous allons nous simplifier le travail en utilisant une configuration déjà prête pour le développement en React. Celle-ci est tirée du code annexe à la série de tutoriels suivis. Le code de cette configuration est disponible sur GitHub [15], et il s’agit de la configuration que j’ai utilisée. Tout le code décrit plus tard repose donc sur cette configuration.

Le premier prototype s’est concentré sur l’implémentation des aspects qui ont été estimés comme étant centraux à l’application. Voici une liste des fonctionnalités du côté frontend qui doivent être implémentées pour réaliser le prototype :

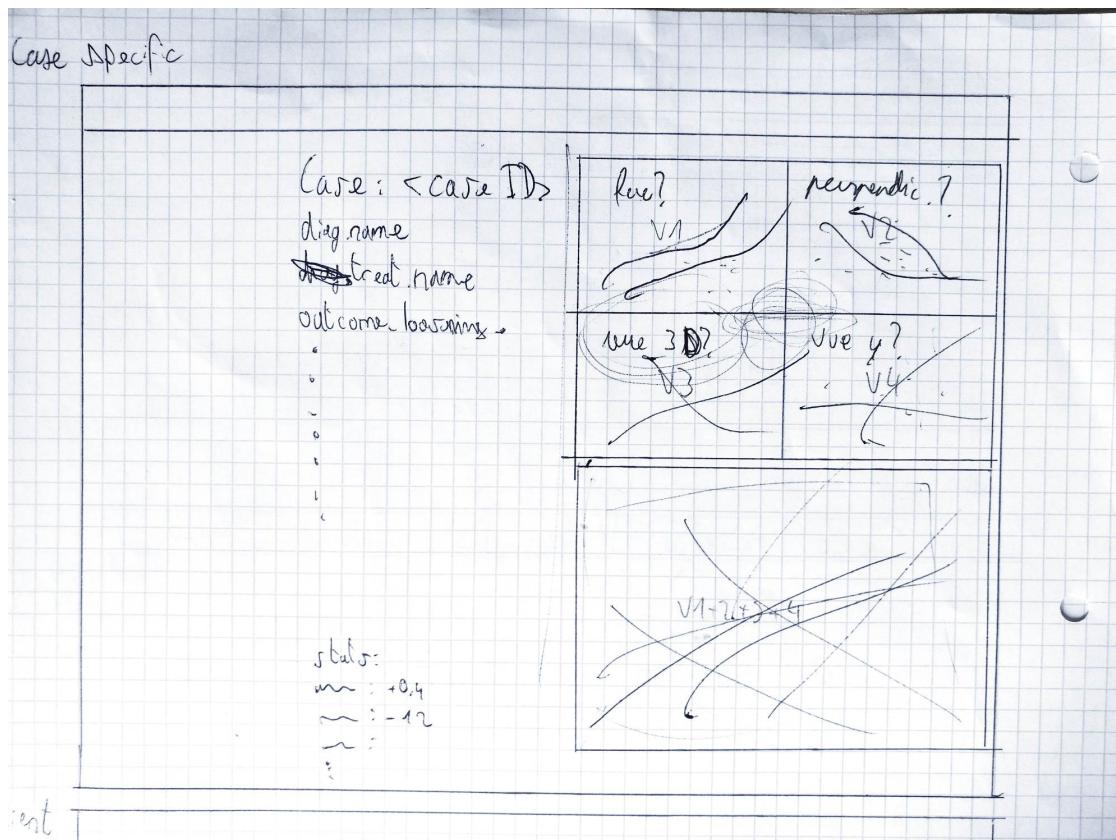


FIGURE 3.3 – Maquette sur papier de l'interface d'un cas particulier

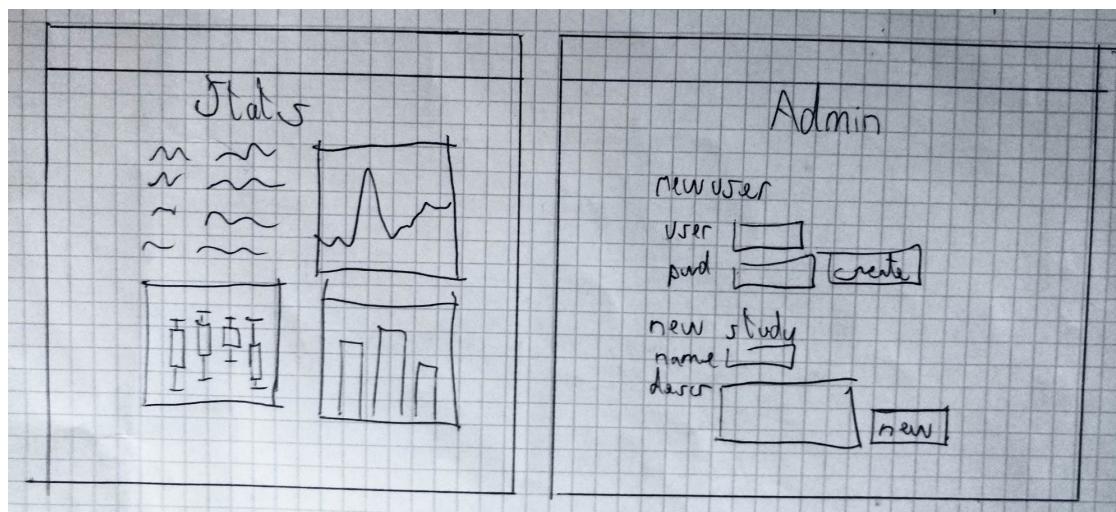


FIGURE 3.4 – Sketch sur papier des pages "Stats" et "Admin"

- Disposition générale des éléments de la page
- Routage de chaque page sur sa vue correspondante
- Implémentation des pages "Patients" et "Cases"
- Lien avec les données de la base de données

### 3.3.1 Disposition générale des éléments sur la page

Avant de démarrer à organiser les éléments d'une page, il est d'abord essentiel d'organiser les informations qui sont communes à toutes les pages. Comme le montre la figure 3.2, il est prévu de naviguer entre les pages via une barre contenant les liens des autres vues, et qui se situe en haut de la page actuelle.

Toutefois, la décision a été prise de respecter au maximum les guidelines de Material Design[9]. Ici, il est préconisé d'utiliser une barre vertical contenant les liens vers les vues, appelée "**Drawer**". Ce Composant est d'ailleurs disponible dans la librairie Material-UI, mais il n'a pas pu être utilisé directement car il ne permet pas d'être continuellement visible. Dans ce cas ainsi que comme dans plusieurs cas dans le reste de l'application, il est nécessaire de l'implémenter en utilisant uniquement le CSS fourni par Material Design.

Les liens vont donc rester sur la gauche de la page, et la vue principale sera affichée dans la partie droite.

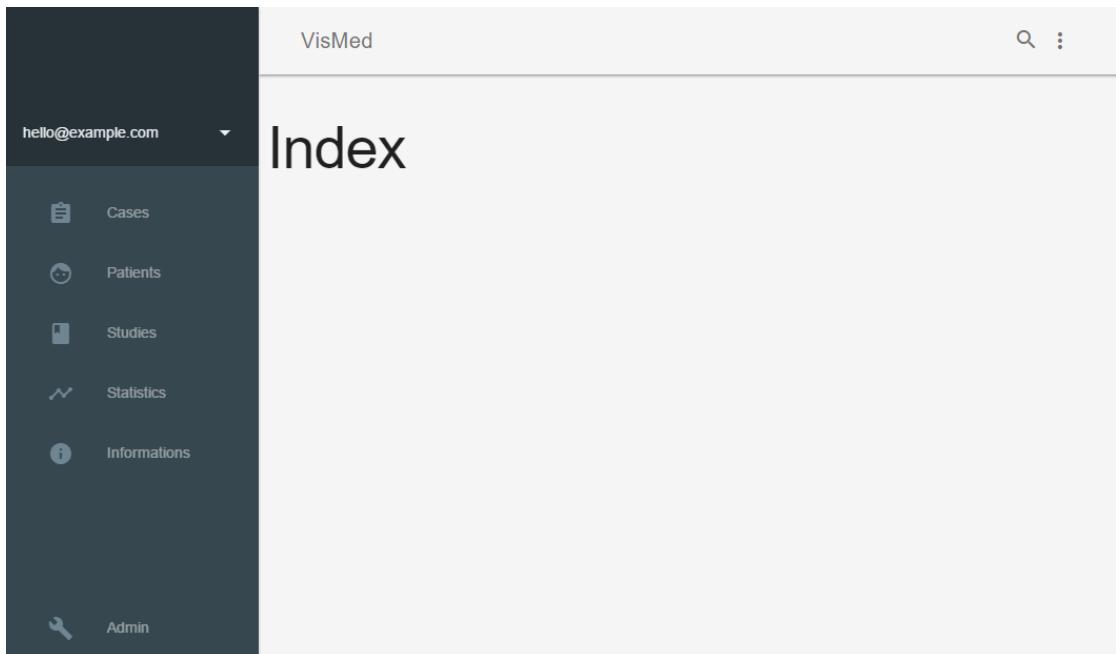


FIGURE 3.5 – Premier prototype fonctionnel de l'application

La figure 3.5 montre une des premières versions de la page, avec les liens du menu à gauche de celle-ci.

### 3.3.2 Routage

Etant donné que le but est de réaliser une "one-page app", nous allons utiliser une méthode de routage bien intégrée à React. Plutôt que télécharger une vue depuis le serveur à chaque fois que l'on désire changer de page, l'application entière est téléchargée au lancement de la page, et le changement de vue ainsi que le routage des liens se fait du côté client.

