Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2019

Interface d'administration pour la plateforme Pryv.io

Étudiant : Vlado Mitrovic

Professeur : Dr. Michael Ignaz Schumacher

Déposé le : 31 juillet 2019

Résumé

Comment collecter des données sensibles lorsque l'on mène un projet de recherche ? Cette question constitue le point d'ancrage de notre travail dont le but repose sur l'analyse et l'implémentation d'une solution permettant la centralisation des données récoltées.

Pour ce faire, nous débuterons par présenter Pryv. Cette entreprise Suisse est spécialisée dans la gestion de données personnelles dans le domaine de la santé numérique en particulier.

Ensuite, nous allons définir les besoins de la DAunit (Data Acquisition Unit), partie intégrante de l'e-santé à l'Hes-so (Haut école spécialisé de Suisse occidentale), qui souhaite intégrer la technologie de Pryv. Dans cette optique, nous allons définir les cas d'utilisation afin de modéliser par la suite les prototypes visuels de cette interface.

A posteriori, le côté technique de ce développement sera analysé afin de choisir l'architecture et les technologies les plus adaptées à ce projet.

Nous documenterons finalement la phase pratique de cette étude dans le but de décrire l'intégration de Pryv à notre solution ainsi que les problèmes rencontrés et les solutions apportées.

Mots clés : Pryv, données sensibles, e-santé, gestion projets

Avant-propos

Ce travail a été effectué dans le cadre d'un travail de Bachelor en informatique de gestion au sein de l'Hes-so Valais/Wallis. Il s'agit de la dernière étape qui conclut un programme d'étude de 3 années. Celui-ci a débuté le 30 mai 2019 et s'est terminé le 31 juillet 2019 pour une charge de travail représentant 12 ECTS (European Credit Transfer Scale) qui est évalué à environ 360 heures de travail.

Le sujet a été proposé par le Dr. Prof. Michael Ignaz Schumacher en lien avec la DAunit. Lors du choix du sujet pour ce travail, mon attention s'est tout de suite portée vers ce thème, car je trouvais très intéressant de découvrir et intégrer une nouvelle technologie telle que Pryv. De plus, les nouvelles lois en vigueur sur la protection de données en font un sujet d'actualité, ce qui rend l'entreprise Suisse très prometteuse dans ce marché encore trouble.

Ce rapport a pour objectif de documenter la planification faite afin de mener à bien ce projet, ainsi que les difficultés rencontrées tout au long de cette étude.

Remerciements

Au Dr. Prof. Michael Ignaz Schumacher, pour nous avoir suivis et encadrés tout au long de ce projet ainsi que pour sa disponibilité et son implication.

À M. Jean-Paul Calbimonte Pérez, pour ses idées et ses conseils techniques mais également pour sa disponibilité et son implication.

À M. Roger Schaer pour son soutien technique sur le déploiement du projet.

Toutes les personnes qui ont pris le temps de relire et corriger ce rapport afin d'y apporter une meilleure qualité.

Liste des illustrations

Liste des tableaux

Liste des abréviations

RGPD	Règlement général sur la protection des données
DAunit	Data Acquisition Unit
PB	Product Backlog
PO	Product Owner
SP	Story point
US	User Story
Http	Hypertext Transfer Protocol
API	Application programming interface
URL	Uniform Resource Locator
VCS	Version Control System
JSON	JavaScript Object Notation
SQL	Structured query language
NoSQL	Not only SQL
REST	Representational state transfer
MVC	Model view controller
MVT	Model view template
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

TABLE DES MATIÈRES

Introduction

Comment collecter des données sensibles de manière sécurisée lorsque l'on mène un projet de recherche ?

De nos jours, l'un des défis majeurs du monde technologique est de respecter la vie privée des utilisateurs, c'est pour cela que le nombre de lois à ce sujet est grandissant. Cette raison constitue la principale motivation de notre choix pour la réalisation de ce travail.

L'objectif de ce projet est de développer une interface qui permettrait de gérer des projets d'acquisition de données tout en incluant un système de stockage simple et sécurisé. Pour cela, un des buts de ce travail est d'intégrer la plateforme de gestion du cycle de vie des données de l'entreprise Pryv SA.

Pour réaliser ce travail, nous devons dans un premier temps comprendre les concepts et le fonctionnement de cette plateforme. Nous définirons ensuite les besoins de notre utilisateur final, la DAunit dans notre cas, afin de lister les fonctionnalités requises par cette interface. Une fois ces tâches réalisées, nous analyserons les différentes possibilités quant aux choix technologiques afin de finalement implémenter le produit final.

Toutes les décisions qui seront prises au cours de ce projet seront très importantes et déterminantes quant au succès de notre objectif.

1. Problématique

Lorsque l'on entreprend une recherche au niveau médical, il est important de respecter certaines lois qui touchent au traitement de données sensibles et privées comme le RGPD (Règlement Général sur la Protection des Données) par exemple. Afin de se conformer à cet aspect législatif, l'entreprise Suisse Pryv a développé une plateforme de gestion de données privées qui offre une solution permettant l'accès à ces données. De cette manière, l'échange de données personnelles entre les patients et les médecins est facilité et sécurisé.

La DAunit (Data Acquisition Unit), qui fait partie de l'unité e-santé de l'Hes-so de Sierre, offre des services pour les chercheurs du domaine médical en ce qui concerne les acquisitions de données. La solution de Pryv est idéale en termes de collecte et partage de données. C'est pourquoi, une instance a été installée sur le serveur de l'Hes-so.

Actuellement, Pryv ne dispose d'aucune interface d'administration, ce faisant, l'ensemble des manipulations est réalisé avec des requêtes HTTP (Hypertext Transfer Protocol). Les personnes en charge de ces projets ne sont pas forcément liées au domaine de l'informatique ce qui rend la tâche difficile voire impossible à réaliser.

1.1. Solution

Afin d'intégrer cette solution à la DAunit, l'idée serait de créer une interface pour la gestion de projets avec des fonctionnalités telles que la création d'utilisateurs Pryv. Cela procurerait une vue d'ensemble sur cette instance et permettrait la génération de token[1] pour l'accès aux données par exemple. De plus, une gestion de projet permettrait le regroupement des patients et faciliterait, ainsi, l'export et le traitement de ces données.

[1] Token : jeton en français, clé pour accéder aux données d'un utilisateur Pryv.

2. Déroulement du projet

2.1. Les étapes

Pour la réalisation de ce travail, nous définissons les quatre phases suivantes selon les données du travail de bachelor.

1. Se familiariser avec la plateforme Pryv et ses concepts
2. Définir les besoins de la DAunit et réaliser des mockups[2] de cette interface
3. Faire un état de l'art des technologies afin de choisir la plus adaptée pour ce projet
4. Développer cette interface d'administration de manière itérative avec la DAunit

2.2. Gestion de projet

Comme la phase de développement se fait de manière itérative, nous allons utiliser la méthodologie « Scrum » tout au long de ce projet. À ce sujet, il est important de clarifier certaines notions qui seront mentionnées dans ce rapport.

Le PO (Product Owner) est la personne qui prend les décisions et détermine les priorités du projet. Il ne s'agit pas forcément du client. Cela peut être une personne avec des connaissances techniques qui est mandatée par le client créant, ainsi, le lien avec le développeur.

Le PB (Product Backlog) liste les fonctionnalités qui devront être implémentées sous forme de cas d'utilisation appelé ici user story. Celles-ci seront évaluées en story point, représentant la quantité de charge de travail requise pour l'implémentation de la fonctionnalité.

Un sprint représente une itération dans le développement et dure généralement 2 à 4 semaines. Au début de chaque sprint, nous sélectionnons avec le PO quelques user stories qui seront implémentées selon les priorités et le temps à disposition. Ensuite, à la fin de chaque sprint, chacune d'entre elles est validée indépendamment des autres par le PO. Le sprint 0

[2] Mockups : Prototype de l'interface utilisateur

représente la mise en place de tous les outils avant de commencer le développement du produit.

2.3. Planification

Au niveau de la planification, le travail de bachelor débute le 30 avril et se termine le 31 juillet 2019. Nous fixons le délai au 14 mai pour réaliser les mockups selon les besoins de la DAunit. La fin du sprint 0 prévu le 3 juin marquera le début de la phase de développement qui sera composée de trois sprints d'une durée de deux semaines chacun. Dès le 16 juillet, nous clôturerons le dernier sprint et donc la partie pratique de ce travail. Le reste du temps sera ensuite consacré à la rédaction de ce rapport ainsi qu'aux dernières tâches administratives requises par la Hes-so.

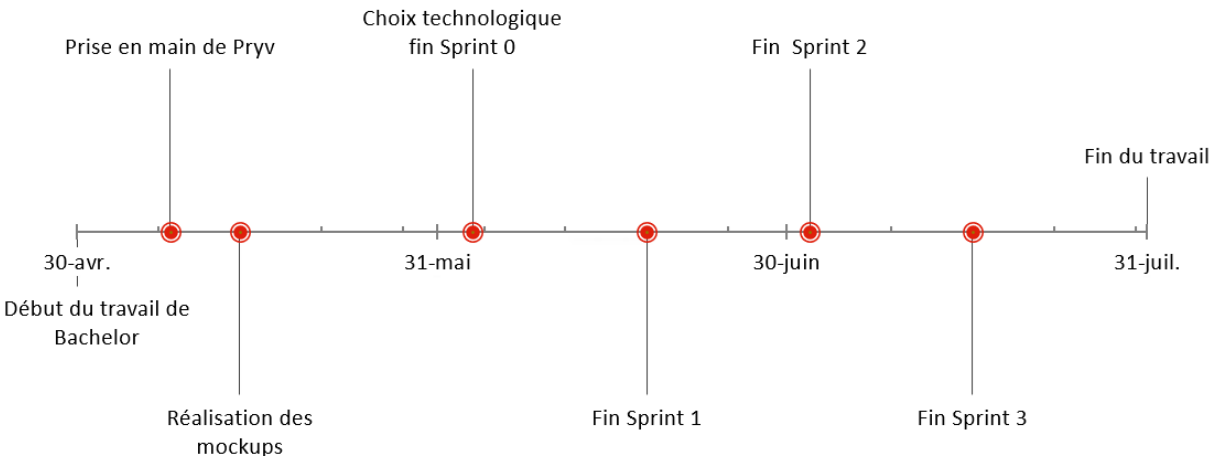


Figure 1 : Déroulement du projet

3. La solution Pryv

« La société Pryv SA, dont le siège est à Ecublens (VD), a pour but toutes activités quelconques dans le domaine de l'informatique, notamment la conception, l'exploitation, la gestion et la sécurisation des données, la distribution de logiciels et de services, ainsi que le conseil en matière de systèmes d'information. » (Registre du commerce du canton de Vaud, s.d.)

Pryv.io est une plateforme destinée à faciliter la création, l'utilisation, le partage et l'élimination de données personnelles. Elle a pour but de garantir la protection de données privées afin de respecter certaines lois comme le RGPD notamment. (Pryv SA, s.d.)

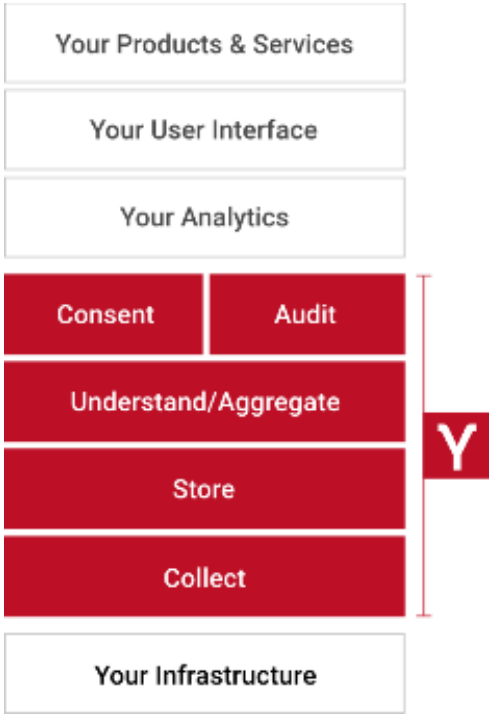


Figure 3 : Positionnement de la solution (Pryv, 2019)

Comme l'illustre cette figure, la solution Pryv ici en rouge agit en tant que middleware, c'est-à-dire que l'on rajoute une couche entre notre base de données et notre service. De ce fait, l'instance de Pryv gérera tout ce qui se rapporte au stockage et au partage de ces données.

On peut ainsi enregistrer nos données personnelles, les consulter puis les partager à des tiers. De plus, il est possible de donner différents niveaux d'accès allant de la lecture simple à la gestion totale. Par ailleurs, il est également possible de révoquer ces autorisations à tout moment s'il l'on ne limite pas la durée de validité lors de la création.

En résumé, Pryv est une plateforme de gestion du cycle de vie des données personnelles, directement prête à l'emploi, et qui a été conçue pour permettre aux développeurs de créer et de s'adapter rapidement aux produits et services conformément au RGPD. (Pryv SA, s.d.)

3.1. Structure de l'API

Pryv fonctionne comme une api REST (Representational state transfer) et communique donc au moyen de diverses méthodes HTTP. Celle-ci possède plusieurs URL (Uniform Resource Locator) qui permettent d'interagir avec la solution.

Tout d'abord, le point de terminaison « reg », celui-ci est dédié à toute la partie de gestion des utilisateurs comme l'enregistrement notamment. C'est également ici que nous allons retrouver les hébergements disponibles. L'adresse « http://reg.pryv.hevs.ch/hostings » nous permet de récupérer les lieux de stockage proposés par Pryv appelé « core » dans ce contexte.

Comme le lieu de stockage peut différer d'un utilisateur à un autres, chacun d'entre eux possède leur propre URL dans l'API. Sur l'instance de l'Hes-so, le chemin sera composé de cette manière :

- https://*{nom d'utilisateur}*.pryv.hevs.ch

Ce faisant, l'instance Pryv redirigera la requête vers le core que l'utilisateur aura choisi lors de son inscription. Chacun détiendra sa base de données indépendamment des autres utilisateurs présents sur le même core. Voici un schéma illustrant cela :

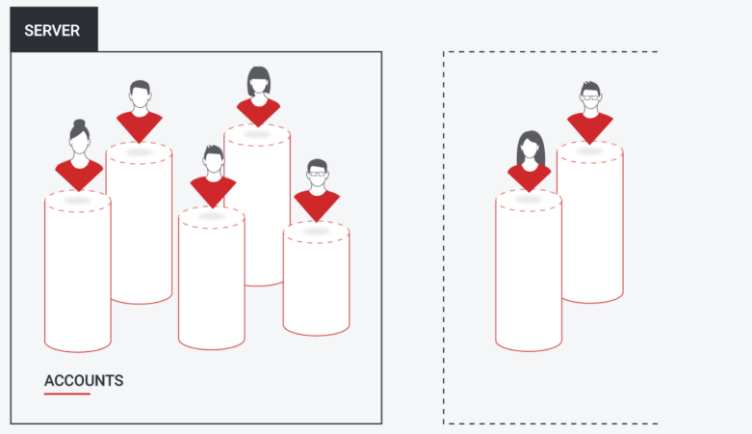


Figure 4: Les utilisateurs dans un core (Pryv SA, 2019)

Dans le cas de la Hes-so, un seul lieu de stockage est disponible. En revanche, si nous prenons l'instance de démonstration « pryv.me », on y retrouve cinq choix différents.

3.2. Structure des données

La structure des données est très similaire à une base de données non relationnelle telle que mongoDB qui possède des collections contenant des documents. Malgré cela, la terminologie de Pryv est différente, car on parle ici de « stream » et « event ».

3.2.1. Stream

Celui-ci correspond à une collection d'events, on pourrait le comparer à un dossier contenant des documents. Chaque Stream contient les propriétés suivantes :

- name : nom du stream
- parentId : identifiant du stream parent
- created : heure de création Unix[3]
- createdBy : identifiant du créateur
- modified : heure de modification Unix
- modifiedBy : identifiant de l'éditeur
- id : identifiant
- children : liste des streams enfants

3.2.2. Event

Un event correspond à un document dans un stream, on pourrait le comparer à un document dans un dossier. Chaque Event contient les propriétés suivantes :

- streamId : l'identifiant du stream dont l'event fait parti
- type : le type de la donnée
- content : la valeur de la donnée
- time : heure Unix
- tag : tag permettant de filtrer les events

[3] Heure Unix : mesure du temps basée sur le nombre de secondes écoulées depuis le 01.01.1970 00:00, souvent utilisée dans le domaine de l'informatique

On retrouve également, id, created, createdBy, modified, modifiedBy qui ont les mêmes caractéristiques que pour un stream.