Cet effet est accompli à l'aide de la librairie React-Router[17]. Elle permet de déclarer quel Composant doit s'afficher en fonction de l'URL demandée, et de naviguer entre les vues sans rafraîchir la page complète.

```

1 <Router>
2   <MuiThemeProvider muitheme={muiTheme}>
3     <div class="mdl-layout_container">
4       <div class="demo-layout mdl-layout mdl-js-layout mdl-layout--fixed-
5         drawer mdl-layout--fixed-header">
6         <Header/>
7         <Aside/>
8         <main class="mdl-layout__content mdl-color--grey-100">
9           <div class="mdl-grid demo-content">
10             <Route path="/" exact={true} name="index" component={Index}></
11               Route>
12             <Route path="/cases" name="cases" component={Cases}></Route>
13             <Route path="/patients" name="patients" component={Patients}></
14               Route>
15             <Route path="/studies" name="studies" component={Studies}></Route>
16             >
17             <Route path="/statistics" name="statistics" component={Statistics
18               }></Route>
19             <Route path="/informations" name="informations" component={
20               Informations}></Route>
21             <Route path="/admin" name="admin" component={Admin}></Route>
22           </div>
23         </main>
24       </div>
25     </div>
26   </MuiThemeProvider>
27 </Router>
```

FIGURE 3.6 – Utilisation des composants **Router** et **Route**

La figure 3.6 montre comment est déclaré le Router, ainsi que les différentes Routes de l'application. Le composant **Router** se trouve tout en haut de la hiérarchie, et c'est à l'intérieur de lui que les autres composants vont se placer. On peut voir qu'il n'y a aucun problème à mélanger du HTML (comme aux lignes 3 et 4)

avec des composants React. Ce morceau de code représente à peu de choses près tout ce qui sera affiché sur les pages de l'application. Voici une liste des composants qu'on y trouve, et leur utilité :

**Router** Englobe tous les composants devant pouvoir s'afficher ou se cacher selon l'URL demandée. Il doit donc englober toutes les **Routes** de l'application.

**MuiThemeProvider** Nécessaire pour l'utilisation de la librairie Material-UI. Doit englober tous les éléments désirant utiliser des composants de cette librairie.

**Header** Barre horizontale se trouvant au haut de la page. Affiche le titre de la page actuellement visitée.

**Aside** Barre à gauche de la page, contenant les liens vers les différentes vues.

**Route** Définit une route associée à une URL particulière, et quel composant afficher dans le cas où l'URL correspond à cette Route.

On voit donc ainsi que chaque Route va permettre d'afficher une page particulière. Par exemple, la ligne 10 traite le cas où l'URL se termine par `/cases`, et va donc se charger d'afficher le composant **Cases** à cet endroit. Chaque composant référencé par une **Route** (**Index**, **Cases**, **Patients**, **Studies** etc...) affiche donc sa vue lorsqu'il est appelé par le Router.

### 3.3.3 Pages "Patients" et "Cases"

Dans ce prototypes, les pages "Patients" et "Cases" sont très similaires. La figure montre la vue de la page "Patients".

On remarque que celle-ci reprend au mieux les éléments dessinés sur la maquette de la figure 3.2. À nouveau, certains éléments ont été implémentés à l'aide de Material-UI, mais certains autres ont également dû être implémentés en utilisant directement le CSS de Material Design.

Le principal élément de la page est un tableau présentant les données résumées de l'ensemble des patients. Ici, il s'agit de données d'exemple.

Le but est ici de permettre deux actions différentes sur la liste des patients affichés : La partie "Display" permet d'afficher ou de cacher certaines colonnes, et la partie "Filter" permet. À noter que bien que l'interface soit implémentée ici, toutes les parties visibles ne sont pas encore fonctionnelles à ce stade. Les données affichées ne sont des données d'exemple, et les inputs de l'utilisateur (sur les parties Display et Filter) sont sans effet.

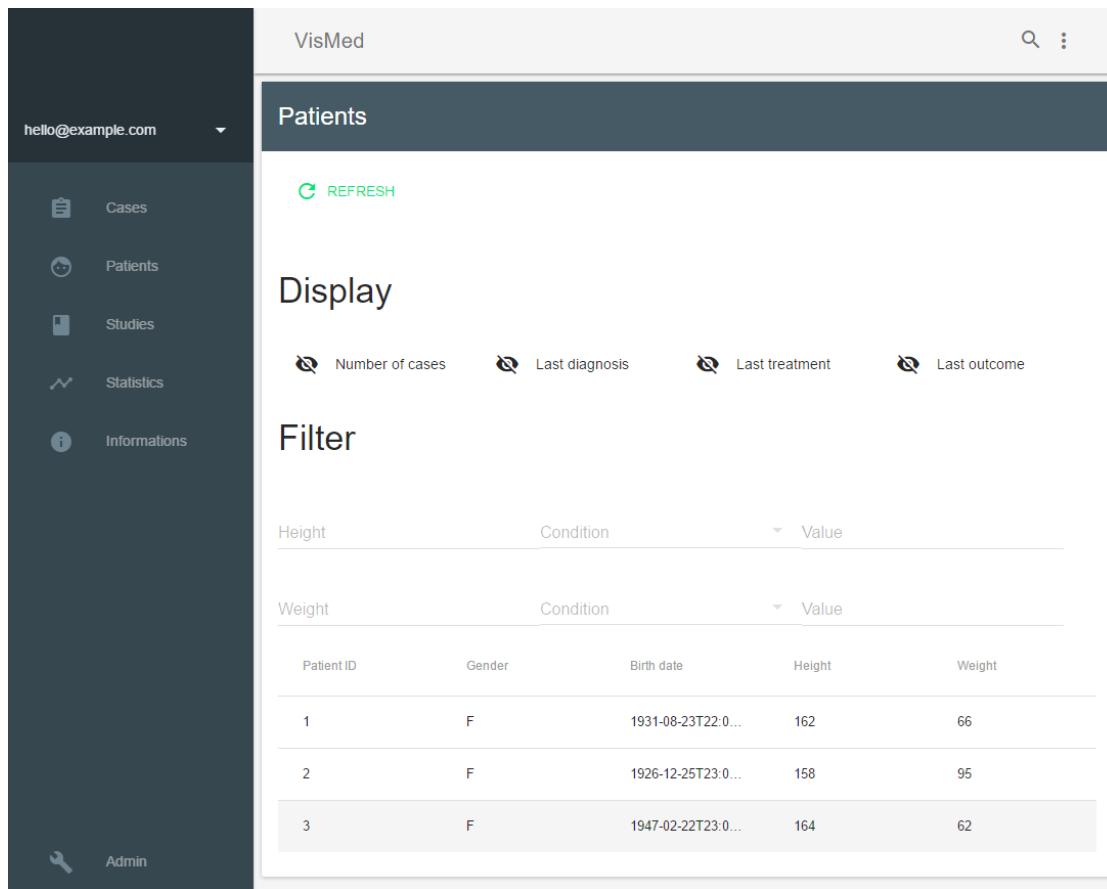


FIGURE 3.7 – Premier prototype de la page "Patients"

### 3.3.4 Communication des données

Une fois que le frontend est capable d'afficher des informations, il est encore nécessaire de le faire communiquer avec le serveur afin d'afficher les vraies données.

#### Flux de données

Comme nous utilisons et suivons le modèle Flux, il est nécessaire d'établir la logique du flux des données. Comme décrit dans le chapitre Analyse, Flux recommande l'utilisation d'un flux unidirectionnel des données.

En suivant leurs recommandations, il a été possible d'établir le schéma illustré à la figure 3.8, montrant la manière dont les données sont échangées à travers de l'application React. On y voit les différents Classes et Composants qui interagissent dans l'échange de données :

**CaseActions et PatientActions** Classes qui vont communiquer avec l'API

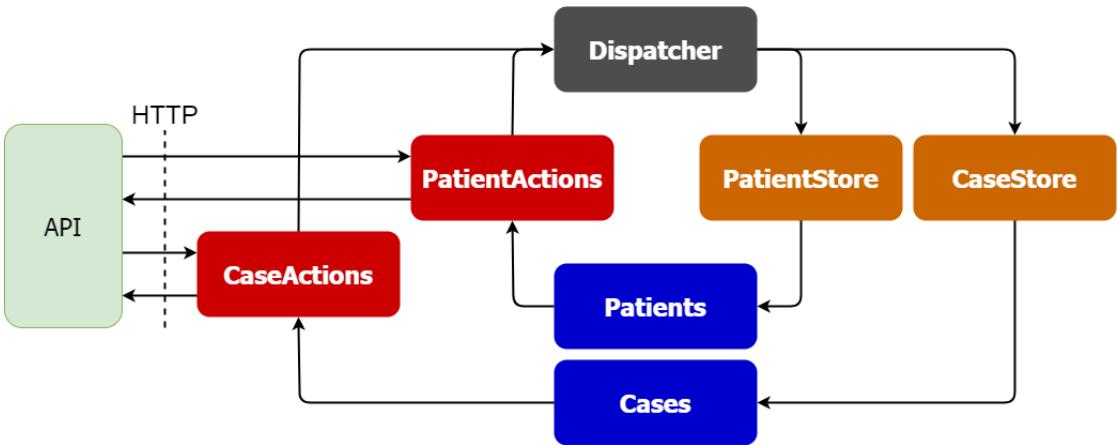


FIGURE 3.8 – Flux de données de l’application

du serveur lorsqu’on leur en fait la demande. Vont demander les données, et les transmettre au **Dispatcher** une fois reçues.

**Dispatcher** Classe provenant de Flux, qui s’occupe de transmettre les données reçues aux Stores qui les stockent.

**PatientStore** et **CaseStore** Classes qui stockent les données reçues depuis le **Dispatcher**. Elles servent à alimenter les vues **Patients** et **Cases** en données lorsqu’elles en ont besoin.

**Patients** et **Cases** Composants React qui affichent respectivement la liste des patients, et la liste des cas. Chaque composant s’attache à son Store respectif, et affiche les données de celui-ci.

Au lancement de l’application, **PatientActions** et **CaseActions** agissent en demandant au serveur les données une première fois, ce qui va déclencher le reste de la boucle de réaction et ainsi pouvoir fournir des données aux composants **Patients** et **Cases** lorsque ceux-ci doivent être affichés.

## 3.4 Implémentation backend

### 3.4.1 Structure du backend

Pour la réalisation de la connexion entre la partie applicative métier et la partie base de données, il a été nécessaire de s’adapter à la structure hiérarchique des machines hébergeant l’application à l’EPFL. En effet, une contrainte est que le serveur hébergeant la base de données MySQL est accessible uniquement depuis l’intérieur du réseau.

Bien que cela ne soit pas un problème pour l'application finale qui tournera sur un serveur de l'EPFL, cela complique la phase de développement durant laquelle il est nécessaire d'utiliser un autre serveur de l'EPFL comme proxy pour s'y connecter.

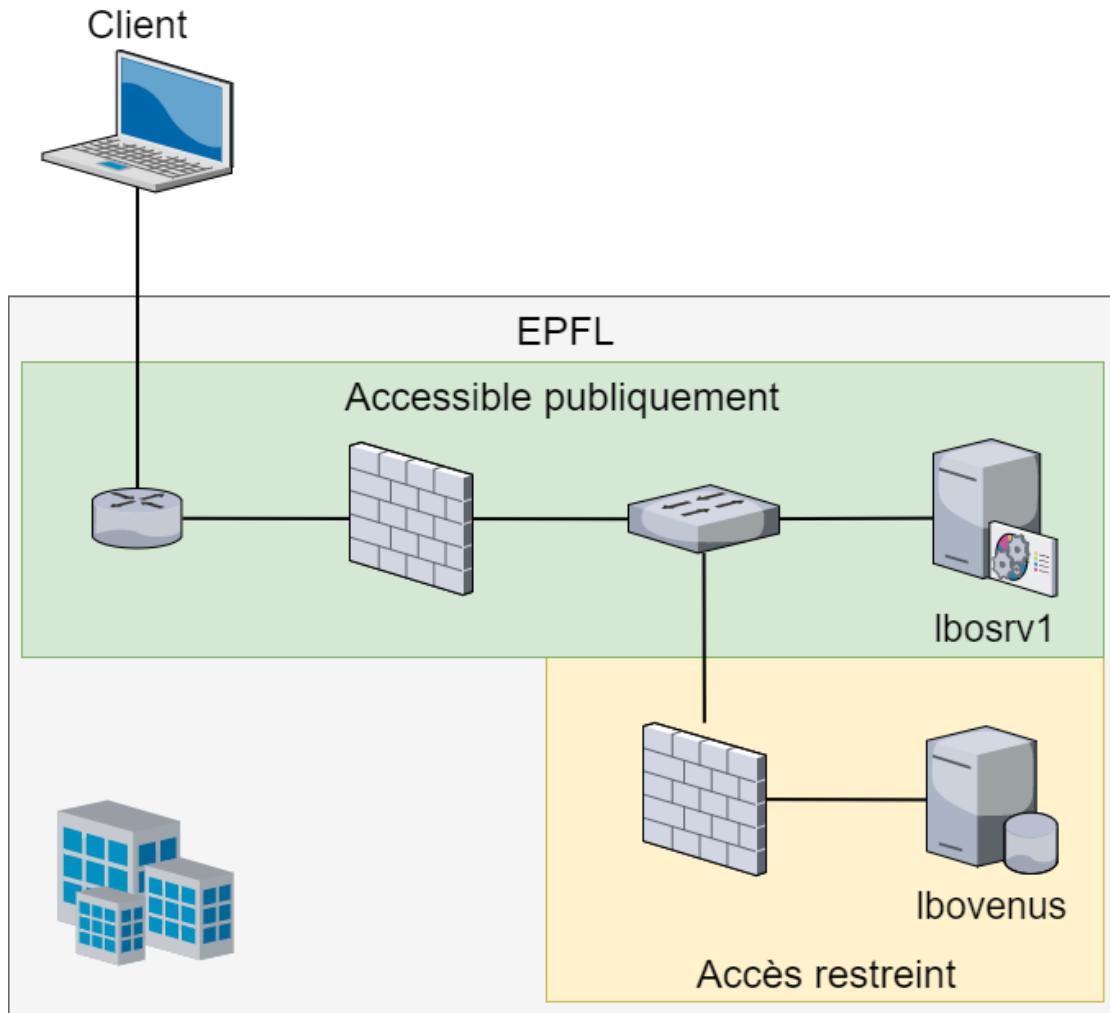


FIGURE 3.9 – Schéma de l'architecture réseau de l'EPFL utilisée par VisMed

La figure 3.9 montre, parmi l'architecture de l'EPFL traversée, les deux serveurs utilisés par l'installation de VisMed. On y voit :

**lbosrv1** Il s'agit du serveur "principal", sur lequel tourne le code backend de l'application. Il s'agit également du serveur qui va écouter le port 80, et donc recevoir les requêtes HTTP de clients externes et y répondre. Mais afin de pouvoir faire son travail correctement, il aura généralement besoin d'accéder à des données se trouvant sur la base de données des cas et des

patients. Celle-ci se trouve sur le deuxième serveur utilisé, qui se nomme **lbovenus**.

**lbovenus** Il s'agit du serveur contenant la base de données médicales actuelle. Celle-ci, de type MySQL, contient les données concernant les patients de l'hôpital, ainsi que leurs cas. Cette base de données se trouve derrière un pare-feu supplémentaire, et est inaccessible depuis l'extérieur de l'EPFL. Le seul moyen de pouvoir la questionner est d'être à l'intérieur du réseau de l'EPFL, où se trouve également **lbosrv1**. C'est pourquoi il y a accès sans problème.

### 3.4.2 Express.js

La logique du serveur a été développée en utilisant node.js ainsi que la librairie Express. Express s'occupe d'écouter sur un port particulier et de répondre aux requêtes avec les informations adéquates. Dans notre cas, le port utilisé est le **3000**. Elle sert les quelques ressources statiques nécessaires (script JavaScript dont l'application compilée, mais également les fichiers CSS). Mais également, elle répond aux requêtes sur l'API définie.

Les requêtes possibles de l'API sont peu nombreuses :

**GET /api/patients** Renvoie la liste complète des patients ainsi que toutes leurs informations.

**GET /api/cases** Renvoie la liste complète des patients ainsi que toutes leurs informations.

Pour fournir les informations nécessaires lors de l'appel à l'un de ces deux méthodes, l'application se connecte à la base de données, effectue une requête sur la vue correspondante, et formate le résultat pour l'envoyer en JSON au client.

### 3.4.3 Base de données

Pour simplifier la sélection des données utiles à partir de la structure initiale de la base de données, des vues SQL ont été ajoutées. Les deux vues, **case\_view\_all** et **patient\_view\_all**, joignent toutes les données nécessaires de plusieurs tables afin de constituer la réponse voulue aux requêtes sur les deux méthodes de l'API précédemment décrites.

## 3.5 Evaluation

### 3.5.1 Heuristique et adaptations

Ce prototype a été discuté à de nombreuses reprises. Des évaluations ont été faites de semaine en semaine, et ont mené à de nombreuses améliorations basées sur des critères ergonomiques.

Durant ces séances d'évaluation heuristique, beaucoup de points discutés ont été à adapter afin de proposer une interface avec une meilleure ergonomie. Les modifications qui en résultent suivent les critères ergonomiques de Bastien et Scapin[18] tels que la cohérence, la lisibilité ou le principe du groupement/distinction. Par exemple, sur la page Patients, on peut noter :

- Augmentation de la taille de police
- Retrait des titres "Display" et "Filter"
- Elargissement de la zone d'affichage à toute la largeur de la page
- Utilisation du Header pour y placer les filtres de display
- Retrait du bouton d'actualisation
- Augmentation de la visibilité des titres de colonne

De même, des changements plus généraux ont été appliqués :

- Augmentation de la taille de police des liens
- Augmentation du contraste de la police et des icônes des liens
- Retrait du titre "VisMed" dans le Header
- Passage du bouton "Admin" en haut du Drawer, avec les autres liens
- Surbrillance du lien actuellement sélectionné

Les évaluations heuristiques ont donc permis au prototype d'évoluer à la fois en termes de fonctionnalités grâce au contact avec le client, et à la fois utilisabilité grâce aux critères d'ergonomie qui ont été appliqués. Les adaptations se sont fait itérativement sur plusieurs semaines, et ont mené à la production d'un deuxième prototype.

### 3.5.2 Validation par les clients

Les prototypes ont été montrés au client à plusieurs reprises afin de bénéficier d'un feedback, en plus de la première réunion en personne avec tous les intervenants. La confirmation de celui-ci de la direction du développement nous a permis de s'assurer que le projet allait dans le sens demandé.