Il est possible d'utiliser n'importe quel type pour un event. Malgré cela, il est fortement conseillé d'utiliser les types proposés par Pryv dans le but de garder une interopérabilité qui facilite l'échange de données entre différents systèmes (Pryv SA, s.d.). En juillet 2019, l'entreprise Suisse compte un peu plus de 300 différents types. Il faut relever qu'il est également possible d'en proposer de nouveaux. La manière, dont Pryv gère la structure de ses données, offre la possibilité d'avoir un niveau plus détaillé en comparaison d'un stockage traditionnel.

L'ajout de tags à nos events nous permet de les filtrer et par conséquent offre une certaine classification de nos données. Cela devient très utile lorsque nous souhaitons traiter un gros volume d'events.

Nous présentons, ci-dessous, un schéma illustrant les données d'un utilisateur avec les notions évoquées précédemment.

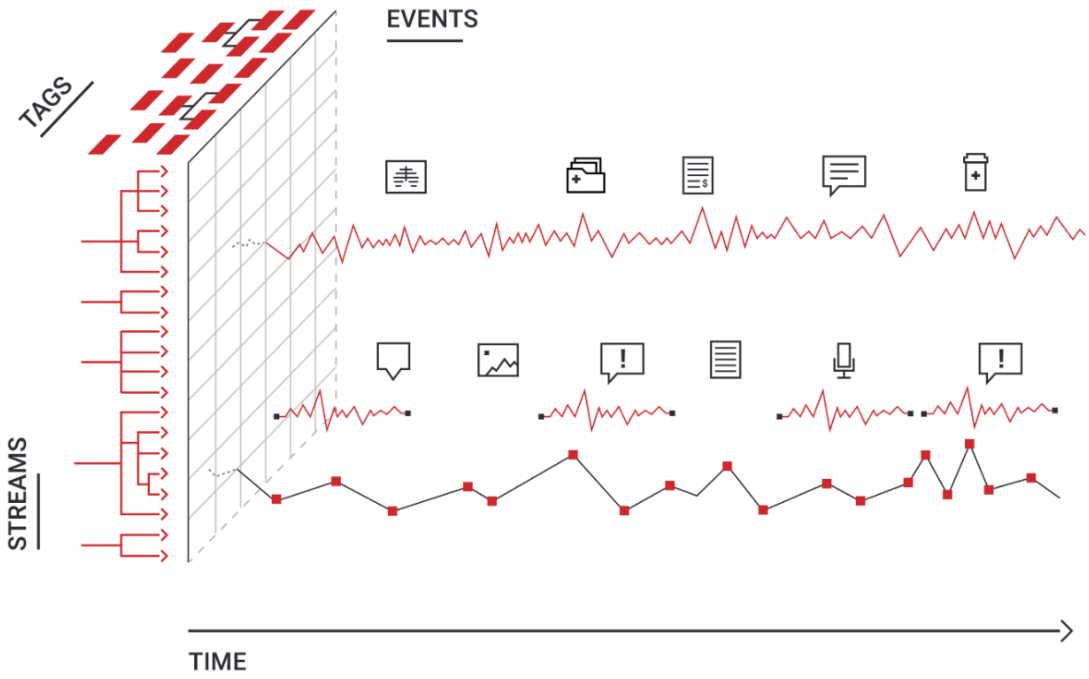


Figure 5: Structure des données (Pryv SA, s.d.)

3.2.3. Différence avec mangoDB

Dans le cas de Pryv, un Stream peut contenir des Events mais également d'autres Streams enfants, qui, eux-mêmes, peuvent renfermer des Events et des Streams ainsi de suite. En comparaison, dans mongoDB, nous avons une collection comportant seulement des documents pouvant eux-mêmes contenir des sous-documents. Cette différence s'illustre au travers de ce schéma :

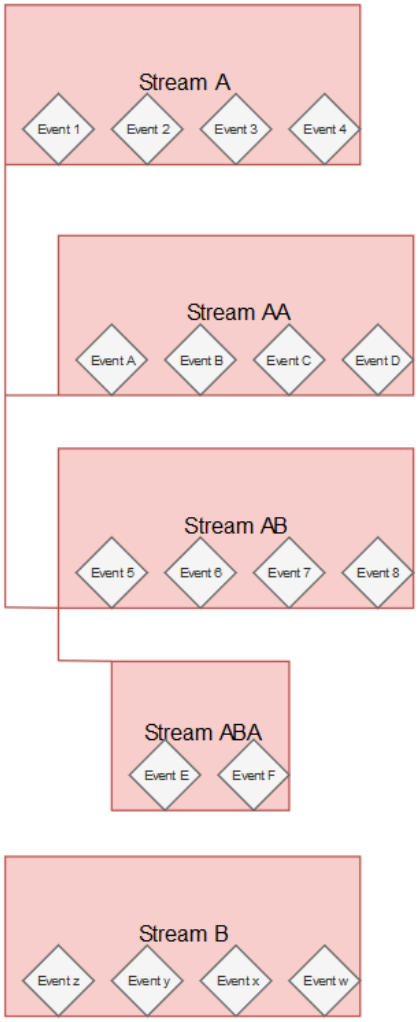
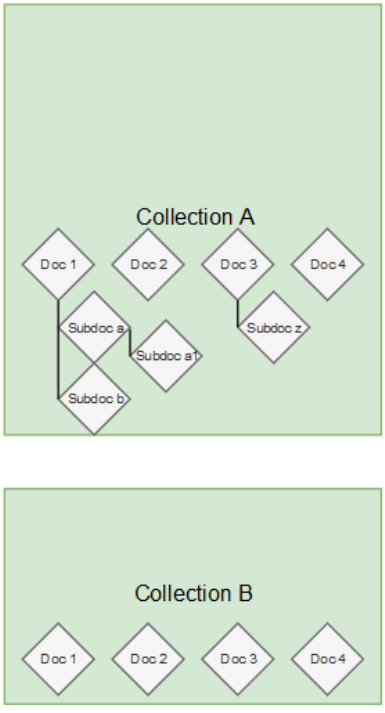
Pryv	MongoDB
	

Tableau 1: Comparaison structure de Pryv et MongoDB

3.3. Les autorisations

Lorsque l'on veut partager ses données avec des personnes tierces, Pryv nous permet de générer un token. Il s'agit tout simplement d'une clé qui est liée à un utilisateur et nous autorise l'accès à un contenu. Ce token est caractérisé par un « appId » correspondant à son nom, ainsi que les permissions qu'il donne. Ces dernières ont les propriétés suivantes :

- streamId : Identifiant du stream
- defaultName : Nom du stream
- level : Niveau d'autorisation
    - Read : lecture seule
    - Contribute : lecture et ajout
    - Manage : lecture, ajout et suppression

Voici ci-dessous un exemple d'une requête pour la génération d'un token pour l'application « my-app » avec une lecture seule pour le stream « Journal ».

```
{
  "requestingAppId": "my-app",
  "requestedPermissions": [
    {
      "streamId": "diary",
      "level": "read",
      "defaultName": "Journal"
    }
  ],
  "languageCode": "fr"
}
```

Il faut relever qu'il est également possible de fixer une durée de validité limitée pour un token, très utile lorsque l'on veut donner des accès temporaires.

4. Besoins de la DAunit

Les besoins de la DAunit ont été définis avec Mme Anne-Laure Kaufmann, collaboratrice de la DAunit et M. Jean-Paul Calbimonte Pérez, collaborateur à l'institut d'informatique de la Hes-so à Sierre qui est le PO dans ce projet. Nous distinguerons deux rôles principaux quant à l'utilisation de notre interface.

4.1. Administrateur

Le but premier de cette interface s'articule autour de la simplification et de l'exploitation maximale des fonctionnalités proposées par Pryv. L'administrateur aura dans un premier temps la possibilité d'avoir une vue d'ensemble sur les utilisateurs enregistrés. S'agissant surtout de collectes de données au niveau médical, il est important de grouper ces utilisateurs dans des projets de recherche afin de maintenir une certaine structure.

Ainsi, un administrateur aura la possibilité de créer un projet, d'y ajouter des utilisateurs, de leur demander l'accès à leurs données pour finalement exporter les résultats et permettre la suite de la recherche.

De cette manière, le responsable aura une vue d'ensemble sur l'état des autorisations émises par les patients qui participent à la recherche.

4.2. Utilisateurs Pryv

Les utilisateurs Pryv correspondent aux patients faisant partie du projet de recherche dans notre cas. Ceux-ci n'auront que peu d'interactions avec notre solution, car celle-ci cible principalement les administrateurs.

Tout d'abord, l'utilisateur pourra créer un nouveau compte afin de s'enregistrer sur l'instance Pryv déployée à l'Hes-so. Ensuite, il aura également la possibilité de créer, afficher et supprimer ses tokens. La question relative à l'autorisation d'un projet reste encore en suspens car l'idéal serait d'automatiser le processus au maximum. En effet, les personnes consentant ces projets ne seront pas forcément à l'aise avec la technologie, d'où la nécessité de réduire les actions requises par ces utilisateurs.

Après analyse, nous partons sur l'idée d'envoyer un mail automatique au patient dès lors qu'il est ajouté à un projet. Celui-ci contiendrait un lien menant directement à notre solution et guidant l'utilisateur pour nous accorder l'accès. Ainsi, la seule interaction serait la connexion à Pryv qui générerait le nouveau token et notre service s'occuperait ensuite de l'enregistrer au niveau du projet.

Une interaction via une application mobile est aussi prévue dans le projet à des fins démonstratives. Les fonctionnalités seront définies plus tard dans le projet en fonction du temps restant à notre disposition.

4.3. Le Product backlog

Comme nous utilisons la méthodologie Scrum tout au long de notre projet, nous avons défini un product backlog avec toutes les user stories. Au départ de ce travail, nous relevons un total de 95 story points, mais celui-ci peut évoluer tout au long du projet.

Ce document est joint en annexe I de ce rapport.

4.4. Mockups

Maintenant que nous avons défini tous nos cas d'utilisation, nous réalisons nos mockups. L'objectif est d'avoir une idée claire de l'interface que nous voulons développer et par conséquent les valider avec notre PO avant de débuter la phase de développement le 3 juin.

Nous avons à notre disposition plusieurs outils pour le faire, mais nous décidons de les modéliser directement avec du HTML et du CSS statique. Comme nous avons déjà une idée assez précise de notre projet, nous trouvons cette démarche très intéressante car ce choix nous permet d'avoir une bonne base utilisable une fois ces prototypes validés.

Voici ci-dessous la présentation d'un point de vue côté patient tout d'abord, puis sous l'angle de l'administrateur. Par ailleurs, la totalité de ces prototypes est disponible en annexe II.

4.4.1. Le patient

Sur le site de Pryv, nous pouvons retrouver la documentation concernant la requête à envoyer au serveur Pryv afin de créer le nouvel utilisateur. Nous pouvons remarquer que nous avons quatre informations spécifiques à l'utilisateur : l'adresse mail, le nom d'utilisateur, le mot de passe et le lieu où les données seront stockées. De ce fait, la page d'enregistrement sera un simple formulaire comme suit :

Figure 6: Page d'enregistrement

4.4.2. Administrateur

Voici, ci-dessous, une vue de la partie administrateur, il s'agit ici du détail d'un projet. On peut noter les détails d'un projet ainsi qu'une liste des patients faisant partie de ce dernier. De plus, on peut retrouver la fonctionnalité finale de cette recherche, soit la possibilité d'exporter les données collectées au cours de l'étude.

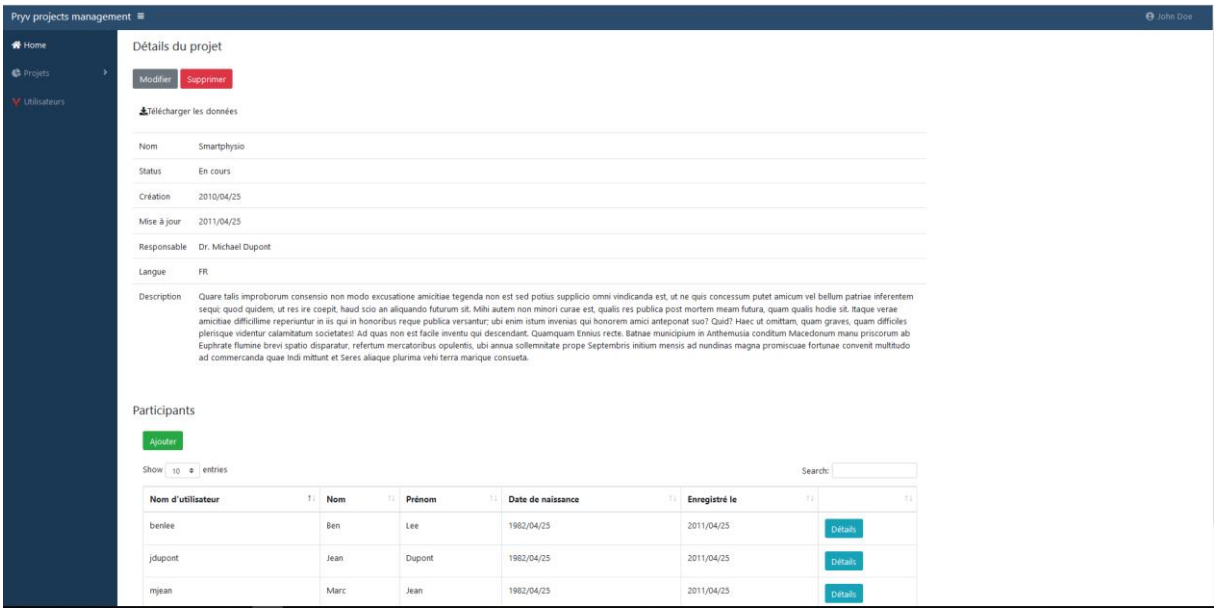


Figure 7: Page d'administration

Les autres prototypes sont très similaires à cette vue et sont disponibles à l'annexe II.

4.4.3. Application mobile

Les besoins ainsi que les mockups de l'application mobile seront élaborés au cours du chapitre 9 de ce rapport.

5. Analyse d'architecture et de technologie

Avant de débuter la phase pratique, nous devons dans un premier temps définir la technologie que nous allons utiliser. La solution de Pryv est basée sur des requêtes API REST ce qui nous offre un large choix pour le développement de notre solution.

Mise à part l'interaction avec l'instance de Pryv, nous avons besoin d'une base de données propre à notre gestion de projet. À cet égard, il nous faut un service backend pour communiquer avec notre base de données, d'où le besoin d'un choix très important quant à l'architecture et la technologie de notre solution.

Nous allons dans un premier temps analyser les différentes possibilités d'architecture de notre application. Nous continuerons ensuite avec un comparatif des différentes technologies à notre disposition. Pour terminer, nous choisirons notre système de gestion de base de données.

5.1. Architecture

Nous avons le choix entre deux possibilités, la première consiste à avoir un rendu côté serveur en utilisant un framework MVC (Model view controller). La deuxième possibilité porte sur une architecture orientée service avec un rendu côté client en utilisant un framework frontend.

5.1.1. Application MVC avec rendu côté serveur

Cette première alternative nous permet d'avoir la totalité de notre solution sur un serveur. Ainsi, lorsque le client fait une requête au serveur, notre application génère la page demandée puis la renvoie au client. De cette manière, le serveur peut réaliser les requêtes de façon autonome directement au serveur Pryv. Cependant, il reste envisageable d'avoir des appels depuis le client directement.

Voici ci-dessous un schéma illustrant notre architecture :

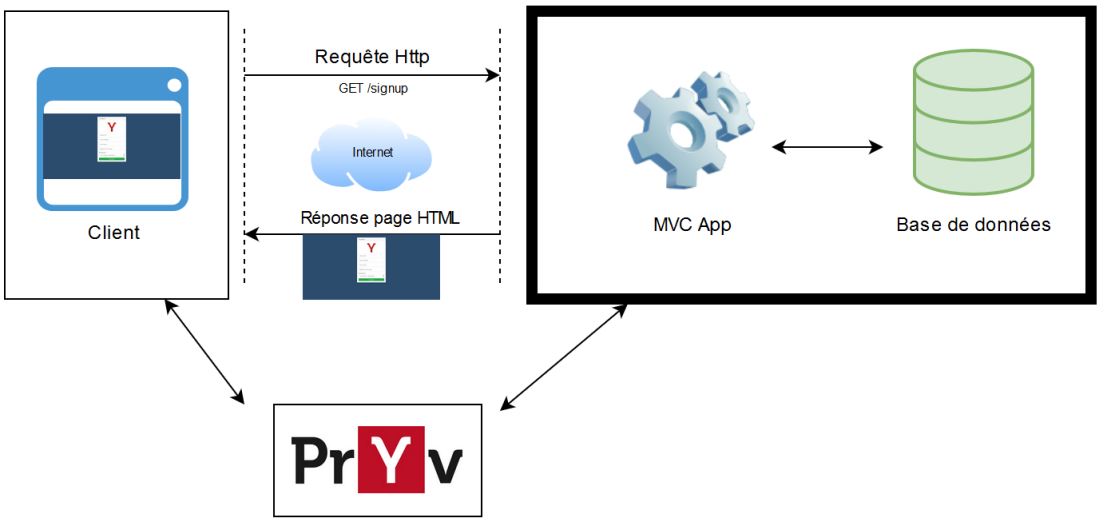


Figure 8: Application MVC avec rendu côté serveur (Réalisé sur draw.io)

Nous pouvons observer sur la droite les deux composants de notre application. L'application MVC communique avec notre base de données ainsi qu'avec le serveur Pryv. De l'autre côté, nous avons le client qui fait des requêtes à l'application ainsi qu'à Pryv.