# Chapitre 4

## Prototype 2

### 4.1 Fonctionnalités

La deuxième itération de l'application vise à couvrir certaines autres fonctionnalités :

- Recherche d'informations dans la page "Cases" et "Patients"
- Page d'édition d'un "Case"
- Page d'édition d'un "Patient"

### 4.2 Conception

Le deuxième prototype de l'application comprend donc des nouveautés en termes de fonctionnalités. En effet, après avoir déployé et présenté le premier prototypes, des remarques de la part du client ont centré le développement sur les deux fonctionnalités suivantes les plus intéressantes ; la recherche d'informations, et l'édition de données. La fonctionnalité principale est l'édition des données, et il fallait pouvoir éditer la plupart des données affichées : Aussi bien celles d'un cas que celles d'un patient.

### 4.3 Pages d'édition

Deux nouvelles pages sont donc consacrées à afficher et permettre la modification d'une part d'un cas, et d'autre part d'un patient. Ces pages sont accessibles en cliquant respectivement sur la ligne d'un, ou sur la ligne d'un patient dans la liste. Cette page affiche toutes les informations relatives au cas ou au patient concerné, mais chacune a une spécificité supplémentaire :

- La page "Case" affiche également les informations du Patient concerné, ainsi qu'un bouton "Edit" redirigeant l'utilisateur vers la page d'édition du Patient.
  - La page "Patient" affiche l'ensemble des cas qui concernent ce patient. Cliquer sur un cas dirige l'utilisateur vers la page d'édition de celui-ci.
- La figure 4.1 montre un prototype de la page de modification d'un cas.

Champ	Valeur
sCase_id	276
shoulder_id	276
folder_name	P65
last_update	2017-02-07T13:42:32.000Z

FIGURE 4.1 – Prototype de la page de modification d'un cas

## 4.4 Recherche

Une autre fonctionnalité a été modifiée : Le filtrage des données. Le système initial de filtres a été changé en un champ de recherche, qui est plus facile et plus rapide à utiliser. Ce filtre sera présent aussi bien sur la page Cases que sur la page Patients. Ecrire dans le champ recherche la chaîne de caractères entrée dans l'ensemble des champs des données affichées. La figure 4.2 montre un prototype de la page de la liste des cas.

## 4.5 Implémentation

### 4.5.1 Pages d'édition

L'implémentation des pages d'édition d'un cas et d'un patient s'est naturellement faite pas l'addition de deux nouveaux composants **Case** et **Patient**.

Cases	Folder	Diagnosis	Treatment	Outcome	grafting
	ID	Folder	Diagnosis	Treatment	Outcome
395	P197	OA	rTSA : glenoid bone grafting		
431	P234	OA	aTSA : LCB Tenodesis+Glenoid bone grafting...	N : NaN	
434	P237	CTA	rTSA : greffe glène+transfert dorsal+rond=gle...		
436	P239	OA	rTSA : glenoid bone grafting		
438	P241	OA : Glenoid dysplasia	rTSA : glenoid bone grafting		
439	P242	OA : Glenoid dysplasia	rTSA : glenoid bone grafting		
449	P253	OA : Dysplasia	rTSA : glenoid bone grafting		
462	P266	OA : "+sublux post statique+usure glénoidien..."	aTSA : LCB Tenodesis+Glenoid bone grafting...		
486	P291	OA	rTSA : glenoid bone grafting		
503	P319	OA	aTSA : LCB Tenodesis+Glenoid bone grafting...		
509	P325	OA	rTSA : glenoid bone grafting		

FIGURE 4.2 – Prototype de la liste des cas avec recherche de mot clé

## Case

Les données à afficher sur la page de Cas sont plus larges que les données affichées sur la liste des cas : En effet, il est nécessaire ici de montrer la liste des CT scans liés à un cas. Cet ajout a nécessité des modifications dans le backend : L'ajout d'un endpoint destiné à renvoyer non seulement toutes les informations d'un cas, mais également l'ensemble des informations des CT liés à ce cas.

La méthode suivante a donc été ajoutée au serveur :

**GET /api/case/<id>** Renvoie les informations du cas <id> ainsi que les CT qui sont liées

Celle-ci fait appel à une vue SQL implémentée sous le nom de `ct_view`.

De plus, la possibilité de modifier les données du cas a également nécessité tout d'abord l'ajout d'une méthode sur le serveur :

**POST /api/case/<id>** Modifie les informations du cas <id>

Cette méthode fait appel à une procédure stockée SQL implémentée pour cette fonctionnalité. Nommée `updateCase`, elle prend en paramètre l'ensemble des données modifiables d'un cas. Si un paramètre est laissé vide (C'est le cas lorsqu'il n'est pas modifié par l'utilisateur dans l'interface), il est ignoré et n'est pas mis à jour. Ceci a été fait pour éviter le cas où plusieurs modifications concurrentes pourraient écraser des nouvelles données avec des données non mises à jour.

Afin de repérer facilement quelles données vont être mises à jour, les champs modifiés dans l'interface sont également mis en surbrillance avec une couleur bleue. La figure montre un champ "Commentaire" modifié, qui se distingue des autres champs de sa catégorie.

The screenshot shows a 'Diagnosis' form with three input fields:

- Name:** normal
- Comment:** Test
- CMI:** None

The 'Comment' field is highlighted in blue, indicating it has been modified.

FIGURE 4.3 – Mise en évidence du champ modifié

## Patient

L’ajout de la page de modification d’un Patient a demandé des efforts semblables à ceux de la page Case. On retrouve donc l’ajout de deux nouvelles méthodes à l’API, cette fois-ci pour un patient :

**GET /api/patient/<id>** Renvoie les informations du patient <id> ainsi que des cas qui lui sont liées

**POST /api/patient/<id>** Modifie les informations du patient <id>

Une des différences est que la méthode **GET /api/patient/<id>** renvoie également la liste des cas appartenant à un patient.

De même, une procédure stockée SQL a également été implémentée pour permettre la modification facile des données d’un Patient, sous le nom de **updatePatient**.

### 4.5.2 Recherche

L’implémentation du champ de recherche ainsi que les boutons d’affichage des colonnes dans le **Header** a nécessité la création de composants spécialisés pour gérer ce type de Header, en dehors des Headers classiques. Les composants **CasesHeader** et **PatientsHeader** ont été créés, et affichent donc les contrôles nécessaires pour afficher ou cacher des colonnes, ainsi que gérer la recherche.

Ces deux composants doivent communiquer avec les composants **Cases** et **Patients** de base. Il a donc été nécessaire de faire passer cette communication par le flux d’informations installé jusqu’ici.

Lorsque l’utilisateur modifie l’affichage de colonnes ou la recherche appliquée, une action est donc créée et dispatchée sur le Store correspondant (soit CaseStore, soit PatientStore). La vue de la liste est ensuite mise à jour à partir des informations du store. Cependant, contrairement à précédemment, toutes ces opérations se sont du côté client, et aucune communication avec le serveur n’est requise.

## 4.6 Evaluation heuristique et adaptations

Au cours du développement de cette application, plusieurs évaluations formatives ont été réalisées. Chaque évaluation se focalise sur une ou plusieurs parties de l'application différentes, mais le but est généralement commun : On souhaite faire le point sur l'état actuel de l'utilisabilité de l'application en prenant un point de vue utilisateur, et en se posant des questions sur le design actuel de l'interface.

Bien que fonctionnelle, ces vues n'étaient pas optimales sur le plan ergonomique. Une évaluation formative a montré plusieurs points à changer :

- La première information affichée sur la page Cas concerne le Patient : Cela est inadéquat
- Le bouton "Save" ne devrait concerner que la partie de la page effectivement modifiable
- Les champs d'input contiennent un label trop peu visible : Il est nécessaire d'améliorer leur visibilité
- Le champ de date est présent dans la base de données, mais pas sur la page : Il faut l'ajouter

Ces dernières remarques ont donc été prises en compte afin de réaliser la dernière version de l'interface de l'application.

## 4.7 Résultat

L'application est fonctionnelle, et propose les éléments suivants :

Les figures 4.4, 4.5 montrent respectivement la vue de la page de l'ensemble des Cas, ainsi que la vue de la page d'un Cas spécifique.

Les figures 4.6, 4.7 montrent respectivement la vue de la page de l'ensemble des Patients, ainsi que la vue de la page d'un Patient spécifique.

ID	Folder	Diagnosis	Treatment	Outcome
243	P24	OA	rTSA : NaN	
437	P240	CTA	rTSA : NaN	
438	P241	OA : Glenoid dysplasia	rTSA : glenoid bone grafting	
439	P242	OA : Glenoid dysplasia	rTSA : glenoid bone grafting	
440	P243	OA	aTSA : NaN	
441	P244			
442	P245	CTA	rTSA : NaN	
443	P247	OA	aTSA : NaN	
444	P248	CTA : antero-superior subluxation	rTSA : LD+Tmajor transfert	
445	P249	OA (secondary)	aTSA : AMO vis omoplate, uncemented	

FIGURE 4.4 – Page "Cases" mise à jour, montrant la recherche de l'information 'P24'

### Edit Case N34

Folder Name N34	Diagnosis	Treatment
Name normal (sub-)	Name None	Comment None
Comment None	Date None	Score None
CMI None	ASA	Comment None
DRG None	Date None	Loosening None
Date None		Date None

**Patient** EDIT

- Right shoulder
- Female
- 164cm
- 62kg

**CT scans**

Field	Value
sCase_id	3

FIGURE 4.5 – Page finale d'édition d'un cas

ID	Person	Last Diagnosis	Last Treatment	Last Outcome
				1978
109	GS   F   170cm   75kg   1978-07-25	normal		
116	SS   M   170cm   90kg   1978-09-25	normal		
183	DD   M   170cm   70kg   1978-07-28	normal		
191	BS   M   174cm   85kg   1978-06-20	normal		
215	DF   M   178cm   68kg   1978-01-25	normal		
216	VI   M   178cm   73kg   1978-10-09	normal		
382	NN   M   ?cm   ?kg   1978-05-22	OA (secondary)	aTSA : uncemented	N : PSI

FIGURE 4.6 – Page "Patients" mise à jour, montrant la recherche de l'information '1980'

### Person

Gender  
F

Height  
158

Weight  
95

Birth Date  
1926-12-26

### Anonymity

Initials  
VM

IPP  
0

### Patient's Cases

ID	Folder	Diagnosis	Treatment	Outcome
2	N32	normal		

FIGURE 4.7 – Page finale d'édition d'un patient

# Chapitre 5

## Conclusion

### 5.1 Conclusion du projet

#### 5.1.1 Délivrables

Ce projet est passé par plusieurs étapes distinctes qui ont mené à la production de plusieurs délivrables :

- Analyse des besoins
- Analyse des technologies
- Maquette papier
- Premier prototype : Développement et évaluation
- Deuxième prototype : Développement et évaluation
- Application finale

Chacune de ces étapes nous a amené à produire une itération supplémentaire contenant à la fois des nouveautés fonctionnelles, mais aussi des changements et améliorations ergonomiques. Chaque itération nous a donc rapproché du produit final.

#### 5.1.2 Conclusion générale

Bien que le scope initial du projet soit surdimensionné, nous avons pu communiquer avec le client afin de clarifier quelles étaient les attentes principales. Les fonctionnalités centrales ont donc été implémentées, et le produit final répond donc aux besoins les plus importants du client. Ainsi, l'application finale permet de :

- Afficher des informations de l'ensemble des cas
- Rechercher des informations parmi les cas
- Afficher et modifier les données d'un cas particulier
- Afficher des informations de l'ensemble des patients
- Rechercher des informations parmi les patients

- Afficher et modifier les données d'un patient particulier

### 5.1.3 Perspectives

Le projet dans son état final contient plusieurs pages non implémentées :

- Studies
- Stats
- Admin

Ces pages sont actuellement vides et un projet futur pourrait commencer par y ajouter du contenu, après avoir défini clairement leur utilité.

De plus, l'application dans son ensemble pourrait bénéficier d'un véritable système de login, ainsi que d'une sécurité accrue, notamment dans les accès aux méthodes de l'API.

## 5.2 Conclusion personnelle

J'ai beaucoup apprécié travailler sur ce projet pour deux raisons différentes :

Tout d'abord, j'ai pu voir et gérer un processus centré sur l'utilisateur. Cette approche itérative change de ce que j'ai eu l'habitude de faire, et m'a permis de découvrir une autre manière de produire un résultat.

Et également, j'ai pu me familiariser et utiliser dans une situation pratique certaines des dernières technologies du monde du Web. Bien que toutes les librairies ne m'aient pas plu (J'ai trouvé Material-UI assez lacunaire), j'ai beaucoup aimé travailler avec React et Flux, qui ont l'air assez matures pour pouvoir produire des interfaces de qualité et maintenables.

# Bibliographie

- [1] Facebook Inc., *A JavaScript library for building user interfaces - React*, <https://facebook.github.io/react>, Consulté en ligne en Mars 2017, 2017.
- [2] ValueCoders, *5 reasons to choose Facebook's ReactJS*, <http://www.valuecoders.com/blog/technology-and-apps/5-reasons-choose-facebook-reactjs/>, Consulté en ligne en Avril 2017, 2017.
- [3] Node.js, *Node.js*, <https://nodejs.org>, Consulté en ligne en Avril 2017, 2017.
- [4] npm, *npm*, <https://npmjs.com>, Consulté en ligne en Avril 2017, 2017.
- [5] webpack, *webpack*, <https://webpack.github.io/docs/>, Consulté en ligne en Avril 2017, 2017.
- [6] Material-UI, *Material-UI*, <http://www.material-ui.com>, Consulté en ligne en Avril 2017, 2017.
- [7] Express, *Express - Infrastructure d'application Web Node.js*, <http://expressjs.com/fr/>, Consulté en ligne en Avril 2017, 2017.
- [8] MySQL, *MySQL*, <https://www.mysql.fr/>, Consulté en ligne en Avril 2017, 2017.
- [9] Material Design, *Introduction - Material Design - Material design guidelines*, <https://material.io/guidelines/>, Consulté en ligne en Avril 2017, 2017.
- [10] React, *JSX in Depth*, <https://facebook.github.io/react/docs/jsx-in-depth.html>, Consulté en ligne en Avril 2017, 2017.
- [11] React, *Tutorial : Intro to React*, <https://facebook.github.io/react/tutorial/tutorial.html>, Consulté en ligne en Avril 2017, 2017.
- [12] Facebook Inc., *Flux / Application Architecture for Building User Interfaces*, <https://facebook.github.io/flux/>, Consulté en ligne en Mai 2017, 2017.
- [13] GitHub Inc., *facebook/flux : Application Architecture for Building User Interfaces*, <https://github.com/facebook/flux>, Consulté en ligne en Mai 2017, 2017.

- [14] YouTube, *React JS Tutorials*, <https://www.youtube.com/playlist?list=PLoYCgNOIyGABj2GQSlDRjgvXtqfDxKm5b>, Consulté en ligne en Mars 2017, 2017.
- [15] GitHub Inc., *Spartano/LearnCode.academy-REACT-JS-TUTORIAL*, <https://github.com/Spartano/LearnCode.academy-REACT-JS-TUTORIAL>, Consulté en ligne en Mars 2017, 2017.
- [16] Material UI, *Drawer*, <http://www.material-ui.com/#/components/drawer>, Consulté en ligne en Mars 2017, 2017.
- [17] React Training, *React Router : Declarative Routing for React.js*, <https://reacttraining.com/react-router>, Consulté en ligne en Avril 2017, 2017.
- [18] Ergoweb, *Critères ergonomique de Bastien et Scapin*, <http://ergoweb.ca/criteres/>, Consulté en ligne en Juin 2017, 2017.

# Glossaire

**CHUV** est l'abréviation de "Centre Hospitalier Universitaire Vaudois" .. 6

**design pattern** se traduit en "Patron de conception". Décrit un arrangement caractéristiques de modules ou composants en tant que solution à un problème typique.. 17

# Remerciements

Je tiens à remercier ma superviseure Sandy Ingram pour m'avoir guidé lors des décisions à prendre, ainsi que m'avoir soutenu et aidé tout au long de ce projet.

# Déclaration d'honneur

Je, soussigné, Kewin Dousse, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

Lieu

Date

Signature

## Annexe A

### Historique des versions

Voici l'historique des versions de ce document.

- 0.1 : Template du document
- 0.2 : Rédaction de la partie Analyse
- 0.3 : Rédaction de la partie technique
- 1.0 : Rédaction de la plupart du reste du rapport
- 1.1 : Correction de la structure de chapitres dans les Prototypes

# Annexe B

## Cahier des charges

### B.1 Activités

Le développement du projet peut se découper en plusieurs phases, qui elles-mêmes se divisent en plusieurs activités. Voici la liste de ces activités :

1. Analyse des besoins
  - (a) Analyse des features demandées
  - (b) Etude des cas d'utilisation
  - (c) Etude de la faisabilité des demandes
  - (d) Analyse de la correspondance avec la base de données existante
2. Analyse des technologies
  - (a) Analyse et choix des technologies pertinentes
  - (b) Apprentissage des technologies choisies
3. Conception de l'application
  - (a) Design de l'architecture générale de l'application
  - (b) Design de l'interaction et de la structure de navigation
  - (c) Prototypage
4. Réalisation de l'application
  - (a) Réalisation du code backend
  - (b) Réalisation du code frontend
  - (c) Evaluation du comportement fonctionnel de l'application
  - (d) Evaluation de la robustesse de l'application
5. Evaluation itérative
  - (a) Evaluation formative
  - (b) Evaluation sommative informelle et empirique
  - (c) Rétrospective du développement

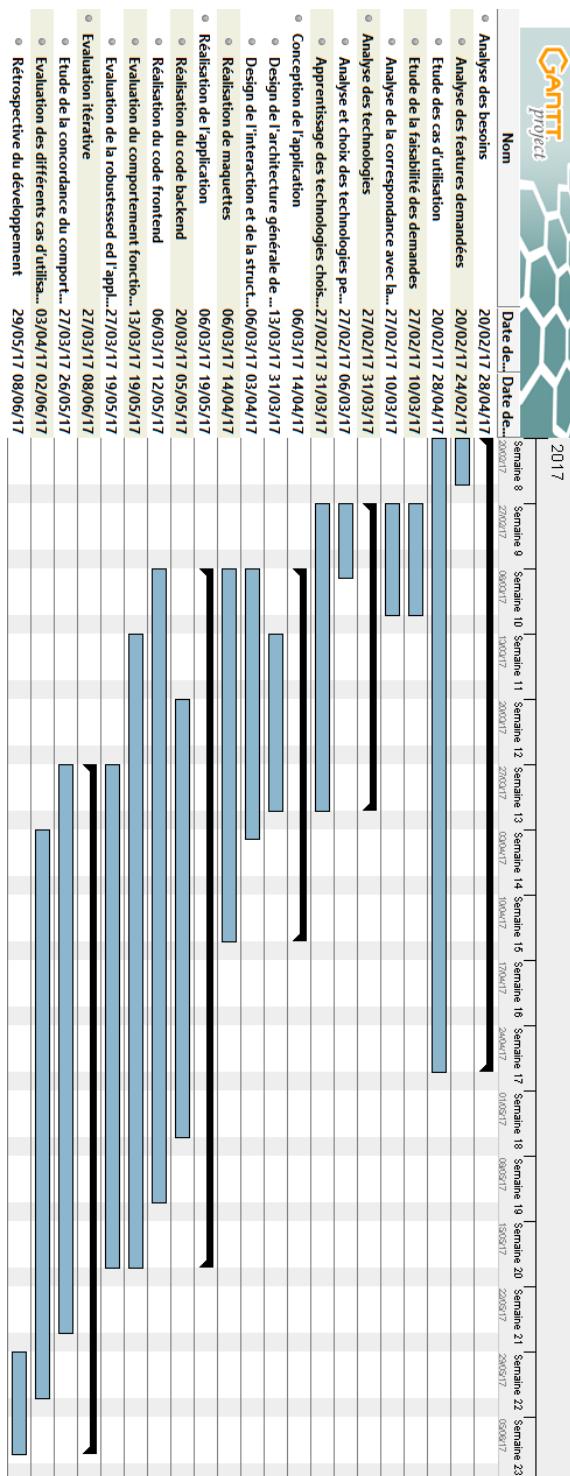
## B.2 Planification

Le projet comporte une série de dates-clé qu'il est important de respecter :