Ce choix nous permet d'avoir une certaine isolation entre le client et l'application mais requiert plus de charge de travail pour le serveur.

5.1.2. Application avec un rendu côté client

La deuxième possibilité est orientée service, c'est-à-dire que nous aurons deux applications dans notre cas. La première va gérer toute la partie affichage et interaction avec l'utilisateur alors que la seconde va s'occuper de la partie logique métier. Cette solution permet d'avoir une certaine modularité. Par conséquent, il devient possible de réutiliser une application et de l'intégrer à d'autres projets.

Voici un schéma illustrant cette architecture :

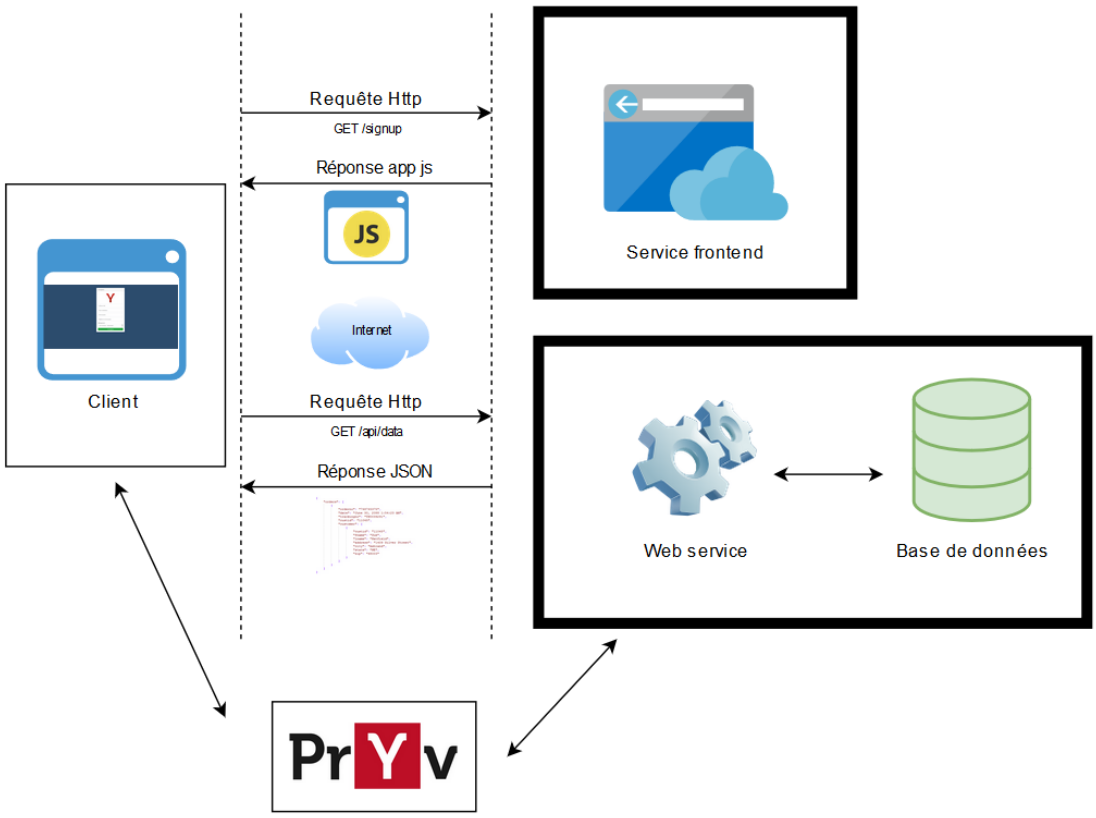


Figure 9: Application avec rendu côté client (Réalisé sur draw.io)

Nous avons un service pour l'interface utilisateur qui sera envoyé au client afin d'afficher le visuel et notre service backend pour la partie logique qui s'occupera, quant à lui, des accès à la base de données. Ainsi, lorsque le client navigue sur l'interface, l'application fera des requêtes à l'autre service pour récupérer les données brutes ce qui permettra au client de générer l'affichage. De cette manière, la logique de l'affichage se fera du côté du client contrairement au rendu côté serveur qui renverra l'affichage dans son entièreté. Cela réduit considérablement la charge de travail du serveur.

Un service frontend comme celui-ci nous permet d'avoir un contenu très dynamique et améliore considérablement l'expérience utilisateur. Cependant, il faut soulever qu'une telle architecture requiert le développement de deux services et plus de travail au niveau de la sécurité. En effet, si le service backend est disponible via un point API il faut le sécuriser au moyen d'une authentification et d'une gestion de token propre à ce service.

5.1.3. Choix de l'architecture

Finalement, notre choix se portera sur la première possibilité. De fait, une application web avec rendu côté serveur est moins complexe à réaliser et nous permet de nous concentrer davantage sur l'objectif final du projet. De plus, notre solution ne requiert pas un grand dynamisme des pages et le nombre d'utilisateurs sera moindre.

5.2. Framework de développement

Un framework est un ensemble de composants logiciels qui sert de structure pour la création d'une application web. Cela nous permet d'avoir une architecture standard et adaptée aux besoins du projet. Il est conçu pour faciliter le développement et permet de diminuer les coûts. Ainsi, la productivité des développeurs est augmentée. (Fournier, 2015)

Les spécificités de notre travail nous laissent le libre choix au niveau de la technologie. Néanmoins, il demeure très important de faire un comparatif parmi les solutions à notre disposition afin de choisir la plus adaptée à nos besoins. C'est pourquoi nous avons sélectionné quatre différents frameworks que nous allons analyser selon les six critères suivants :

- Connaissances : Nos connaissances personnelles
- Communauté : Communauté disponible sur internet
- Testabilité : Facilité à intégrer des tests
- Licence : Prix de la licence
- Déploiement : Facilité à déployer
- Documentation : Quantité et qualité de la documentation en ligne.

Les notes sont attribuées de 0 à 5, avec un 0 on ne répond pas du tout au critère tandis qu'un 5 répondra totalement au critère. Cette échelle nous permet de différencier les possibilités de manière significative.

5.2.1. Express avec Node.js

Node.js est un environnement open source, crée par Rayan Dahl en 2009, qui exécute du code JavaScript côté serveur. Il est actuellement basé sur le moteur JavaScript V8 développé par Google. (Node.js, s.d.). Comme il est orienté événement et s'exécute de manière asynchrone, il permet de développer des applications très performantes même en cas de grand accès à la base de données par exemple. Il offre également accès à NPM (Node Package Manager), le gestionnaire de paquet officiel de Node.js. Il s'agit d'un outil permettant d'accéder à des milliers de librairies open source qui le rend très modulaire et permet ainsi de s'adapter à chaque projet.

Figure 10: Node.js et Express (Projects Plaza, 2018)

Le framework Express, disponible sur NPM, est un outil très puissant sur Node.js. En effet, il nous permet d'être plus haut niveau en venant rajouter une couche supplémentaire permettant de gérer plus facilement les routes notamment. De plus, il offre la possibilité d'utiliser divers outils pour les templates comme Pug ou Handlebars par exemple. En conclusion, il s'agit d'une infrastructure web minimaliste et flexible qui apporte un ensemble de fonctionnalités sans masquer la force et la robustesse de Node.js. (Express.js, s.d.)

Nos connaissances personnelles sont plutôt solides car nous avons eu l'occasion de l'utiliser à plusieurs reprises au cours de notre cursus. En outre, la communauté est grandissante et nous pouvons y retrouver beaucoup d'aides sur les forums.

Connaissances	Communauté	Testabilité	Licence	Hébergement	Doc
4	3	4	5	3	4

Tableau 2: Résultat Express

5.2.2. Laravel avec PHP

PHP est un langage Open source qui permet de développer principalement des applications web. Il s'agit d'un langage de script conçu en 1994 par Rasmus Lerdorf qui était à ses débuts une librairie C[4]. Les pages PHP contiennent directement des fragments HTML qui seront ensuite interprétés par le serveur web. Mis à part son but de langage script côté serveur, il permet également de créer des scripts exécutables en ligne de commande ainsi que des applications graphiques (The PHP Group, s.d.). Sa maturité et sa popularité en font un des pionniers actuels dans le domaine. De fait, il est utilisé par environ 80 % des sites web.

Figure 11: Laravel (Laravel, s.d.)

Le framework Laravel est actuellement le plus connu et le plus utilisé dans les solutions sous PHP. En effet, il regroupe les meilleures librairies pour chacune de ses fonctionnalités ce qui lui a permis d'acquérir une grande popularité parmi ses concurrents. (Herrenschnieder, 2017). De base, il intègre déjà de base beaucoup de fonctionnalités telles que l'authentification, la gestion des routes, etc. … De plus les commandes de PHP artisan se révèlent très utiles pour les migrations des tables dans la base de données. Il possède également son propre système de template nommé « blade ».

Nous avons eu l'occasion de travailler une fois sur ce framework et la prise en main y est très rapide. Il s'agit d'un framework très complet qui permet un développement rapide.

Connaissances	Communauté	Testabilité	Licence	Hébergement	Doc
2	4	4	5	3	4

Tableau 3: Résultat Laravel

[4] C : Langage de programmation de bas niveau

5.2.3. Django avec Python

« Django est un framework objet de développement web écrit en Python qui a pour objectif de rendre le développement web simple et rapide.

Python est un excellent langage de programmation, déployant de nombreux paradigmes (programmation objet, fonctionnelle, etc.). De ce fait Django bénéficie de toute la souplesse et de la puissance de Python. Django bénéficie également de toute la richesse des librairies Python dont certaines sont exceptionnelles. » (Fournier, 2015)

Figure 12: Python et Django (Shuup, 2017)

Ce framework possède, comme Laravel, beaucoup de fonctionnalités qui permettent d'accélérer le développement et d'automatiser énormément de processus comme la génération de formulaire ou l'authentification notamment. Contrairement à ses concurrents, il utilise une architecture MVT (Modèle, Vue, Template). Django va s'occuper lui-même de gérer la partie contrôleur et ainsi faciliter le développement.

Bénéficiant d'une très grande popularité, ce framework devient dès lors un choix très intéressant. Malgré cela, nous n'avons aucune expérience sur cette technologie. Il nous faudrait donc consacrer plus de temps de développement avant d'être parfaitement à l'aise à ce sujet

Connaissances	Communauté	Testabilité	Licence	Hébergement	Doc
0	5	4	5	3	4

Tableau 4: Résultat Django

5.2.4. C# avec .NET Framework

Tout d'abord, C# est un langage de programmation développé par Microsoft en 2002. Il permet aussi bien de créer des applications comme des logiciels de traitement de texte ou des applications web comme un site de e-commerce. Sa syntaxe est similaire à C++ ou Java qui sont des langages très utilisés. (openclassrooms, 2019)

Figure 13: C# et .NET (Dotnettutorials, 2018)

« Le .NET Framework est un environnement d'exécution managé pour Windows qui fournit divers services à ses applications actives. Il comporte deux composants principaux : le Common Langage Runtime (CLR) qui est le moteur d'exécution gérant les applications actives, et la bibliothèque de classes .NET Framework qui fournit une bibliothèque de code testé réutilisable que les développeurs peuvent appeler à partir de leurs propres applications. » (Microsoft, 2019)

Nous avons quelques connaissances et expériences sur ce framework. Malgré cela l'hébergement nécessite un serveur Microsoft et le rend plus limité.

Connaissances	Communauté	Testabilité	Licence	Hébergement	Doc
3	3	3	5	1	4

Tableau 5: Résultat .NET

5.2.5. Choix du framework

À la vue des notes attribuées pour chacun des frameworks, nous arrivons avec express à la première place suivi de Laravel, Django puis .Net. Nos critères ne permettent pas de départager clairement les trois premiers frameworks, c'est surtout le critère de connaissance qui va nous donner la possibilité de faire un choix.

Django et Laravel sont des technologies très complètes que nous envisageons sérieusement de choisir. Cependant, le temps de développement à notre disposition et très court et ne nous permet pas de rajouter une charge supplémentaire d'apprentissage. C'est pourquoi notre choix final se portera sur Express.

	Connaissances	Communauté	Testabilité	Licence	Hébergement	Doc	Total
Express	4	3	4	5	4	4	24
Laravel	2	4	4	5	3	4	22
Django	0	5	4	5	3	4	21
.NET	3	3	3	5	1	4	19

Tableau 6: Agrégation résultats framework

5.3. Base de données

Afin de stocker nos données de projet, nous avons besoin d'une base de données. C'est pourquoi nous souhaitons également faire un choix parmi deux solutions.

5.3.1. MySQL

MySQL est un système de gestion de base de données relationnelle basé sur le langage SQL et fonctionne sur quasiment tous les systèmes d'exploitation. Il est déployé sur plus de dix millions de serveurs dans le monde et ainsi le leader mondial du marché des bases de données. (PHP sources, s.d.) Sa syntaxe est très simple et permet un apprentissage rapide.

MySQL	
Avantages	Inconvénients
• Très rapide en lecture • Haute capacité de stockage	• Structure définie • Changements nécessitent plus de travail

Tableau 7: Avantage - Inconvénients MySQL

5.3.2. MongoDB

MongoDB est une base de données orientée document. C'est-à-dire que les données sont stockées sous une forme proche d'un objet JavaScript. Une collection est un ensemble de documents comparables à une table en relationnelle. (Scenari-Community, s.d.)

MongoDB	
Avantages	Inconvénients
• Pas de schéma requis • Flexible en cas de modification • Répartition des données	• Syntaxe parfois complexe • Exposé à la redondance

Tableau 8: Avantage - Inconvénients MongoDB

5.3.3. Choix de la base de données

Notre choix se portera finalement sur MongoDB, la raison principale de cette décision est la flexibilité. En effet la structure de nos documents n'est pas prédéfinie et nécessite donc d'être favorable aux changements. Comme les champs d'un projet risquent fortement d'être modifiés au cours du développement, cette solution semble la plus adaptée. Malgré le fait que nous n'avons pas beaucoup d'expérience avec cette technologie, il s'agit d'un système de gestion de base de données facile à prendre en main et ne devrait donc pas poser de problèmes.

5.4. Conclusion

Pour résumer les choix technologiques faits durant ce chapitre, voici un petit rappel :

- Une architecture avec un rendu côté serveur
- Une application Node.js avec le framework Express.js
- Une base de données mongoDB

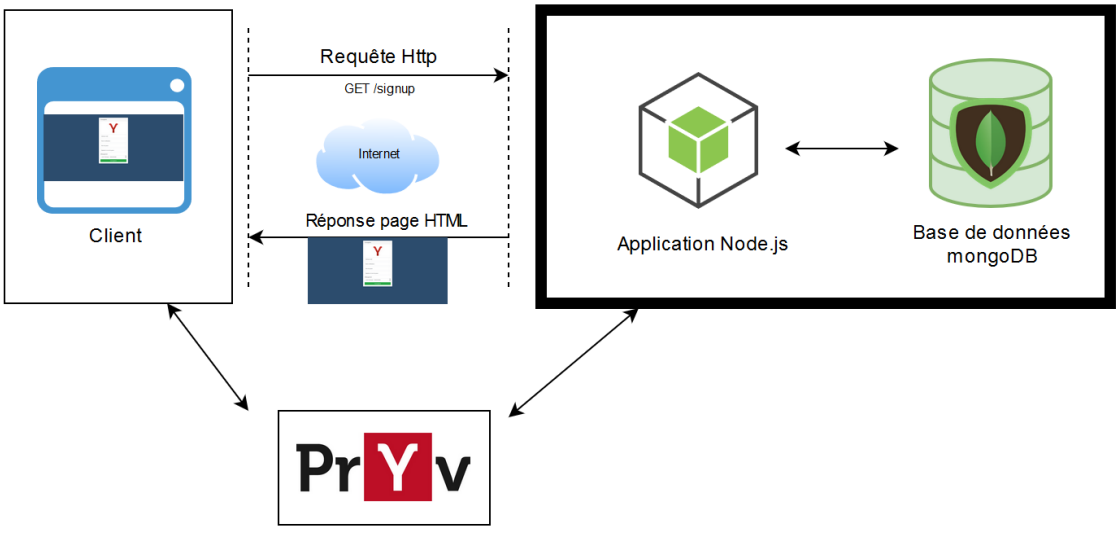


Figure 14: Architecture finale de la solution (Réalisé sur draw.io)

6. Mise en place de l'environnement de développement

6.1. Environnement de développement intégré

Pour notre développement, nous décidons d'utiliser l'environnement WebStrom faisant parti de la suite intelliJ. Il s'agit d'un outil très puissant qui offre énormément possibilités facilitant ainsi le développement. De plus, il permet d'avoir une licence étudiant gratuite.

Nous utiliserons la version 2019.1.2

6.2. Test

Afin d'avoir un prototype qui pourrait éventuellement être mis en production, il est impératif de couvrir suffisamment de tests unitaires. À cet égard, WebStorm offre plusieurs outils mais nous choisissons d'utiliser Mocha un des frameworks de test open source qui est le plus populaire sur Node.js.

Nous utiliserons la version 6.1.4

6.3. Gestion de versions

Pour avoir un suivi ainsi qu'une sauvegarde de notre développement, nous avons besoin d'un logiciel de gestion de versions. Son concept est très simple, il sauvegarde la totalité des modifications apportées au projet et assure ainsi une chronologie de l'évolution du code.

Dans ce projet nous allons utiliser Github, ce VCS (Version Control System) nous offre beaucoup de fonctionnalités telles que la gestion de projet ou encore un système de suivi des bugs. De plus, notre statut d'étudiant nous offre la possibilité d'avoir des répertoires privés avec un compte gratuit.

6.4. Docker

Docker est un outil très performant qui permet de concevoir, tester et déployer des applications très rapidement en utilisant des containers. L'idée consiste à exécuter un programme dans un environnement isolé, semblable à une machine virtuelle, mais sans

disposer de tout le système d'exploitation. Voici ci-dessous un schéma illustrant la différence entre les containers de docker et les machines virtuelles. (Docker Inc., s.d.)

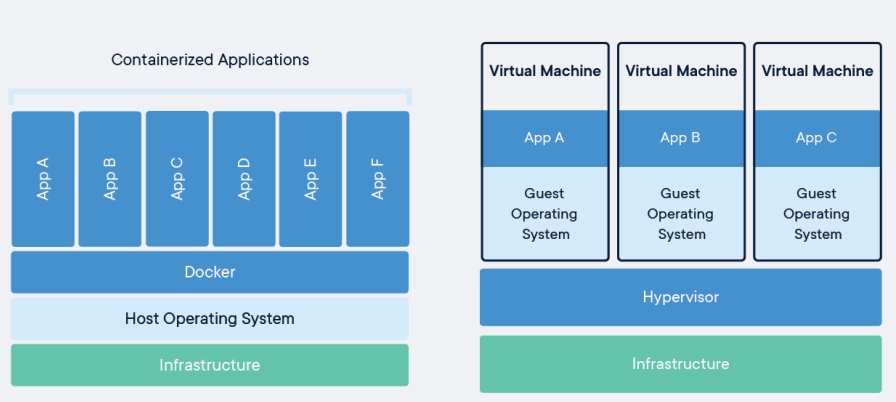


Figure 15 : Container vs Machine virtuelle (Docker Inc., s.d.)

De cette manière notre solution sera composée de deux containers, un pour l'application Node.js et l'autre pour notre base de données MongoDB. Vous trouverez le fichier « docker-compose.yml » qui contient toute la configuration de docker en Annexe III de ce document.

Notre solution sera déployée sur une instance docker installée sur un serveur de la Hes-so.

7. Sprint 1

Lors du sprint planning du 3 juin avec Michael Ignaz Schumacher et Jean-Paul Calbimonte Perez, nous avons décidé de prendre 10 user stories pour un total de 31 story points. L'objectif pour ce sprint est d'avoir une interface pour l'enregistrement des utilisateurs sur Pryv avec la possibilité de générer des tokens et la création/modification d'un projet.

De plus, nous décidons de contacter Pryv afin d'organiser une rencontre pour leur montrer le travail réalisé jusqu'ici et avoir quelques conseils sur le développement.

7.1. L'enregistrement sur Pryv

Comme nous disposons déjà d'un mockup avec les champs requis par Pryv, nous décidons de le réutiliser tel quel dans notre application. Les hébergements disponibles sont chargés dynamiquement en fonction de ceux disponibles. C'est pourquoi nous devons faire une requête au serveur Pryv comme ci-dessous afin de récupérer les différents choix possibles.

- `GET : https://reg.pryv.hevs.ch/hostings`

Et nous avons la réponse suivante :

```
"regions": {
    "europe": {
        "name": "Europe",
        "localizedName": {
            "fr": "Europe"
        },
        "zones": {
            "switzerland": {
                "name": "Switzerland",
                "localizedName": {
                    "fr": "Suisse"
                },
                "hostings": {
                    "core1": {
                        "url": "https://hosting-provider.com",
                        "name": "Hosting provider name",
                        "description": "Hosting provider slogan",
                        "localizedDescription": {},
                        "available": true
                    }
                }
            }
        }
    }
}
```

Il s'agit d'un JSON (JavaScript Object Notation) structuré avec la région, la zone et l'hébergeur. Pour l'enregistrement de l'utilisateur, nous avons besoin des clés enfants du nœud « hostings », dans le cas de Hes-so un seul est disponible (core1).

La page d'enregistrement se trouve sous le chemin/signup de notre application, qui une fois les hébergeurs passés en paramètre, renvoie la page d'enregistrement au client.

Ensuite, le formulaire est renvoyé au serveur qui exécutera lui-même la requête suivante au serveur Pryv :

- POST : https://reg.pryv.hevs.ch/user
- Header: Content-Type: application/json
- Data: {

```
"appid": "Pryv-Hes-so",
"hosting": "core1",
 "username ": "vlami",
 "password": "123456",
 "email": "vlado.mitrovic@gmail.com",
 "invitationtoken": "enjoy"
}
```

Si notre création s'est bien déroulée, nous redirigerons l'utilisateur vers la page de départ avec un message de confirmation.

7.2. Gestion des tokens

Afin de gérer ses tokens, l'utilisateur dispose de quelques fonctionnalités en accédant au chemin/token. Cette partie nous permettra de voir les autorisations accordées et de la supprimer notamment. Cette partie est très importante d'un point de vue patient, il s'agit pour lui de l'unique moyen d'avoir une vue d'ensemble sur les autorisations qu'il a accordées. Depuis cette zone, nous ajoutons également la possibilité de génération de tokens totalement indépendants de notre solution.

7.2.1. Liste et informations

Tout d'abord, il doit s'identifier avec son nom d'utilisateur et mot de passe Pryv afin de récupérer un token. Pour cela, nous faisons une requête avec ces paramètres, et nous devons aussi ajouter le champ « Référer » en entête.

- POST : https://{*nom d'utilisateur*}.pryv.hevs.ch/auth/login
- Header:

  Content-Type: application/json

  Referer: https://{nom d'utilisateur}.pryv.hevs.ch
- Data: {

```
"appid": "Pryv-Hes-so",
"username": "vlami",
"password": "123456"
}
```

Et nous obtenons la réponse suivante :

```
{
    "token": "cjyice46200100aqp21onm10b",
    "preferredLanguage": "fr",
    "meta": {
       "apiVersion": "1.2.18",
       "serverTime": 1564038923.165
    }
}
```

Nous stockons la paire nom d'utilisateur et token dans la session de l'utilisateur, puis le redirigeons vers le chemin/token/accesses. À partir de là le serveur fait une requête au serveur pryv pour récupérer la liste des tokens pour ce nom d'utilisateur.

- GET : https://{*nom d'utilisateur*}.pryv.hevs.ch/accesses
- Header : Authorization : {token récupéré lors de l'étape précédente}

Et nous avons en réponse un JSON contenant une liste des tokens incluant les détails :

```
{
    "accesses": [
        {
            "token": "cjy8upvfu000i0aqpduejwni9",
            "name": "Pryv-Hes-so",
            "type": "personal",
            "deleted": null,
            "created": 1563465042.286,
            "createdBy": "system",
            "modified": 1564039065.815,
            "modifiedBy": "system",
            "lastUsed": 1564039388.628,
            "id": "cjy8upvfy000j0aqp3mk0om40"
        },
        …
    ],
    "meta": {
        "apiVersion": "1.2.18",
        "serverTime": 1564039426.964
    }
}
```

Ces informations nous donneront la possibilité d'afficher une liste de nos tokens actifs ainsi que les détails correspondants.

7.2.2. Génération d'un token

Pour la génération de nouvelles autorisations, nous décidons de la créer en utilisant le token récupéré précédemment lors de l'authentification. La requête est similaire à celle pour récupérer la liste, mais dans ce cas-ci, il s'agit d'un POST avec les données d'autorisations.

- POST : https://{nom d'utilisateur}.pryv.hevs.ch/accesses
- Header : Authorization : {token récupéré lors de l'étape précédente}
- Data: {

```
"name": "Pryv-Hes-so The app",
"permissions" : [{
                    "streamId": "theApp",
                    "defaultName": "TheApp",
                    "level": "read"
                    }]
}
```

Cette requête nous renvoie en résultat un token avec tous les détails le concernant. Malgré le fait que cette fonction nous génère une autorisation valide, nous n'allons finalement pas l'utiliser. En effet, à la suite de la rencontre avec les développeurs de Pryv, on nous déconseille fortement de faire de cette manière. Ceci ne va pas fonctionner si le stream en question n'existe pas et le token d'authentification n'est pas prévue pour ce type de cas.

7.2.3. Génération du token avec poll

Cette méthode plus sûre nous offre également un autre avantage. Nous pouvons en effet avoir un seul lien qui génère automatiquement un nouveau processus de création de token pour un projet donné. De plus, si le stream n'existe pas chez l'utilisateur celui-ci est automatiquement créé. La requête est très similaire à la précédente méthode, mais nous l'envoyons vers le point de terminaison « reg » :

- POST : https://reg.pryv.hevs.ch/accesses
- Data: {

```
"requestingAppId": "Pryv-Hes-so The app",
"requestedPermissions": : [{
                          "streamId": "theApp",
                          "defaultName": "TheApp",
                          "level": "read"
                          }],
"returnURL" :  "",
"languageCode": "fr"
}
```

En nous avons la réponse suivante :

```
{
    "status": "NEED_SIGNIN",
    "code": 201,
    "key": "SMi2vGnrK286Qkkz",
    "requestingAppId": "Pryv-Hes-so The app",
    "requestedPermissions": [
        {
            "streamId": "theApp",
            "defaultName": "TheApp",
            "level": "read"
        }
    ],
    "url":
       "https://sw.pryv.hevs.ch/access/access.html?lang=fr&key=SMi2vGnrK286Qkkz&requestingA
       ppId=Pryv-Hes-so The
       app&returnURL=https%3A%2F%2Freg.pryv.hevs.ch%2Faccess&domain=pryv.hevs.ch&registerUR
       L=https%3A%2F%2Freg.pryv.hevs.ch&requestedPermissions=%5B%7B%22streamId%22%3A%22theA
       pp%22%2C%22defaultName%22%3A%22TheApp%22%2C%22level%22%3A%22read%22%7D%5D",
    "poll": "https://reg.pryv.hevs.ch/access/SMi2vGnrK286Qkkz",
    "returnURL": "https://reg.pryv.hevs.ch/access",
    "poll_rate_ms": 1000
}
```

Avec ces informations, nous avons une URL qui pointe directement vers le serveur Pryv où l'utilisateur pourra se connecter. Une fois authentifié, il pourra autoriser l'accès afin qu'un nouveau token soit généré. Notre serveur fera des requêtes sur l'adresse « poll » avec un intervalle de 1000 millisecondes et ainsi il pourra enregistrer l'autorisation une fois acceptée.

```
{
    "status": "ACCEPTED",
    "username": "vlami",
    "token": "cjyicnriq00140aqpqbn0mq9g",
    "code": 200
}
```

On peut observer ci-dessus le résultat de l'adresse « poll » avec le statut, le nom d'utilisateur, le token et le code HTTP de la réponse. En cas de refus, la réponse suivante est retournée :

```
{
    "status": "REFUSED",
    "reasonID": "REASON_UNDEFINED",
    "message": "Access refused by user",
    "code": 403
}
```

7.3. Protection de la zone administrateur

Notre application web possède deux parties distinctes, d'une part nous en avons une accessible publiquement sans authentification tandis que l'autre est réservée aux administrateurs. Voici ci-dessous un schéma de notre structure de routage avec la partie qui requiert une identification en jaune.

```
-/                          index.js
-------/signup
-------/token
        ---------/accesses
-------/login
        ---------/admin      admin.js
        ---------/…
```

Par l'intermédiaire de Node.js, nous créons un middleware qui testera à chaque requête afin de s'assurer que l'utilisateur est bien connecté. De cette manière, la fonction « isAuthenticated » vérifiera la variable de session « authenticated ». Si l'utilisateur est connecté nous continuons, dans le cas contraire nous le redirigeons vers la page d'authentification.

Nous plaçons ce middleware au niveau de la déclaration de la route admin :

app.js

```
app.use('/admin',isAuthenticated, adminRouter);

//Check if the user is logged
function isAuthenticated(req, res, next){
    if(req.session.authenticated != true)
       res.redirect('/login');
    next();
}
```

Lorsque l'utilisateur se connecte et que les informations sont correctes, nous mettons à jour la variable de session afin de lui autoriser la partie administrateur et nous stockons également son adresse mail. Inversement, lors d'une déconnexion, la session est détruite. Nous supprimons donc toutes données stockées en session. Il faut noter que cette valeur est stockée côté serveur ce qui rend impossible toute manipulation non autorisée et ainsi une gestion sécurisée des accès.

7.4. Création d'un projet

Selon les besoins définis durant la phase d'analyse, nous avons décidé de stocker les champs suivants dans notre base de données lors de la création d'un nouveau projet :

- Nom du projet
- Responsable
- Description
- Statut du projet
- Langue
- Autorisations requises

L'intégralité de ces champs sera du texte simple sauf exception des autorisations qui s'apparenteront à une liste d'objet. À cela s'ajoute la liste des patients participant au projet, mais nous nous y attarderons lors du prochain sprint.

Une fois que le projet est enregistré, il apparaitra de la manière suivante dans la base de données :

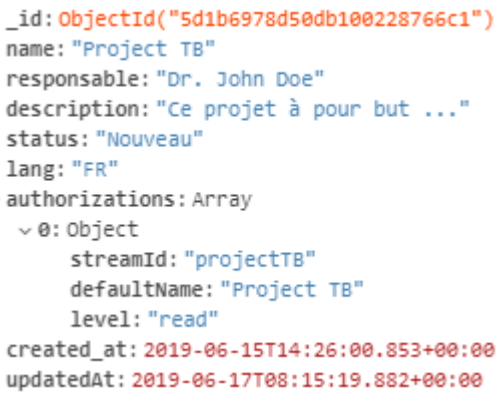


Figure 16: Structure d'un projet

Nous pouvons observer qu'un identifiant unique est généré automatiquement par mongoDB. Il est ainsi possible de retrouver un projet via ce champ. Par ailleurs, nous avons également le temps de création et de modification qui nous offrent des informations complémentaires pour avoir un certain suivi.

7.5. Rencontre avec Pryv

Nous fixons la rencontre directement le vendredi suivant le sprint planning. Accompagné de Jean-Paul Calbimonte Pérez, nous nous rendons à Morges et nous présentons nos mockups ainsi que le travail réalisé jusqu'à présent à Ilia Kebets, chef de projet et son équipe.

Plusieurs points sont soulignés, comme le processus de génération de token prévue au départ qui fait l'objet d'une grande discussion. En effet, notre idée de départ est incertaine et risque de poser quelques problèmes en cas de double autorisation. Plus spécifiquement, notre processus génère en réalité des sous-tokens à partir de celui reçu lors de l'authentification (voir 7.1.2) et ce token n'est pas prévu pour ça. Pour cette raison, il nous est conseillé d'utiliser le processus du « poll » (voir 7.2.3) beaucoup plus adapté à nos besoins.

De plus, nous apprenons que certaines fonctionnalités comme la suppression d'utilisateurs ou la gestion de token admin ne sont possibles qu'avec un accès direct en ligne de commande. C'est pourquoi ces fonctionnalités ne seront pas implémentées. Nous avons également l'information qu'il n'est pas possible d'avoir un historique quant à l'utilisation d'un token.

Ces quelques erreurs sur la compréhension du fonctionnement de Pryv, ainsi que tous les conseils reçus de la part de l'équipe, sont encourageants pour la suite de notre développement.

7.6. Sprint review

Lors de cette séance du 18 juin, avec Michael Ignaz Schumacher et Jean-Paul Calbimonte Perez, nous présentons le travail réalisé. Toutes les user stories sont acceptées. Nous relevons le fait que le projet avance plus vite que prévu. C'est pourquoi l'idée de faire une application mobile plus complète que celle prévue au départ est émise. Pour notre prochain rendez-vous, nous allons préparer les mockups de la version mobile.