Date	Semaine	Tâche
Lundi 20 février 2017	Semaine P1	Début du projet
Lundi 31 mars 2017	Semaine P6	Fin de l'apprentissage des technologies choisies
Lundi 20 mai 2017	Semaine P12	Fin de l'implémentation de l'application
Vendredi 9 juin 2017	Semaine P15	Dépôt du rapport
19-30 juin 2017	-	Défense orale

Les dates en rouge sont des dates de rendu officielles. Les autres représentent des jalons dans l'avancement du projet.

## B.3 Diagramme de Gantt



# Annexe C

## Documentation

### C.1 Localisation

L'ensemble des documents du projet est disponible à l'adresse suivante : <https://forge.tic.heia-fr.ch/projects/vismed>

Le code à son état du 15.07.2016 se trouve sur le dépôt git à l'adresse suivante : <https://gitlab.forge.hefr.ch/kewin.dousse/VisMed>

### C.2 Contenu

#### C.2.1 Forge

Le projet présent sur la forge contient toutes les versions de chacun des documents suivants, sous l'onglet « Documents » :

- Les procès-verbaux réalisés durant le projet.

## **Annexe D**

### **Procès-verbaux**

Voici les documents des procès-verbaux réalisés.

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



## PV de réunion

20 février 2016, de 10h30 à 11h20, en D20.19

Présent : Sandy Ingram, Kewin Dousse

Rédaction du PV le 20 février

### Compte-rendu

#### Points de discussion

- Accès SSH et base de données

La réunion a débuté sur le sujet des accès aux ressources existantes, comprenant une VM ainsi qu'une copie de la base de données. Sandy a donné les accès comprenant nom d'hôte, nom d'utilisateur et mot de passe à Kewin.

- Technologies

La discussion s'est ensuite portée sur le choix des technologies pour le projet. En prenant en compte les objectifs du projet ainsi que les objectifs pédagogiques, il reste tout de même une bonne liberté de choix parmi beaucoup d'outils différents. Kewin mènera une recherche pour la semaine prochaine de la possibilité d'utilisation de plusieurs technologies/mots clés, parmi lesquels ont été mentionnés : PWA, React Native, React Ionic, Cordova, Redux, Firebase, Elastic Search.

- Schéma de la base de données

Le schéma de la base de données a été passé en revue afin de comprendre l'organisation des informations. Celui-ci semble correct et compris.

- Etapes et planning

Différentes étapes pour l'élaboration du projet ont été relevées, qui correspondent à peu près à un planning qui sera établi la semaine prochaine : Validation de la structure de la base de données ; Affichage, filtrage et tri des cas ; Visualisations des scans ; Statistiques ; Comparaisons entre population saine et population pathologique. Pour la semaine prochaine, quelques esquisses d'interfaces seront réalisées afin d'aider à la compréhension du fonctionnement voulu de l'application. Une question soulevée est également restée en suspens : Est-ce que les formalités de ce projet sont semblables à celles d'un projet de semestre ?

- Prochains rendez-vous

La réunion de la semaine prochaine a été fixée à 16h45 exceptionnellement. Les réunions suivantes auront probablement lieu à 10h30. Une date de rencontre avec le Dr. Alexandre Terrier a été proposée : Le 17 mars à 9h. À confirmer.

#### Conclusion

Le projet démarre par une recherche d'informations sur les technologies disponibles et pertinentes au but, ainsi que des questionnements à se poser en vue de bien comprendre le but et les étapes du projet.

## ANNEXE D. PROCÈS-VERBAUX

---

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



# PV de réunion

27 février 2016, de 10h30 à 11h05, en D20.19

Présent : Sandy Ingram, Kewin Dousse

Rédaction du PV le 28 février

## Compte-rendu

### Points de discussion

- **Base de données**

La réunion a débuté sur le sujet de la base de données existantes, et des champs de celle-ci. Bien que la structure semble correcte, quelques questions se sont posées sur l'utilité et le sens de certains des champs existants, d'autant plus que certaines tables sont actuellement vides, donc sans exemples. Ces questions seront formalisées et posées afin d'avoir une compréhension totale de la base de données.

- **Schémas maquettes**

Les schémas de maquettes ont été réalisés et revus. Donnant une vue générale de l'application, ils ont été utiles pour se poser plusieurs questions sur les informations nécessaires à afficher. Le but n'était pas ici d'aller dans le détail, mais de se rendre compte de la logique et des composants principaux des différentes pages.

- **Technologies**

Kewin a effectué une recherche sur les quelques technologies discutées la dernière fois. Après avoir récolté ces informations, bien que certaines technologies semblent plus pertinentes que d'autres actuellement, aucune n'a été complètement écartée. Certaines ont tout de même été mises en lumière : Le principe des PWA, React, React Native ou Ionic, Cordova et Redux. À l'inverse, il semblait que Firebase et ElasticSearch n'étaient pas une priorité.

- **Rapport écrit**

Il a été défini que les modalités de l'écriture du rapport écrit étaient les mêmes que celles définies dans le cadre d'un projet de Bachelor. À savoir que le document sera écrit progressivement sur la durée du projet, et des retours allaient avoir lieu en tout cas sur la première moitié de celui-ci.

- **Prochaines étapes**

La réunion de la semaine prochaine aura lieu à 10h30. Jusqu'ici, l'accent va être mis sur la compréhension et la mise en place d'un exemple d'utilisation des technologies, afin d'être le plus au clair possible avec le fonctionnement et les possibilités de celles-ci.

### Conclusion

Le but est maintenant de se faire la main sur des technologies telles que React afin de se rassurer sur le fonctionnement du stack complet.

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



## PV de réunion

6 mars 2016, de 10h30 à 11h15, en D20.19

Présent : Sandy Ingram, Kewin Dousse

Rédaction du PV le 6 mars

### Compte-rendu

#### Points de discussion

- Frontend

Kewin a commencé à apprendre à utiliser le framework React ainsi que Redux. Quelques composants sont implémentés, mais pour l'instant pas suffisamment pour constituer une page servant d'exemple. Ceci sera terminé pour la semaine prochaine, et quelques pages servant de mockup seront effectuées et présentées.

- Backend

Le développement sur le backend a subi quelques imprévus. Le développement a tout d'abord commencé en Python, avant de remarquer un problème au niveau de l'ouverture d'un port. Ce souci est encore en suspens, et une demande sera faite à Alexandre demain pour débloquer un port (par exemple, le port 8080). La plupart des ports actuels semblent être bloqués par un firewall sur le réseau.

En dehors de ceci, le développement backend se poursuivra peut-être en node.js afin de rester avec du JavaScript sur toute la stack technologique.

- Questions

Plusieurs questions sur la conception du service sont actuellement sans réponses. Celles-ci ont été listées durant la séance, et seront posées demain lors de la réunion sur Skype. Elles concernent par exemple la méthode de remplissage de certaines tables de la base de données, ou la manière d'utiliser l'application par les futurs utilisateurs.

- Cahier des Charges

Un planning va être établi rapidement et incorporé dans le Cahier des Charges, afin de pouvoir s'appuyer une découpe temporelle des tâches du projet, ainsi que de valider les objectifs de celui-ci par exemple.

#### Conclusion

Il est maintenant essentiel de répondre aux dernières questions sur l'analyse de la solution à réaliser, et de terminer l'apprentissage des technologies nécessaires à celui-ci.

#### Points d'action

- Implémentation d'exemple avec React, Redux etc
- Planning avec Cahier des Charges
- Envoyer des slides de cours

#### Personne responsable

Kewin Dousse  
Kewin Dousse  
Sandy Ingram

13.03.2017  
07.03.2017  
13.03.2017

## ANNEXE D. PROCÈS-VERBAUX

---

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



# PV de réunion

7 mars 2016, de 17h00 à 17h35, par Skype

**Présent :** Sandy Ingram, Alexandre Terrier, Kewin Dousse

Rédaction du PV le 7 mars

## Compte-rendu

### Points de discussion

- **Interactions avec la base de données**

La réunion a porté autour de questions posées à Alexandre. La première d'entre-elles tenait sur les sources venant remplir la base de données. Après discussion, il est noté que l'interface à concevoir devra comporter une page pour la saisie manuelle de données de mesures.