8. Sprint 2

Nous décidons de prendre 6 user stories pour un total de 25 story points. Dans ce sprint, l'objectif principal consiste à faire le lien entre les utilisateurs et les projets pour permettre de faire un export des données collectées.

8.1. Lien entre patients et projets

Lors de la création d'un projet (voir 7.4), nous avions décidé de stocker les patients sous forme d'une liste d'objets. Cette dernière devait impérativement contenir le nom d'utilisateur du patient ainsi que le token d'autorisation pour les données du projet. C'est pourquoi nous allons stocker cette paire dans notre base de données.

En ce qui concerne l'interface, nous implémentons la fonctionnalité d'ajout d'un patient au niveau du détail d'un projet. Afin de la rendre plus ergonomique, nous ajoutons une autocomplétions à partir de la liste des utilisateurs disponible sur l'instance de Pryv. Une fois validé, nous ajoutons le patient avec « En attente » dans le champ token qui s'actualisera lors de l'autorisation par le patient.

Pour mieux comprendre ce processus d'ajout, nous exposons un diagramme avec les étapes de ce processus :

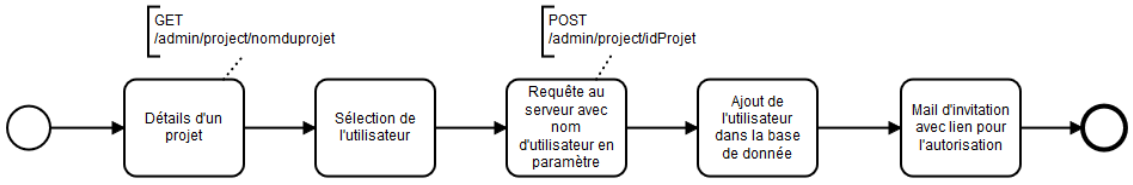


Figure 17: Processus d'ajout d'un utilisateur

Comme nous pouvons l'observer à la fin, un mail est envoyé à l'utilisateur sur l'adresse utilisée lors de l'enregistrement sur l'instance de Pryv. Celui-ci contient toutes les informations nécessaires pour l'accord des autorisations requises par le projet. Il est également possible de renvoyer ce mail dans le cas où un utilisateur aurait omis de suivre le processus.

8.2. Export

Dès qu'un projet de recherche est terminé, nous devons avoir la possibilité d'accéder à ces données afin de les étudier et de les traiter. Pour cela, nous devons dans un premier temps définir un format de fichier pour cet export. Comme nous utilisons déjà un framework JavaScript et que les données de Pryv sont récupérées au format JSON, ce choix de format nous semble plus qu'évident.

Ce fichier contient les détails d'un projet, ainsi que les données récupérées des patients sur le serveur Pryv. Dans le cas où un utilisateur n'aurait pas fait le processus d'autorisation ou l'aurait supprimé, nous afficherons un simple message dans notre fichier d'export. Un exemple de fichier généré est disponible en annexe IV.

8.3. Optimisation du poll token

Lors du premier sprint, nous avions déjà implémenté la génération de tokens (voir 7.2.3). Cette méthode fonctionnait plutôt bien, mais nous n'avions pas la possibilité de générer un message de confirmation étant donné que nous étions redirigés vers le serveur Pryv pour accepter l'autorisation. De plus, le poll se faisait côté serveur ce qui pouvait engendrer un trafic inutile si le processus était abandonné en cours. Ces points négatifs nous ont amenés à réfléchir à une autre solution pour ce processus.

Tout d'abord, le fait d'être redirigé sur le serveur Pryv nous faisait quitter la plateforme. Pour corriger cela, nous étudions la possibilité de générer un popup avec la fenêtre de connexion avec en arrière-plan un polling qui mettra à jour la page une fois l'autorisation acceptée. Finalement, nous décidons d'incorporer cette page de connexion en tant que « iframe » dans notre page car certains navigateurs peuvent bloquer le popup et donc nous freiner dans notre processus.

Comme nous restons sur notre page, nous mettons le polling côté client. Cela nous semble beaucoup plus logique car indépendant du serveur. Ainsi, une fois l'autorisation acceptée, le client fait un appel au serveur avec les informations nécessaires et est redirigé vers une confirmation.

L'implémentation de cette fonctionnalité est un succès et permet d'avoir un réel gain en ergonomie pour les patients. À la réception du mail, le patient arrive sur cette page avec une « iframe » de Pryv sur notre application :

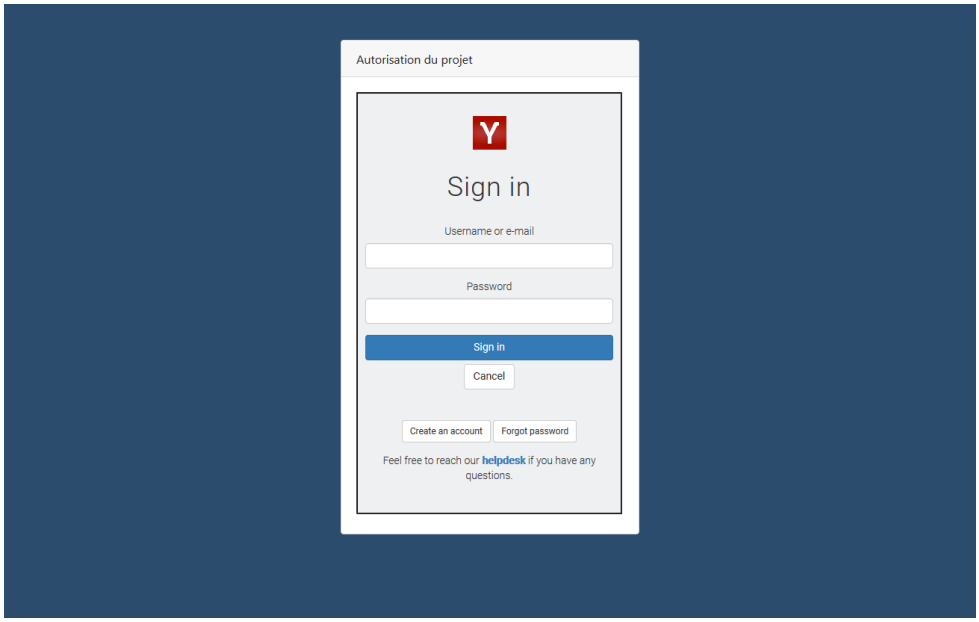


Figure 18: Authorisation avec "iframe"

8.4. Sprint review

Lors du sprint review du 2 juillet, toutes les user stories sont acceptées. Nous discutons sur le fait d'améliorer l'ergonomie, en ayant un système de notification en cas de suppression d'un token par l'utilisateur notamment. De plus, nous souhaitons rajouter des user stories afin d'avoir la possibilité de gérer les comptes ayant accès à la partie administrateur. Comme cette fonctionnalité n'était pas prévue dans le product backlog de départ, nous définissons nos nouvelles user stories. Elles sont évaluées à 23 story points au total.

Comme prévu lors de la dernière séance, nous proposons nos mockups (Annexe V) pour la partie mobile ainsi que les user stories correspondantes. Cette partie sera développée plus en détail dans un prochain chapitre. Un total de 27 story point est également rajouté au product backlog du projet.

9. Analyse de l'application mobile

Afin d'avoir un exemple concret pour utiliser notre plateforme de gestion de projet, nous avons besoin d'une application mobile. Après réflexion, nous décidons de faire une application générique adaptée à plusieurs types de projets.

9.1. Fonctionnalités

Nous voulons dans un premier temps définir les fonctionnalités nécessaires pour notre interface mobile. Comme il s'agit d'une solution standard très générique, et que nous n'avons pas beaucoup de temps à notre disposition, nous décidons de proposer les fonctions suivantes :

- Connexion/déconnexion sur le profil Pryv
- Affichages des projets dont l'utilisateur fait parti
- Formulaire avec possibilité d'ajout d'events dans un sprint
- Liste des précédents events
- Suppression d'events

9.2. Limitations

Comme il s'agit d'une application très générale, nous sommes limités au niveau des possibilités. Dans le cas d'un projet un peu plus complexe, comme plusieurs Stream, notre application ne pourra pas gérer cela. Cependant, le but premier de cette solution est d'avoir un moyen de présenter un exemple concret d'utilisation de notre plateforme de gestion de projets.

9.3. Choix technologique

Pour la réalisation de ce prototype, nous avons à notre disposition plusieurs choix au niveau de la technologie à utiliser. Parmi ces choix, il existe trois familles dans le développement d'applications mobiles :

9.3.1. WebApp

La première des solutions est une simple application web qui sera utilisée sur le navigateur du téléphone. Celle-ci va s'appuyer sur l'adaptation du site web à l'écran mobile afin de rendre une expérience utilisateur optimale. Elle offre notamment un accès multiplateforme qui permet l'accès sur différents terminaux avec une seule application. Par conséquent, une seule solution suffit à être accessible sur différent terminal.

Bien que cette solution puisse être intéressante, elle requiert un serveur web et donc plus de travail au niveau de la maintenance. Par ailleurs, nous n'avons pas la possibilité d'accéder à la mémoire interne du téléphone qui pourrait être très utile dans notre cas pour rester connecté par exemple.

9.3.2. Hybride

Une application hybride est développée afin d'être déployée sur différent système d'exploitation tel que IOS, Android ou Windows phone. Cette solution est très utile en vue de réduire les coûts de développement. Elle est développée en utilisant une technologie similaire à une application web mais elle est englobée dans une application native. Pour obtenir un bon résultat, il faut trouver un juste milieu entre les systèmes visés, car le fait de privilégier un système d'exploitation diminuera inévitablement l'ergonomie des autres cibles.

De plus, certains projets peuvent rapidement devenir complexes, voire impossibles à réaliser si l'on utilise des composants spécifiques au téléphone comme la caméra. Malgré cela, notre projet relève d'une faible complexité donc une telle solution reste envisageable.

9.3.3. Native

Le développement natif est, comme son nom l'indique, basé sur le langage natif du système d'exploitation. Malgré le fait que cela nous amène à faire un projet propre à chaque système d'exploitation, il s'agit de la solution la plus optimale. Android et IOS mettent à disposition leur environnement de développement, respectivement Android Studio et Xcode. Il faut relever que le développement sous IOS requiert un système d'exploitation Mac OS.

10. Sprint 3

Nous décidons avec notre PO de prendre un total de 40 user stories. Un nombre un peu plus conséquent que pour les sprints précédents, mais ceci se justifie par un nombre de jours de travail plus important. L'objectif de ce sprint s'articule autour de la possibilité de gérer les utilisateurs qui ont accès à la plateforme et d'une application mobile qui permettra l'insertion des données sur Pryv. Nous demandons également que les certificats SSL soient mis à jour car il s'agit d'une condition indispensable afin de communiquer avec l'API de Pryv depuis notre interface mobile.

Le sprint se terminera le 16 juillet marquant ainsi la fin de la partie développement. Nous décidons tout de même de fixer une séance entre-deux pour faire un point sur l'avancée du développement et avoir, de ce fait, la possibilité d'y apporter des corrections.

10.1. Gestions des comptes utilisateurs

Nous souhaitons ajouter ici la possibilité de créer de nouveaux accès à la plateforme. Pour cela nous définissons 2 rôles :

- Le super administrateur
- Le chef de projet

Le super administrateur s'apparentera à la personne externe au projet de recherche. Sa tâche consistera à gérer les comptes utilisateurs de la plateforme. En ce qui concerne les chefs de projets, ils seront là pour mener la recherche. Ils bénéficieront ainsi d'un accès à l'ensemble des données des projets dont ils prennent part. Le rôle de l'utilisateur authentifié sera stocké au moment de la connexion dans la variable de session.

Cette nouvelle fonctionnalité requiert une petite analyse quant au choix de l'implémentation. En effet nous souhaitons comparer deux possibilités pour ajouter un nouveau compte :

- L'inscription du chef de projet sur la plateforme puis le super administrateur valide le compte de l'utilisateur

- Création du compte par le super administrateur puis envoi du mot de passe autogénéré sur le mail du chef de projet

Méthode 1 : Inscription puis validation par le super administrateur	
Avantages	Inconvénients
• Charge de travail • Validation rapide	• Tout le monde peut s'inscrire • Risque d'erreur dans la validation

Tableau 9: Comparaison des méthodes de création de comptes (Méthode 1)

Méthode 2 : Ajout par le super administrateur, puis mail avec accès	
Avantages	Inconvénients
• Diminution du risque d'erreur • Moins complexe à implémenter	• Saisie par le super admin

Tableau 10: Comparaison des méthodes de création de comptes (Méthode 2)

Finalement, nous pouvons constater que les deux méthodes sont très similaires. Chacune dispose de qualités et de défauts. Cependant, nous portons notre choix sur la deuxième possibilité. En effet, elle relève d'une plus grande sûreté car le super administrateur doit lui-même entrer l'adresse mail de l'utilisateur. De plus, la complexité liée à l'implémentation de cette solution est moindre.

Le super administrateur aura la possibilité d'ajouter l'adresse mail d'un nouvel utilisateur en y spécifiant sa fonction. En parallèle, nous générerons un mot de passe aléatoire de 10 caractères qui sera transmis par mail une fois le processus complété.

10.1.1. Nouvelle route

Afin d'offrir une interface pour gérer les utilisateurs, nous décidons d'ajouter une nouvelle route à la structure de notre routage. Celle-ci viendra s'ajouter en extension de la route admin et se dénommera « management ». Nous aurons finalement le chemin suivant :

```
-/                                          index.js
-------/signup
-------/token
        ---------/accesses
-------/login
        ---------/admin                     admin.js
        ---------/…
                  ---------/management      management.js
```

La protection se fera de la même manière que pour la zone administrateur (voir 7.3), au moyen d'un middleware qui va comparera le rôle de l'utilisateur authentifié. La seule différence résidera dans le fait qu'elle se situera dans « admin.js ».

routes/admin.js
```
router.use('/managment',isSuperAdmin, managmentRouter);

//Check if the user is superadmin
function isSuperAdmin(req, res, next){
    if(req.session.role !== 'superAdmin')
       res.redirect('/admin');
    next();
}
```

Nous n'ajoutons aucun bouton pour accéder à cette partie super administrateur, l'accès se fera directement via le chemin « /admin/managment ».

10.2. Filtrage des accès

Afin de garantir une certaine confidentialité des données, il est prévu de filtrer et restreindre les accès aux projets.

Nous définissons avec notre PO qu'un chef de projet pourra uniquement voir les projets dans lesquelles il est impliqué. Il aura un accès complet avec la liste des patients et une possibilité d'exporter les données s'y référant. Il aura également la possibilité d'ajouter

d'autres chefs de projet afin de partager l'accès. Pour gérer ces autorisations, nous modifions la structure de notre base de données de la manière suivante :

- Le champ responsable sera dorénavant mis automatiquement selon la personne qui crée le nouveau projet.
- Nous rajoutons une liste d'adresse mail « access » contenant les utilisateurs ayant accès au projet

Avec ces changements, les projets apparaitront comme ceci dans notre base de données :

Figure 19: Nouvelle structure d'un projet

De cette manière, au moment de lister les projets, nous récupérons uniquement les projets dont la liste « access » contient l'adresse mail de la personne authentifiée. Cette vérification est également faite lors de l'affichage du détail d'un projet afin d'éviter un accès via un lien direct.

De l'autre côté, le super administrateur aura accès à la liste et aux détails de tous les projets de la plateforme. Cependant, il ne pourra pas exporter les données récoltées par Pryv, car celles-ci sont considérées comme confidentielles. De plus, il ne verra pas la liste des patients faisant partie d'un projet afin de garantir l'anonymat.

10.2.1. Gestion des accès

La gestion des accès au projet se fait directement sur la page de ce dernier. De cette manière un chef de projet peut autoriser l'accès à des collaborateurs. Tout comme pour l'export des données, cette fonctionnalité n'est pas disponible pour le super admin.

10.3. Sprint review

Lors de la séance du 16 juillet avec Jean-Paul Calbimonte Pérez, nous présentons notre travail. La partie gestion des utilisateurs est acceptée malgré le fait que les restrictions du super administrateur ne sont filtrées que de manière programmatique mais ce point sera développé dans les améliorations.

Finalement, nous n'aurons pas la possibilité d'implémenter notre application mobile dans ce sprint. En effet, les certificats SSL de l'instance Pryv n'ont pas pu être mis à jour par le responsable de la Hes-so. Cette interface sera développée au chapitre prochain en fonction du temps à notre disposition.

11.Développement mobile

Nous choisissons finalement d'implémenter notre interface mobile avec une application native. Ce choix est justifié par le fait que Pryv met à disposition un exemple pour les deux systèmes d'exploitation Android et IOS. Notre version sera développée exclusivement pour la partie Android.