- **Sources d'images**

La discussion a ensuite porté sur la source des images à afficher. Celle-ci n'est pas encore totalement fixée. Les images seront stockées d'un serveur différent sur le network, derrière un firewall, mais leur provenance exacte et la manière d'y accéder sera définie plus tard.

- **Données actuelles, sens de la table « Studies »**

La base de données actuelle n'est pas encore complète ; Certaines tables ne sont pas remplies. Ceci se fera durant le développement du projet. Une question a été posée sur le sens de la table « Studies » en particulier : Le but est de pouvoir grouper certains cas dans une même étude, et la configuration actuelle de la table n'est pas définitive et pourra donc être adaptée en cas de besoin.

- **Ports ouverts**

L'application aura besoin d'un port ouvert pour dialoguer. La plupart des ports semblent restreints par un firewall sur le réseau. Deux possibilités pour l'application s'offrent ici : Demander à ouvrir un port supplémentaire sur le réseau, ou occuper un des ports déjà ouverts. Une tentative sera faite d'occuper le port 80 actuellement utilisé par phpMyAdmin, en cohésion avec celui-ci.

### Conclusion

Maintenant que la plupart des zones d'ombres sont levées, des décisions techniques vont pouvoir être prises afin de démarrer le développement de l'application.

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



## **PV de réunion**

**13 mars 2016**, de 10h30 à 11h25

**Présent** : Sandy Ingram, Kewin Dousse

Rédaction du PV le 14 mars

### **Compte-rendu**

#### **Points de discussion**

- **Maquette live**

La réunion a commencé sur la maquette actuelle de l'application. Un template a été réalisé avec React ainsi que MDL. Il se trouve toutefois que MDL souffre de plusieurs lacunes au niveau du contenu, et sera probablement remplacé par Materialize bientôt.

- **Réunion de vendredi**

La question s'est posée de la direction qu'allait prendre la réunion de vendredi. Nous allons profiter de celle-ci pour avoir un point de vue d'un utilisateur de l'application ayant une vue clinique, comparée à la vue d'ensemble que nous avions jusqu'ici. Cela va nous permettre de mieux comprendre une partie des utilisateurs.

- **Cahier des charges**

Quelques corrections ont été apportées sur les activités du cahier des charges : Les activités ont changé, et le diagramme de Gantt sera mis à jour.

- **Design de l'interface**

Le design de l'application actuel a été mis en question sur de multiples points. Etant donné que le design est loin d'être final car nous manquons de points de vue d'utilisateurs, celui-ci est subira encore beaucoup de changements.

#### **Conclusion**

La rencontre avec tous les participants au projet aura lieu dans la semaine. Pendant ce temps, le développement de l'application continue.

<b>Points d'action</b>	<b>Personne responsable</b>	<b>Echéance</b>
• Préparation de la réunion à Lausanne	Kewin Dousse	17.03.2017
• Continuation de l'implémentation	Kewin Dousse	20.03.2017

## ANNEXE D. PROCÈS-VERBAUX

---

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



MASTER OF SCIENCE  
IN ENGINEERING

# PV de réunion

17 mars 2017, de 9h05 à 10h05

Présent : Alain Farron, Alexandre Terrier, Fabio Becce, Kewin Dousse, Sandy Ingram

Rédaction du PV le 19 mars

## Compte-rendu

### Points de discussion

- Première version de maquette

Un premier prototype de l'interface a été montré. Celle-ci n'étant qu'à l'état de prototype basse fidélité, le but était plutôt de lancer une discussion sur l'utilisabilité de celle-ci que de montrer des fonctionnalités terminées.

- Différentes vues

Il s'est rapidement trouvé que les utilisateurs de l'applications pouvaient être classés en deux catégories, chacune nécessitant une vue particulière. La vue « Cases » rassemblera des informations sur les cas, où un cas est associé à une épaule et un ou plusieurs CT. La deuxième vue, « Patients », allait montrer des informations sur les patients eux-mêmes, avec leurs cas correspondants. Un patient pourra évidemment avoir plusieurs cas. La nécessité de cette séparation a été mise en lumière

- Sources de données

Des précisions ont été apportées sur les sources qui vont alimenter la base de données. Les mesures seront ajoutées par un moyen externe à l'interface, et il n'est donc pas nécessaire de prévoir une interface pour ceci.

- Fonctionnalités

Quelques fonctionnalités encore non soulevées jusqu'ici ont été discutées : La possibilité de filtrer la liste des patients sur des paramètres relatifs à leur(s) cas, la possibilité d'exporter les données affichées sous plusieurs formats, et la possibilité d'ajouter manuellement via l'interface des patients qui n'ont pas encore de cas affectés.

### Conclusion

La rencontre avec tous les participants a mené des points de vue différents, et beaucoup de clarifications sur des points du projet qui étaient encore sombres jusqu'ici. Kewin a pu avoir une vision plus claire de l'objectif général du projet, et des fonctionnalités demandées ainsi que la manière dont celles-ci seront utilisées.

Points d'action	Personne responsable
<ul style="list-style-type: none"><li>• Envoi d'images d'exemples à afficher sur l'interface</li><li>• Envoi de la base de données Excel</li></ul>	? Alexandre Terrier Effectué

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



# PV de réunion

**20 mars 2017**, de 13h00 à 13h15

**Présent** : Kewin Dousse, Sandy Ingram

Rédaction du PV le 20 mars

## Compte-rendu

### Points de discussion

- **React**

Après avoir suivi un tutoriel sur React, il se trouve que les changements entre versions sont nombreux et que le tutoriel n'était pas à jour. L'interface actuelle a été réécrite dans une grande partie, et la mise en place du framework React + Flux prend plus de temps que prévu initialement.

- **Rapport écrit**

À présent que la plupart des informations de l'analyse sont connues, Kewin va commencer l'écriture du rapport écrit pour la semaine prochaine.

### Conclusion

La réunion fut courte, étant donné la plupart des informations nécessaires ont été discutées dans la réunion du vendredi passé à Lausanne. L'implémentation du projet continue, et une première partie du rapport va être rédigée.

Points d'action	Personne responsable	Echéance
• Mise à jour de l'interface avec nouvelle version de React	Kewin Dousse	27.03.2017
• Première partie du rapport écrit	Kewin Dousse	27.03.2017

## ANNEXE D. PROCÈS-VERBAUX

---

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



# PV de réunion

27 mars 2017, de 10h30 à 11h05

Présent : Kewin Dousse, Sandy Ingram

Rédaction du PV le 28 mars

## Compte-rendu

### Points de discussion

- **Rapport écrit**

La première version du rapport écrit a été présentée. Celle-ci contient la structure générale du document, ainsi que l'Introduction, la partie d'Analyse des besoins et le cahier des charges en annexe. Plusieurs remarques ont été faites sur la structure des différentes parties, et leur contenu.

- **Choix du backend**

Actuellement, un début de backend avait été écrit en python. L'idée est à présent de se poser la question de quelle technologie utiliser, en explorant les possibilités de nodejs. Utiliser du javascript côté backend également simplifierait en effet le déploiement de l'application finale.

### Conclusion

Les informations récoltées ont été placées dans le rapport, mais la partie technique est encore à rédiger. En parallèle, l'avancement du backend permettra de continuer la réflexion sur l'interface dès la semaine prochaine.

Points d'action	Personne responsable	Echéance
• Correction du rapport écrit	Kewin Dousse	01.04.2017
• Analyse de la partie technologique backend	Kewin Dousse	03.04.2017

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed

# **PV de réunion**

**3 avril 2017**, de 10h30 à 11h05

**Présent** : Kewin Dousse, Sandy Ingram

Rédaction du PV le 4 avril



MASTER OF SCIENCE  
IN ENGINEERING

## **Compte-rendu**

### **Points de discussion**

- Rapport écrit**

Une version comprenant l'Analyse a été envoyé durant le week-end. Des corrections ont été suggérées durant la réunion, et une critique de la partie technique sera envoyée plus tard afin d'être corrigée pour la semaine suivante.

- Backend**

La technologie du backend a changé, et celui-ci sera maintenant écrit en NodeJS. Un début de serveur web a été écrit, mais la connexion avec la base de donnée distante ainsi que la cohabitation avec l'actuel phpMyAdmin en place restent à régler.

### **Conclusion**

Les informations récoltées ont été placées dans le rapport, mais la partie technique est encore à rédiger. En parallèle, l'avancement du backend permettra de continuer la réflexion sur l'interface dès la semaine prochaine.

<b>Points d'action</b>	<b>Personne responsable</b>
<b>Echéance</b>	
•Remarques sur la partie technologique	Sandy Ingram
•Corrections du rapport	10.04.2017
•Implémentation de la logique principale backend	Kewin Dousse
	Kewin Dousse
	10.04.2017

## ANNEXE D. PROCÈS-VERBAUX

---

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed

# PV de réunion

10 avril 2017, de 11h00 à 11h10

**Présent :** Kewin Dousse, Sandy Ingram

Rédaction du PV le 12 avril



MASTER OF SCIENCE  
IN ENGINEERING

## Compte-rendu

### Points de discussion

- **Retard**

Kewin n'a pas trouvé le temps de travailler suffisamment sur le projet pour accomplir les tâches prévues pour cette semaine. La partie logique du backend n'est donc pas terminée, et sera à rattraper pour le lundi après la semaine de vacances.

- **Liaison Front-End**

Afin de continuer d'avancer au rythme convenu, il sera nécessaire également que pour le lundi 24, le front-end soit capable de communiquer avec le backend et récupérer des données de la base de données afin de les afficher correctement.