11.1. API externe sur Pryv admin

Tout d'abord, nous souhaitons avoir un lien entre notre gestion de projets et cette partie mobile. C'est pourquoi nous décidons de faire un point d'entrée depuis notre application web. De cette manière, nous aurons la possibilité de faire des requêtes avec le nom d'utilisateur en paramètre afin de récupérer les projets associés. Il s'agit d'une requête POST avec le nom d'utilisateur dans le corps, puis nous retournons en réponse une liste d'objet contenant les noms des projets ainsi que les identifiants des streams.

routes/api.js
```
/* POST username for projects. */
router.post('/projects', checkForApiToken, function(req, res, next) {

    projects.getProjectsInvolvedApi(req.body.username).then((answer)=>{
       res.send(Object.values(answer));
    })

});

//Check API token
function checkForApiToken(req, res, next){
    if (process.env.API_TOKEN !== req.headers.authorization) {
       res.send(401)
       return;
    }
    next();
}
```

En vue de garantir une sécurité, nous ajoutons un token de manière statique dans nos variables d'environnement. Ainsi, notre middleware pourra contrôler l'accès à cette ressource.

11.2. Mise en place du projet

Pour débuter notre projet, nous suivons le guide de Pryv disponible sur leur compte GitHub. À partir d'un nouveau projet, nous ajoutons la dépendance dans le fichier « build.gradle » de notre solution.

```
implementation 'com.pryv:commons:2.1.1'                    build.gradle
```

Nous ajoutons ensuite les deux classes fournies dans le projet exemple :

- LoginActivity
- Credentials

Celles-ci nous permettent de gérer l'authentification sur Pryv et le stockage de ces informations pour éviter une reconnexion à chaque utilisation. Cette interface va se présenter comme la figure de gauche :

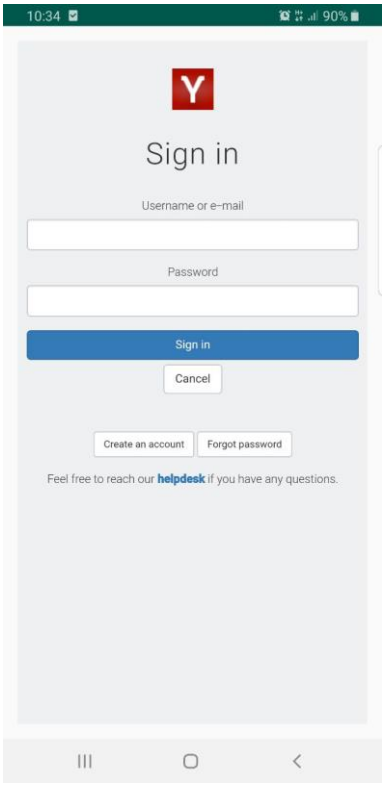


Figure 20: Connexion mobile sur Pryv

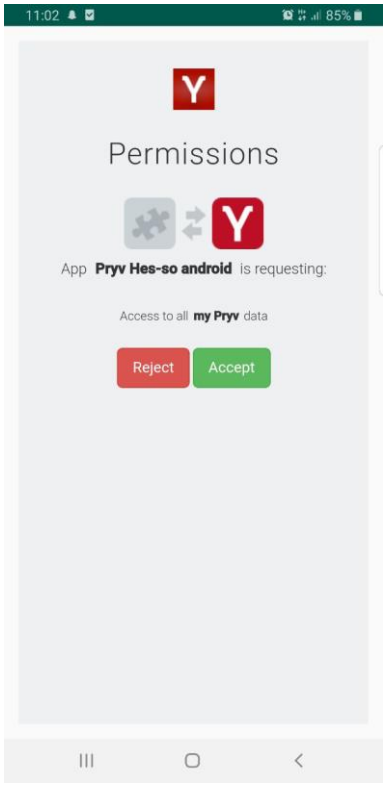


Figure 21: Autorisation de l'application

L'utilisateur pourra ensuite autoriser l'application. Ceci va en réalité générer un token avec tous les droits qui permettront ensuite d'ajouter les données en fonction du projet.

11.3. Liste

Nous souhaitons maintenant afficher la liste des projets dans lesquelles l'utilisateur connecter est impliqué. Pour cela nous utilisons notre API créé précédemment. Celle-ci se présente de cette manière dans notre « MainActivity.java » :

MainActivity.java

```
final OkHttpClient client = new OkHttpClient();

RequestBody body = new FormBody.Builder()
        .add("username", credentials.getUsername())
        .build();
Request request = new Request.Builder()
        .url(DOMAIN_ADMIN+"/api/projects")
        .addHeader("Authorization", API_TOKEN)
        .post(body)
        .build();

client.newCall(request).enqueue(new Callback() {

…
```

Nous pouvons observer que l'appel est fait sur la route « api/projects » de notre interface d'administration. Le corps de la requête contient le nom d'utilisateur stocké préalablement dans la classe « credentials ». Nous avons également notre token API en tant qu'autorisation.

Nous stockons le résultat dans une liste d'objets nommés « Projet » celle-ci contient simplement le nom du projet et l'identifiant du stream qui permettra d'afficher une liste des projets de l'utilisateur connecté.

11.4. Ajout de données

Pour l'ajout d'events, nous mettons en place un simple champ avec un bouton. Puis nous utilisons la classe « Event » mise à disposition par la librairie pour l'ajouter.

AddEventActivity.java

```
Event newEvent = new Event()
        .setStreamId(streamId)
        .setType("note/txt")
        .setContent(etEventField.getText().toString());
connection.events.create(newEvent);
```

Une fois le contenu ajouté, nous retournons également un message de confirmation à l'utilisateur.

12. Bilan

Nous comptons un total de 23 jours de développement partagé en 3 sprints. Durant toute cette phase de développement nous sommes arrivés à un prototype fonctionnel et utilisable.

D'une part, nous avons l'administrateur peut créer, modifier, supprimer et exporter un projet sur notre interface web. À cela s'ajoute également la gestion des patients affiliés à ce dernier. Puis nous avons le patient qui peut s'enregistrer sur Pryv et accepter ou révoquer le partage de ses données pour un projet.

Autrement nous avons notre application mobile qui nous permet, une fois authentifiée, d'avoir la liste des projets affiliés pour y ajouter des events.

L'interface a été déployée sur le serveur de la Hes-so et est disponible à l'adresse https://pryv.p645.hevs.ch jusqu'à fin septembre 2019. Un guide d'utilisation est disponible en annexe VI.

12.1. Améliorations possibles

Bien que notre objectif soit atteint, il est très important de citer les améliorations possibles de notre solution.

12.1.1. User experience

User experience ou expérience utilisateur fait référence à l'ergonomie et l'utilisabilité du logiciel. Dans notre cas, nous avons souligné certains points avec notre PO quant à l'amélioration de cette dernière.

En premier lieu, nous avons le filtrage des accès du super administrateur qui est fait de manière programmatique. C'est-à-dire que l'utilisateur à un accès visuel aux fonctionnalités, mais nous lui renvoyons un message d'erreur lui signalant qu'il n'a pas les droits en cas d'action non autorisée. Il serait préférable d'adapter la page en fonction des droits de la personne authentifiée de manière à afficher uniquement les fonctionnalités accessibles.

Ensuite, nous soulignons un problème lors de l'ajout multiple d'utilisateurs lors de la création d'un projet. Cette fonctionnalité ne pose pas de problème pour l'instant car nous n'avons pas beaucoup de patients enregistrés sur ce service. Mais si nous devions un jour en avoir une beaucoup, il serait préférable de l'améliorer.

12.1.2. Authentification via Hes-so

Si l'on devait choisir d'implémenter une nouvelle fonctionnalité, nous choisirions une authentification exploitant les comptes utilisateurs de la Hes-so. Cela nous permettrait de déléguer une grande partie de la création des comptes et de la gestion des mots de passe. De cette manière, nous pourrions autoriser n'importe quel collaborateur de la haute école spécialisé à accéder sur notre plateforme.

Figure 22: Connexion avec Hes-so

12.1.3. Multi langue

Tout au long de ce projet, la gestion de plusieurs langues ne constituait pas notre priorité. C'est la raison pour laquelle nous ne l'avons pas implémentée. Si nous partons du principe que notre application web pourrait éventuellement être utilisée par différents groupes linguistiques, il serait alors nécessaire de traduire les pages. Le gestionnaire de librairie NPM

dispose d'un module nommé « i18n », qui une fois installé, permet de gérer cet aspect multilingue.

Actuellement, le texte est placé de manière statique dans nos pages. C'est pourquoi il est nécessaire de l'inclure dynamiquement en fonction des variables créées avec i18n.

Pour plus d'informations, voici la librairie en question :
https://www.npmjs.com/package/i18n

12.1.4. Accès aux données via API

Cette fonctionnalité requiert une analyse plus précise quant au besoin et cas d'utilisation possible, mais nous pourrions imaginer que le chef de projet autorise un accès aux données via un token. Ceci permettrait l'intégration de notre solution a d'autres projets comme des analyses de données par exemple.

13.Conclusion

L'interaction avec la plateforme Pryv se faisait jusqu'à présent avec des requêtes HTTP et cela rendait son utilisation difficile, voire impossible. En effet, les personnes en charge des projets de recherche au sein de la DAunit ne sont pas forcément liées au domaine de l'informatique et n'ont pas les connaissances requises pour faire ce type d'interaction. Ce qui est une réelle perte au vu des possibilités qu'offre ce service.

Ce projet nous a permis de mieux comprendre le fonctionnement de Pryv pour analyser ensuite les possibilités et les besoins requis pour cette interface de gestion. Nous avons finalement réussi à implémenter avec succès un prototype stable et fonctionnel malgré les quelques améliorations possibles d'un point de vue ergonomique notamment.

Le stockage des données avec cette nouvelle technologie s'est révélé être très intéressant en dépit des difficultés rencontrées au cours de cette tâche. Mais encore, le soutien des développeurs de l'entreprise Suisse a été très utile en complément de la documentation très détaillée disponible en ligne.

Du côté de l'application mobile, nous avons pu développer une application certes limitée mais tout à fait fonctionnelle permettant d'avoir une base pour un futur projet plus complet. À noter que les problèmes de mise à jour de certificat ont tout de même été un frein pour l'avancée du projet.

Ce projet a été très bénéfique d'un point de vue pratique. Car nous avons eu l'occasion de confronter et mettre en pratique nos connaissances acquises tout au long de cette formation.

RÉFÉRENCES

Coles, E. (2018, 05). *The SQL vs NoSQL Difference: MySQL vs MongoDB*. Récupéré sur www.xplenty.com: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/

Docker Inc. (s.d.). *What is a Container ?* Récupéré sur Docker: https://www.docker.com/resources/what-container

Dotnettutorials. (2018). *C# Tutorials*. Récupéré sur Dotnettutorials: https://dotnettutorials.net/course/csharp-dot-net-tutorials/

Express.js. (s.d.). *Infrastructure Web minimaliste, souple et rapide pour Node.js*. Récupéré sur Express.js.

Fontanet, J., & Lambert, O. (2015). *Node.js, Exploitez la puissance de JavaScript côté serveur.*

Fournier, F. (2015). *Django Industrialisez vos développements Python.*

Herren, J. (2018). *Suivi de patients en physiothérapie à domicile via une application smartwatch.*

Herrenschnieder, D. (2017). *Présentation du framework Laravel*. Récupéré sur Supinfo: https://www.supinfo.com/articles/single/5637-presentation-framework-laravel

Laravel. (s.d.). *Laravel - The PHP Framework For Web Artisans*. Récupéré sur Laravel: https://laravel.com/

Microsoft. (2019). *Bien démarrer avec le .NET Framework*. Récupéré sur Microsoft: https://docs.microsoft.com/fr-fr/dotnet/framework/get-started/

Node.js. (s.d.). *About Node.js*. Récupéré sur Node.js: https://nodejs.org/en/about/

openclassrooms. (2019). *Apprenez à développer en C#*. Récupéré sur openclassrooms: https://openclassrooms.com/fr/courses/1526901-apprenez-a-developper-en-c/1527058-introduction-au-c

PHP sources. (s.d.). *Présentation de MySQL*. Récupéré sur Php source: https://phpsources.net/presentation/mysql/index

Pompidor, P. (2019). *Optimisez le développement de vos applications web avec une architecture MEAN (2e édition).*

Projects Plaza. (2018). *Create simple static website with nodejs,express and jade*. Récupéré sur Projects Plaza: http://projectsplaza.com/create-simple-static-website-nodejsexpress-jade/

Pryv SA. (2019). *Data in pryv.* Récupéré sur Pryv: https://pryv.com/wp-content/uploads/2019/06/Pryv_WhitePaper.pdf

Pryv SA. (s.d.). *Présentation*. Récupéré sur Pryv.io: https://pryv.com/

Registre du commerce du canton de Vaud. (s.d.). *Registre du commerce*. Récupéré sur Etat de Vaud: https://www.rc2.vd.ch/registres/hrcintapp-pub/

Scenari-Community. (s.d.). *MongoDB - Présentation de MongoDB*. Récupéré sur Scenari-Community: https://stph.scenari-community.org/contribs/nos/Mongo1/co/presentation.html

Shuup. (2017). *25 of the Most Popular Python and Django Websites*. Récupéré sur Shuup: https://www.shuup.com/django/25-of-the-most-popular-python-and-django-websites/

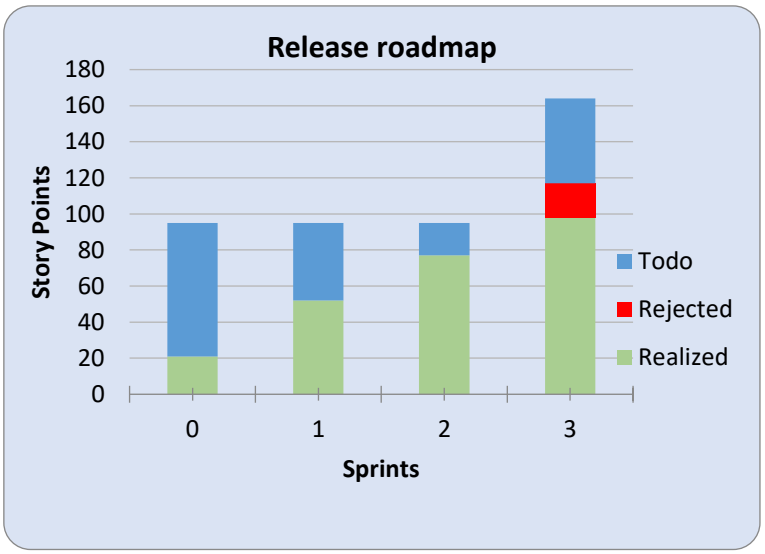
The PHP Group. (s.d.). *Qu'est ce que PHP?* Récupéré sur Php: https://www.php.net/manual/fr/intro-whatis.php

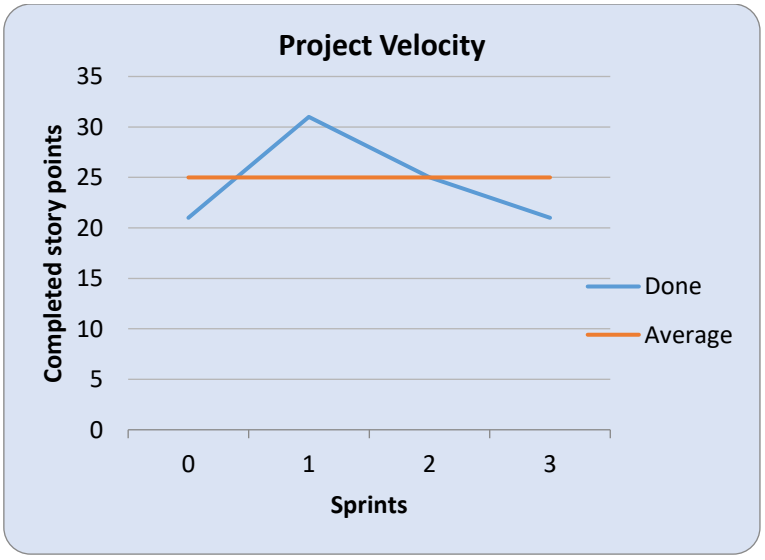
Annexe I : Product Backlog

Nr.	En tant que	Je veux	afin de	Acceptance criteria	Priority	Status	Story Points	moscow	Initial	Current	US accepted (done done)
10	Développeur	Faire un état de l'art des technologies pour l'interface	choisir la technologie la plus adaptée		1 000	●	8	M	0	0	03/06/19
11	Développeur	Faire des mockups de l'interface	avoir un visuel du résultat		990	●	5	M	0	0	03/06/19
12	Développeur	Mettre en place l'environnement de développement	être prêt pour le développement		980	●	8	M	0	0	03/06/19
13	Développeur	Préparer le déploiement	avoir la solution enligne	app accessible sur le serveur	975	●	3	M	1	1	18/06/19
20	Patient	Créer un compte utilisateur	être enregistré sur pryv	enregistré sans doublons	970	●	5	M	1	1	18/06/19
21	Patient	Créer un token	donner accès à mes données	données accessibles avec le token	960	●	3	M	1	1	18/06/19
22	Patient	Afficher mes tokens	avoir une liste des accès à mes données	liste de mes tokens visible	950	●	2	M	1	1	18/06/19
23	Project manager	Ajouter des utilisateurs a un projet	avoir des utilisateurs liés à un groupe	utilisateur présent dans la base de données	900	●	5	M	2	2	02/07/19
25	Patient	Supprimer un token	de bloquer l'accès	token inutilisable	850	●	2	M	2	2	02/07/19
26	Admin/ project manager	Voir les projets par utilisateur	voir à quelle étude participe un utilisateur	avoir une liste des patients d'un projet	800	●	3	S	2	2	02/07/19
28	Project manager	Mise à jour status du token	avoir un visuel dans le projet	"supprimé" dans le détail du projet	795	●	2	S	3	3	16/07/19
30	Admin/ project manager	Afficher un utilisateur	voir les détails sur un utilisateur	détails du l'utilisateur visible	940	●	3	M	1	1	18/06/19
32	Patient	Afficher les hébergements disponibles	afin de voir les possibilités d'hébergements	avoir une liste des hébergements	820	●	2	M	2	2	02/07/19
33	Admin	Trouver le nom d'utilisateur d'un mail	rechercher un utilisateur	email correspond à l'utilisateur	700	●	2	C			
34	Admin	Afficher les informations du serveur	avoir plus de détails sur le serveur	info correspond au serveur	690	●	2	C			
35	Admin	Voir les utilisateurs par core	avoir une vue d'ensemble des utilisateurs	utilisateur correspond au serveur	500	●	2	W			
37	Admin	Trouver le core d'un nom d'utilisateur	savoir sur quel serveur se trouve un utilisateur	core correspond à l'utilisateur	480	●	2	W			
40	Admin/ project manager	Créer un nouveau projet	avoir un groupe de recherche	projet ajouté sur la base de données	930	●	5	M	1	1	18/06/19
41	Project manager	Modifier un projet	de mettre à jour les détails d'un projet	modification visible sur la base de données	920	●	3	M	1	1	18/06/19

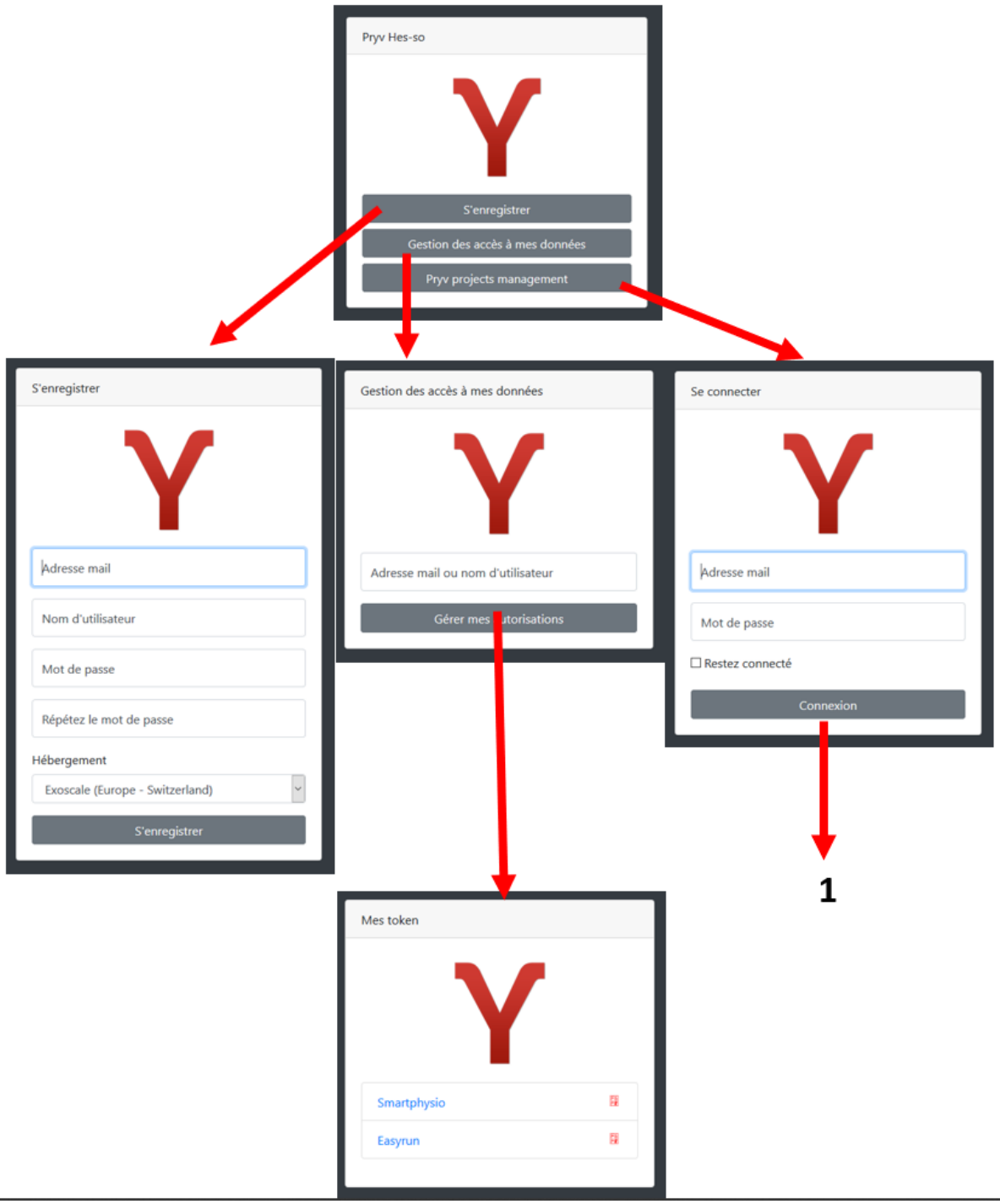
42	Admin/ project manager	Voir la liste de tous les projets	avoir une vue d'ensemble des projets	liste visuelle des projets	910	●	3	M	1	1	18/06/19
43	Project manager	Définir le statut d'un projet	organiser le projet	status à jour dans la base de données	880	●	2	M	1	1	18/06/19
44	Project manager	Définir la langue d'un projet	organiser le projet	langue à jour dans la base de données	870	●	2	M	1	1	18/06/19
50	Project manager	Accéder aux données pour un projet	pouvoir étudier ces données	fichier exporté des données du projet	890	●	8	M	2	2	02/07/19
51	Project manager	Avoir un status des accès	savoir quel utilisateur j'ai accès	liste de tous les accès avec ce token	830	●	5	M	2	2	02/07/19
61	Patient	Connexion au compte pryv	avoir mon compte pryv sur l'app	connecté avec pryv	780	●	3	S	3	3	
62	Patient	Déconnexion du compte pryv	fermer ma session	déconnecté de l'app mobile	785	●	3	S	3	3	
63	Patient	Voir la liste de mes projets	voir les projets dont je fais parti	liste des projets	781	●	5	S	3	3	
64	Patient	Ajouter des données	de participer à l'étude	donnée sur le serveur pryv	782	●	8	S	3	3	
65	Patient	Voir mes données pour un projet	voir un historique de ma participation	liste des events	750	●	5	C			
66	Patient	Supprimer une donnée d'un projet	supprimer une erreur	donnée supprimée du serveur pryv	760	●	3	C			
70	Admin	Créer un utilisateur project manager	donner accès à l'interface	utilisateur présent dans la base de données	794	●	5	S	3	3	16/07/19
71	Admin	Supprimer un utilisateur project manager	supprimer l'accès à l'interface	utilisateur supprimée dans la base de données	793	●	2	S	3	3	16/07/19
72	Admin	Réinitialiser un mot de passe	gérer les mots de passe oubliés	mail avec nouveau mot de passe	776	●	2	C			
73	Admin/ project manager	Modifier mon mot de passe	de mettre à jour mon mot de passe	mot de passe modifié dans la base de données	775	●	2	C			
74	Admin	Avoir seulement accès à l'information non sensible	voir uniquement les données de base	affichage limité	792	●	2	S	3	3	16/07/19
75	Project manager	Avoir seulement accès à mes projets	de ne voir que mes projets	uniquement liste des projets participants	791	●	5	S	3	3	16/07/19
76	Project manager	Ajouter un manager au projet	avoir plusieurs managers de projet	manager présent dans le projet	790	●	3	S	3	3	16/07/19
77	Project manager	Supprimer un manager du projet	interdire l'accès au projet	manager supprimé du projet	789	●	2	S	3	3	16/07/19
80	Project manager	Avoir une notification lors de la suppression de token	savoir si un utilisateur supprime un token	mail avec les détails	680	0	3	C			
81	Admin	Envoyer un mail à tous les utilisateurs d'un projet	notifier les utilisateurs	emails envoyés aux utilisateurs	470	0	5	W			
Tot.							**145**				

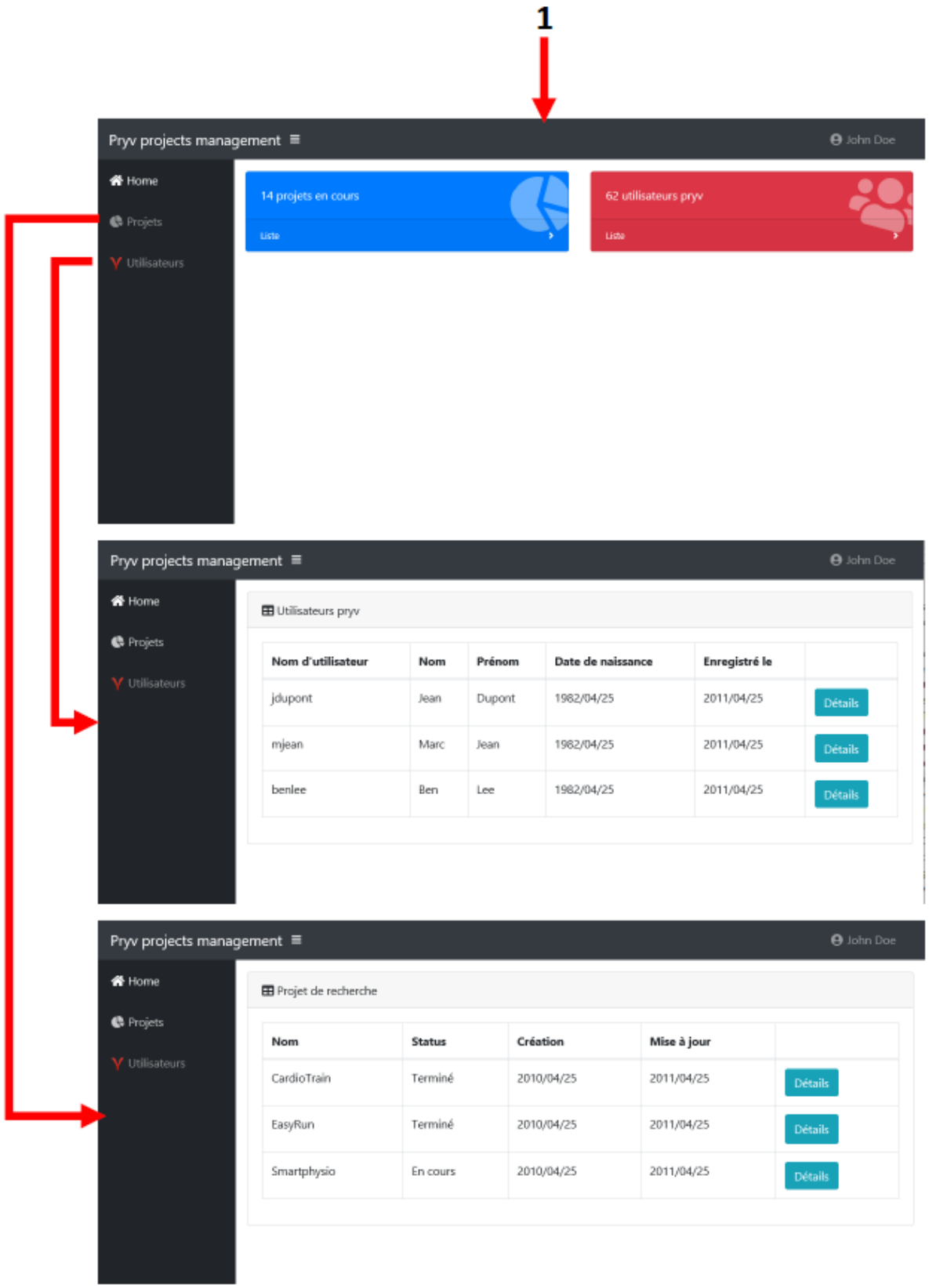
Average		25			
Sprint N°	Initial	0	1	2	3
Nbr SP Sprint		21	31	25	40
Done		21	31	25	21
Todo	95	74	43	18	47
Changes est.		0	0	0	0
New Stories		0	0	0	50
Realized		21	52	77	98
Average		25	25	25	25
Rejected		0	0	0	19
Scope Change		0	0	0	-50
Remaining SP		74	43	18	47
Remaining theoretical (linear)	95	70	45	20	-5
Scope Change		0	0	0	-50