### Conclusion

Un peu de retard est à rattraper cette semaine, mais cela est raisonnable au vu de la semaine au vu de la semaine de congé. Après celle-ci, le prototype devrait se trouver dans un état où le backend est fonctionnel, et où le frontend est capable de communiquer avec.

Points d'action	Personne responsable
<b>Echéance</b>	
•Remarques sur la partie technologique	Sandy Ingram
•Corrections du rapport	Kewin Dousse
•Implémentation de la logique principale backend	Kewin Dousse
•Reprise du développement front-end des maquettes	Kewin Dousse

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



## PV de réunion

24 avril 2017, de 13h15 à 13h45

Présent : Kewin Dousse, Sandy Ingram

Rédaction du PV le 24 mars

### Compte-rendu

#### Points de discussion

- **Implémentation**

Les problèmes de backend ont été résolus, et la logique principale du serveur est implémentée. Celui-ci livre correctement la page web, et communique avec le code front-end pour lui servir des données de la base de données : actuellement la liste des patients. La page phpMyAdmin a été désactivée pour des raisons de simplicité.

Il sera nécessaire de commenter et d'expliquer l'architecture de l'application mise en place dans la partie technique du rapport.

- **Evaluation formative**

Une évaluation de l'interface actuelle a été faite durant la réunion, il en est sorti que plusieurs aspects de la page Patients actuelle devront être corrigés. Ces corrections seront faites pour vendredi, ainsi que l'ajout d'informations actuellement manquantes.

#### Conclusion

À présent que la partie logique principale du backend est en place, les efforts vont se concentrer sur l'implémentation des fonctionnalités manquantes, ainsi que sur la partie d'interface utilisateur afin de la rendre la plus ergonomique possible.

#### Points d'action

- Corrections de la page Patients
- Ajout des explications dans le rapport
- Ajout de la page « Cases »

#### Personne responsable

Kewin Dousse  
Kewin Dousse  
Kewin Dousse

#### Echéance

28.04.2017  
08.05.2017  
08.05.2017

## ANNEXE D. PROCÈS-VERBAUX

---

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



# PV de réunion

8 mai 2017, de 10h30 à 11h40

Présent : Kewin Dousse, Sandy Ingram

Rédaction du PV le 9 mai

## Compte-rendu

### Points de discussion

#### • Etat d'avancement

Kewin a exprimé rencontrer des difficultés à implémenter dans le temps imparti les modifications discutées lors de séances précédentes, ainsi que des doutes sur la faisabilité du reste du projet. Après une discussion, la cause est probablement l'inexpérience avec le stack technologique frontend, à savoir React, Flux et Material-UI. La confusion créée par certains points discutés brièvement par le passé a été levée après un passage en revue de l'état de certaines tâches : Etat de l'UI, du rapport. La faisabilité de certaines fonctionnalités semble compromise et rediscutée plus tard : login, et possibilité de modification des données.

#### • Evaluation formative de l'UI

Une évaluation de l'UI des pages Patients et Cas a été faite durant la réunion. Une liste de points à modifier en est sortie : Ces modifications seront effectuées pour vendredi afin d'avoir un retour jusqu'à la réunion suivante.

#### • Structure du rapport

Kewin était également incertain de quelle partie du rapport allait contenir quelles informations. Une revue du contenu des sections actuelles a donc été faite, et quelques remaniements sont à faire afin d'obtenir une structure logique. De plus, le contenu du rapport sera à rendre jusqu'à la fin mai afin d'avoir une semaine de disponible avant le rendu final.

### Conclusion

Le projet se situe environ aux deux tiers en termes de temps. Le stack est fonctionnel et les deux pages principales sont là, mais pas mal de modifications restent à implémenter, et de fonctionnalités à ajouter afin d'arriver au résultat final.

Points d'action	Personne responsable	Echéance
• Changements de l'UI des pages Patients et Cases	Kewin Dousse	12.05.2017
• Réflexion sur la page d'ajout de Patients	Kewin Dousse	15.05.2017
• Complétion du rapport écrit	Kewin Dousse	29.05.2017

## ANNEXE D. PROCÈS-VERBAUX

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



# PV de réunion

15 mai 2017, de 11h05 à 11h20

Présent : Kewin Dousse, Sandy Ingram

Rédaction du PV le 15 mai

## Compte-rendu

### Points de discussion

- Evaluation de l'UI des pages Cas et Patients

Les points soulevés concernant l'interface des pages Cas et Patient ont été corrigés et implémentés. Le filtrage des informations et la recherche sont fonctionnels sur les deux pages. Quelques changements sont encore à faire pour peaufiner la page, et seront faits pour la semaine prochaine.

- Affichage / Edition des données d'un patient

D'après le feedback de Alexandre, la possibilité d'édition des données est la feature restante la plus importante pour l'application. Une idée a été amenée de faire une page unique pour la visualisation ainsi que l'édition des données d'un patient ou cas. Le prototype de cette page sera fait et envoyé pour vendredi, dans le but d'avoir un retour le lundi suivant. Mentionné également : L'installation de phpMyAdmin n'est plus accessible, car le port 80 est désormais utilisé par l'application.

### Conclusion

Maintenant que les pages montrant les données complètes de la base sont présentes, il reste à créer une page montrant les données d'un seul Patient/Cas.

Points d'action	Personne responsable	Echéance
• Implémentation de la page Affichage/Edition	Kewin Dousse	22.05.2017
• Complétion du rapport écrit	Kewin Dousse	29.05.2017

## *ANNEXE D. PROCÈS-VERBAUX*

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed

# PV de réunion

**22 mai 2017**, de 10h40 à 11h15

**Présent** : Kewin Dousse, Sandy Ingram

Rédaction du PV le 22 mai

MSE | MASTER OF SCIENCE  
IN ENGINEERING

MASTER OF SCIENCE  
IN ENGINEERING

## Compte-rendu

## **Points de discussion**

- **Evaluation de l'UI de la pages Cas**

La page d'affichage et d'édition d'un Cas a été réalisée. Au moment de la réunion, le retour d'Alexandre nous était déjà parvenu. Après discussion des changements à effectuer, tous les points ne pourront pas être réalisés à cause du manque de temps. Les modifications qui seront faites sont le passage de la langue en anglais, l'ajout du champ de date et la mise en valeur du Folder Name. L'ID du cas sera également caché dans le header, au profit du Folder Name. De plus, la première catégorie actuellement nommée « Général » sera renommée « Patient » et affichera les informations du patient, ainsi qu'un bouton d'édition amenant sur la page de celui-ci.

- **Page Patient**

La page Patient permettra d'éditer les données du patient, ainsi que de lister les cas de celui-ci. Cet ajout sera fait pour lundi prochain.

- Rapport écrit

Pour vendredi, la suite du rapport sera rédigée. Le but est d'arriver à un stade presque final du point de vue de la complétude : Les étapes actuellement manquantes seront ajoutées.

# Conclusion

Le temps restant avant la fin du projet s'amenuise, les dernières implémentations prennent place cette semaine.

<b>Points d'action</b>	<b>Personne responsable</b>
<b>Echéance</b>	

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



## PV de réunion

29 mai 2017, de 13h00 à 13h55

Présent : Kewin Dousse, Sandy Ingram

Rédaction du PV le 29 mai

### Compte-rendu

#### Points de discussion

- Evaluation de l'UI des pages Cas et Patient

La page d'affichage et d'édition d'un Patient a été réalisée. Une évaluation de celle-ci ainsi que de la version modifiée de la page de Cas a été faite, et quelques retouches sont encore à faire.

- Rapport écrit

La structure du rapport a été rediscutée durant la réunion, et a changé un peu de forme. Au lieu de présenter les parties « Conception », « Réalisation » et « Evaluation », les chapitres seront divisés par prototypes d'abord, puis chaque prototype sera discuté selon les 3 points de vue « Conception », « Réalisation » et « Evaluation ». Cette structure met en valeur l'approche itérative du projet.

Plusieurs autres changements mineurs sur le rapport ont été proposés, et seront corrigés. De plus, certaines parties restent encore à rédiger : Ceci sera fait au plus vite.

#### Conclusion

L'application est dans un état quasi final, et le temps restant est principalement attribué à la rédaction du rapport.

Points d'action	Personne responsable	Echéance
• Nouvelle table des matières	Kewin Dousse	29.05.2017
• Code de l'application	Kewin Dousse	31.05.2017
• Première version du rapport final	Kewin Dousse	02.06.2017
• Deuxième version du rapport final	Kewin Dousse	05.06.2017
• Renseignement sur la license du code du projet	Sandy Ingram	05.06.2017

## ANNEXE D. PROCÈS-VERBAUX

---

PV de réunion VisMed

Projet d'Approfondissement printemps 17 : VisMed



MASTER OF SCIENCE  
IN ENGINEERING

# PV de réunion

6 juin 2017, de 10h30 à 11h25

Présent : Kewin Dousse, Sandy Ingram

Rédaction du PV le 6 juin

## Compte-rendu

### Points de discussion

- Rapport écrit

La structure du rapport a été en partie revue durant la réunion. Un certain nombre de modifications de structure ont été suggérées afin que certaines informations soient mieux placées dans le rapport complet.

La question du rendu du rapport a également été répondue : Une copie écrite sera livrée à Sandy ainsi que le PDF par mail. Le code est accessible sur le GitLab de la HEIA-FR sur demande.

### Conclusion

Le code est à présent terminé, et seules quelques modifications dans le rapport seront encore faites.

Points d'action	Personne responsable	Echéance
• Renseignement sur la license du code du projet	Sandy Ingram	08.06.2017
• Rapport final	Kewin Dousse	09.06.2017