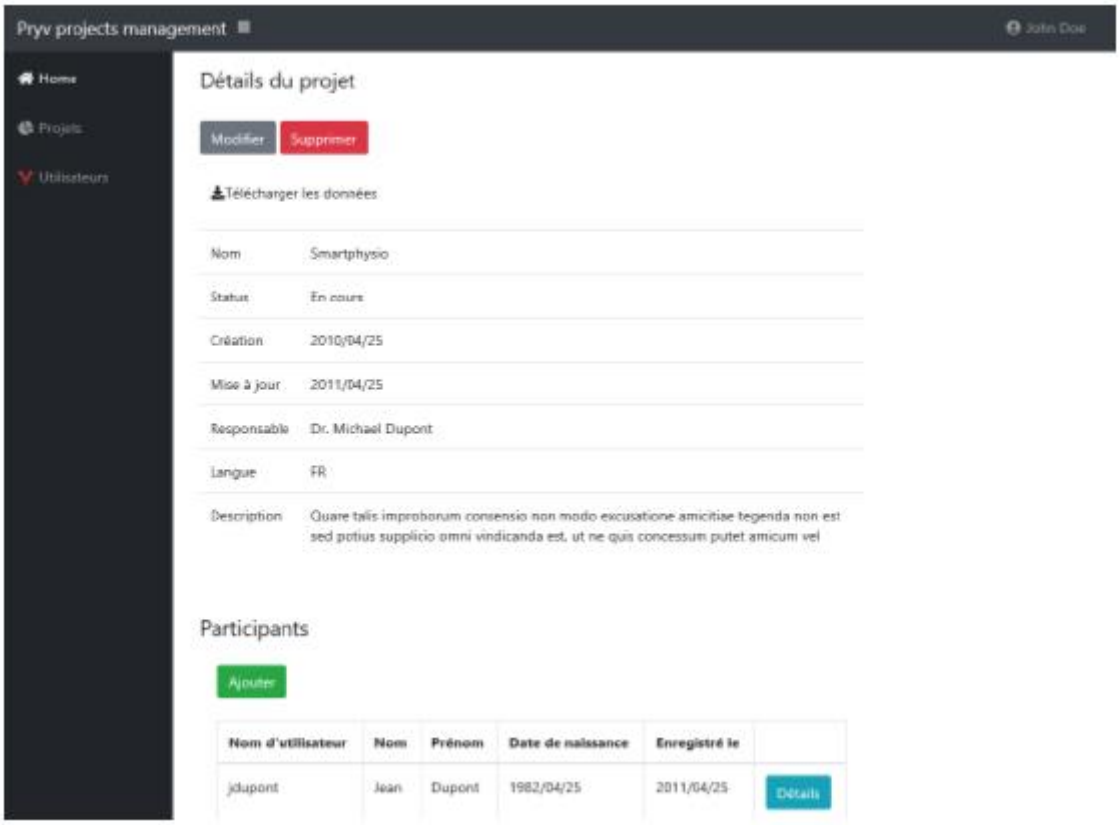


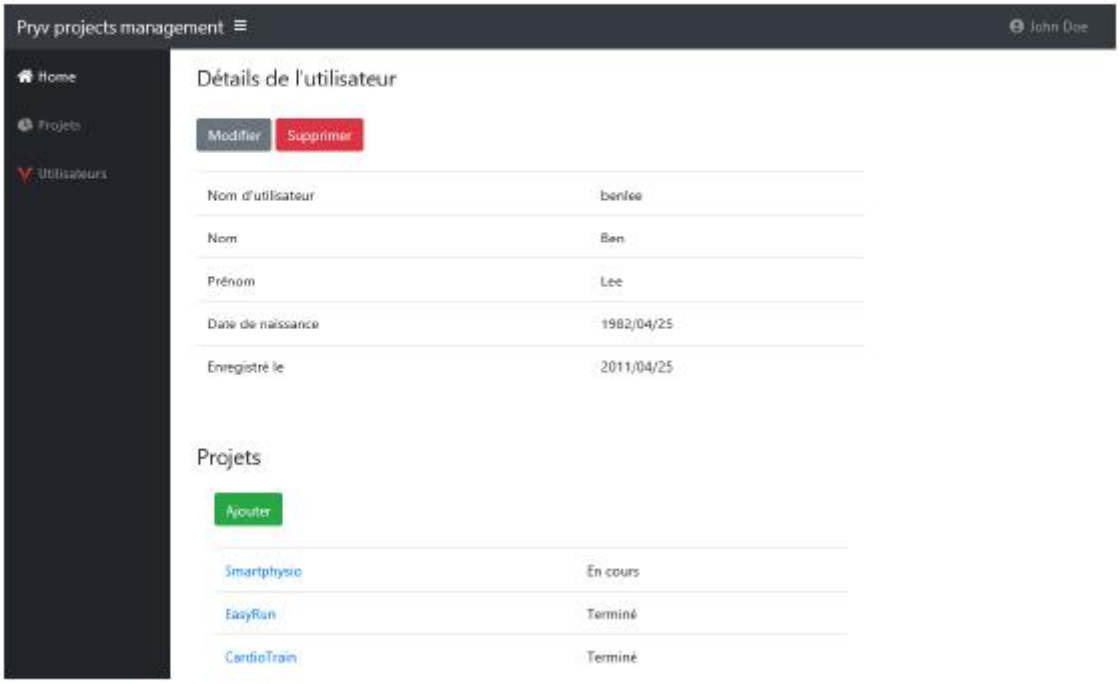


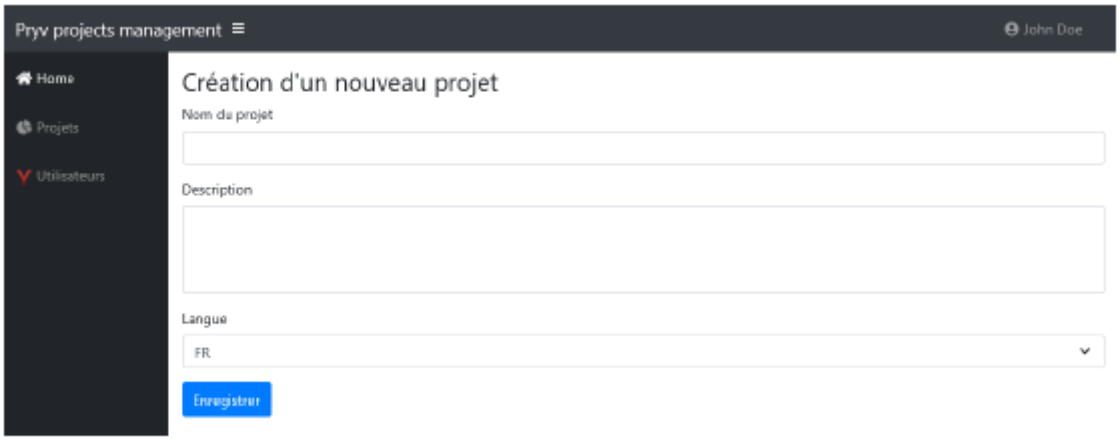
Annexe II : Mockups de l'interface




1
Pryv projects management
John Doe
Home
Projets
Utilisateurs
14 projets en cours
Liste
62 utilisateurs pryv
Liste
Pryv projects management
John Doe
Home
Projets
Utilisateurs
Utilisateurs pryv
Nom d'utilisateur
Nom
Prénom
Date de naissance
Enregistré le
jdupont
Jean
Dupont
1982/04/25
2011/04/25
Détails
mjean
Marc
Jean
1982/04/25
2011/04/25
Détails
benlee
Ben
Lee
1982/04/25
2011/04/25
Détails
Pryv projects management
John Doe
Home
Projets
Utilisateurs
Projet de recherche
Nom
Status
Création
Mise à jour
CardioTrain
Terminé
2010/04/25
2011/04/25
Détails
EasyRun
Terminé
2010/04/25
2011/04/25
Détails
Smartphysio
En cours
2010/04/25
2011/04/25
Détails


Pryv projects management
John Doe
Home
Projets
Utilisateurs
Détails du projet
Modifier
Supprimer
Télécharger les données
Nom
Smartphysio
Status
En cours
Création
2010/04/25
Mise à jour
2011/04/25
Responsable
Dr. Michael Dupont
Langue
FR
Description
Quare talis improborum consensio non modo excusatione amicitiae tegenda non est sed potius supplicio omni vindicanda est, ut ne quis concessum putet amicum vel
Participants
Ajouter
Nom d'utilisateur
Nom
Prénom
Date de naissance
Enregistré le
jdupont
Jean
Dupont
1982/04/25
2011/04/25
Détails


Pryv projects management
John Doe
Home
Projets
Utilisateurs
Détails de l'utilisateur
Modifier
Supprimer
Nom d'utilisateur
benlee
Nom
Ben
Prénom
Lee
Date de naissance
1982/04/25
Enregistré le
2011/04/25
Projets
Ajouter
Smartphysio
En cours
EasyRun
Terminé
CardioTrain
Terminé


Pryv projects management
John Doe
Home
Projets
Utilisateurs
Création d'un nouveau projet
Nom du projet
Description
Langue
FR
Enregistrer

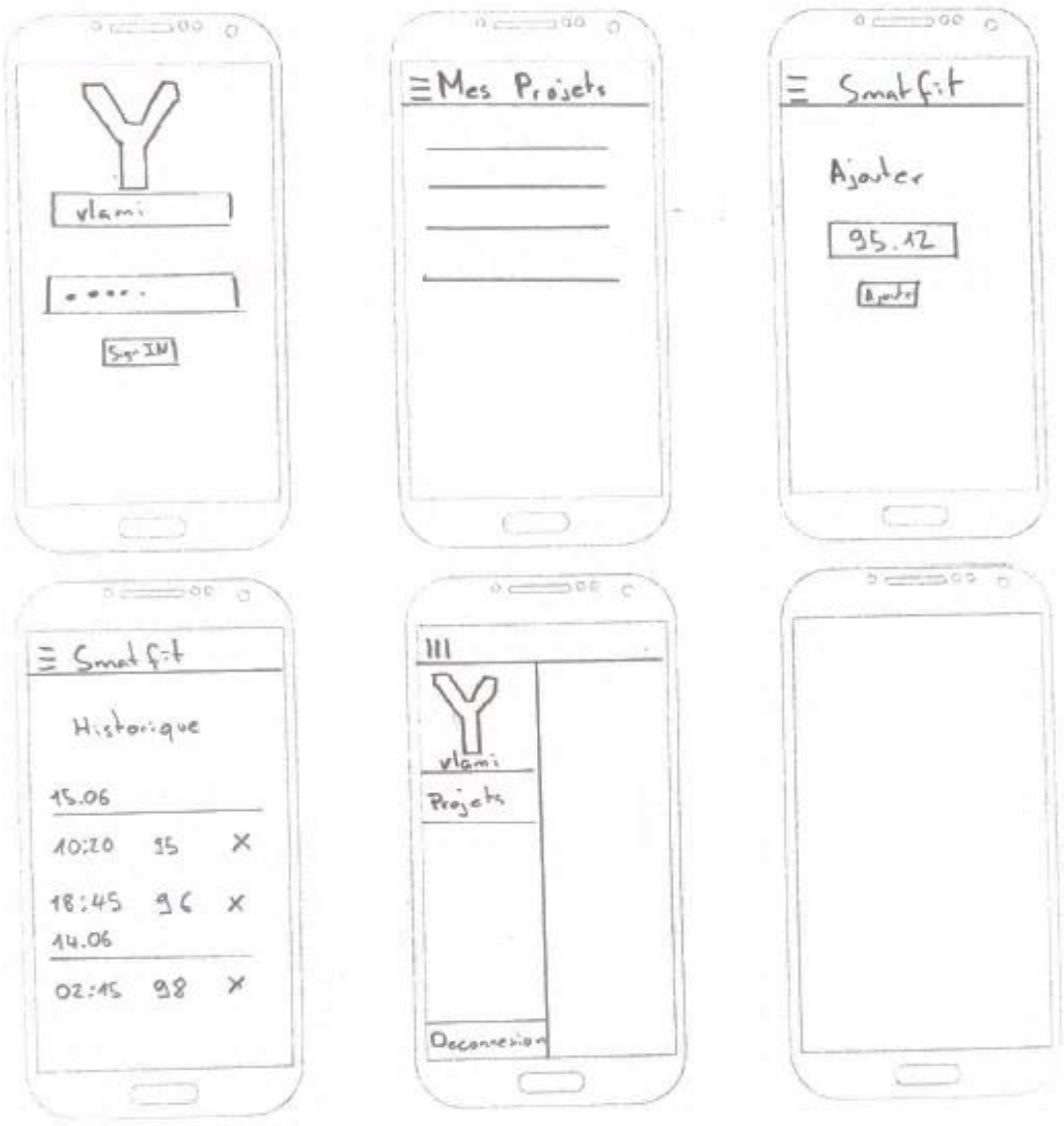
Annexe III : docker-compose.yml

```
version: '3.6'

services:

    backend:
        build: .
        labels:
          - 'traefik.backend=pryv-tb'
          - 'traefik.docker.network=prod'
          - 'traefik.frontend.rule=Host:pryv.p645.hevs.ch'
          - 'traefik.enable=true'
        environment:
          NODE_ENV:          production
          DB_CONNECTION:  $DB_PROTOCOL://$DB_USERNAME:$DB_PASSWORD@$DB_HOST:$DB_PORT$
          PRYV_TOKEN: $PRYV_TOKEN
          PRYV_DOMAIN: $PRYV_DOMAIN
          DOMAINS:       ${DOMAINS}
          SESSION_SECRET: ${SESSION_SECRET}
          EMAIL_HOST: ${EMAIL_HOST}
          EMAIL_ADDR: ${EMAIL_ADDR}
          EMAIL_PWD: ${EMAIL_PWD}
          API_TOKEN: ${API_TOKEN}
        dns: '8.8.8.8'
        restart: always
        networks:
          - default
          - prod
        depends_on:
          - db


    db:
        image: mongo:latest
        environment:
          MONGO_INITDB_ROOT_USERNAME: ${DB_ADMIN_USERNAME}
          MONGO_INITDB_ROOT_PASSWORD: ${DB_ADMIN_PASSWORD}
          MONGO_INITDB_DATABASE: ${DB_NAME}
        restart: always
        networks:
          - default
        volumes:
          - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
          - /etc/timezone:/etc/timezone:ro
          - /etc/localtime:/etc/localtime:ro

networks:
    prod:
        external: true
        name: ${NETWORK}
```

Annexe IV : Exemple d'export de données

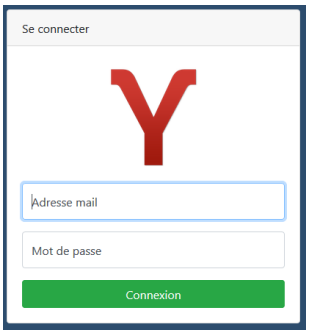

```
{
    "id": "5d19b259287f98001c7214a4",
    "name": "Smartphysio",
    "responsable": "Dr. Jean Dupont",
    "description": "Ce projet à pour but de faire une rechercher sur les niveaux de douleurs",
    "status": "En cours",
    "lang": "FR",
    "authorizations": [
        {
            "streamId": "smartphysio",
            "defaultName": "SmartPhysio",
            "level": "read"
        }
    ],
    "patients": [
        {
            "token": "cjxlevdq6006w0a0l47gs820k",
            "username": "vlami"
        }
    ],
    "created_at": "2019-07-01T07:12:25.415Z",
    "updatedAt": "2019-07-11T11:07:10.648Z",
    "__v": 2,
    "results": [
        {
            "patient": "vlami",
            "data": [
                {
                    "streamId": "smartphysio",
                    "type": "mass/kg",
                    "content": 95,
                    "time": 1562048224.714,
                    "tags": [],
                    "created": 1562048224.714,
                    "createdBy": "cjxlf5ozm00730a0l0cuyplal",
                    "modified": 1562048224.714,
                    "modifiedBy": "cjxlf5ozm00730a0l0cuyplal",
                    "id": "cjxlf6k0a00750a0loi51vxz2"
                },
                {
                    "streamId": "smartphysio",
                    "type": "mass/kg",
                    "content": 91,
                    "time": 1562048215.047,
                    "tags": [],
                    "created": 1562048215.047,
                    "createdBy": "cjxlf5ozm00730a0l0cuyplal",
                    "modified": 1562048215.047,
                    "modifiedBy": "cjxlf5ozm00730a0l0cuyplal",
                    "id": "cjxlf6cjs00740a0l5vni43gh"
                }
            ]
        }
    ]
}
```

Annexe V : Mockups de l'application mobile

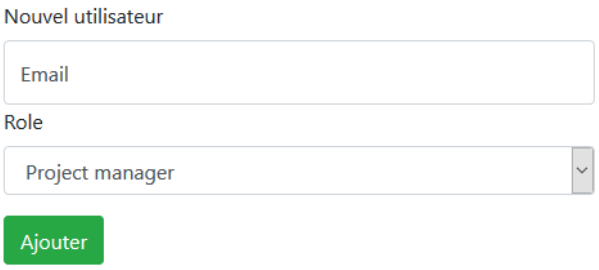


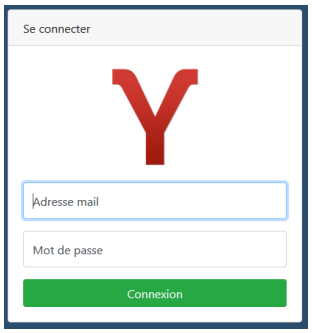
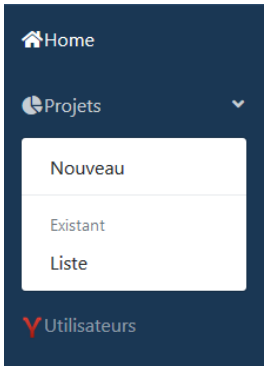
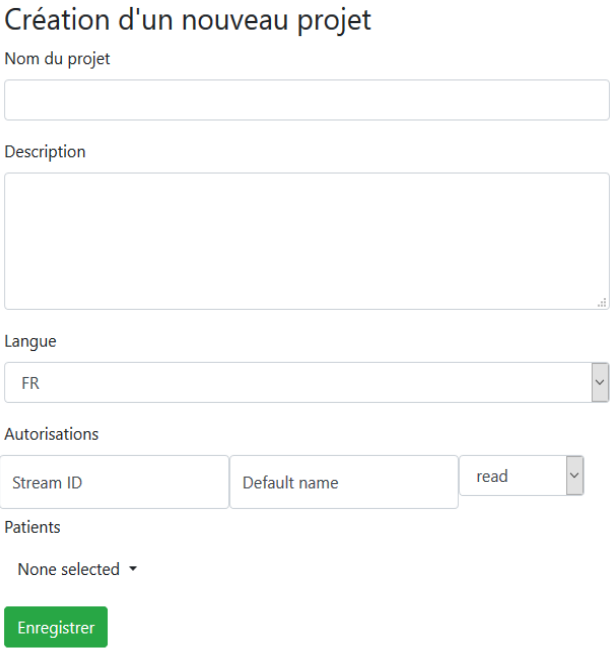
Annexe VI : Guide d'utilisation

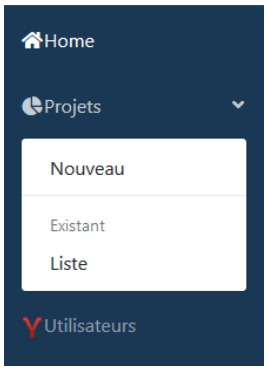
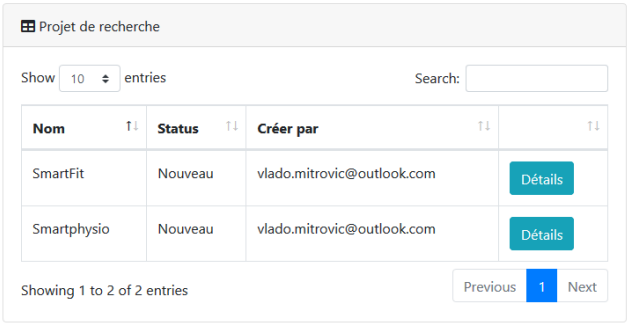
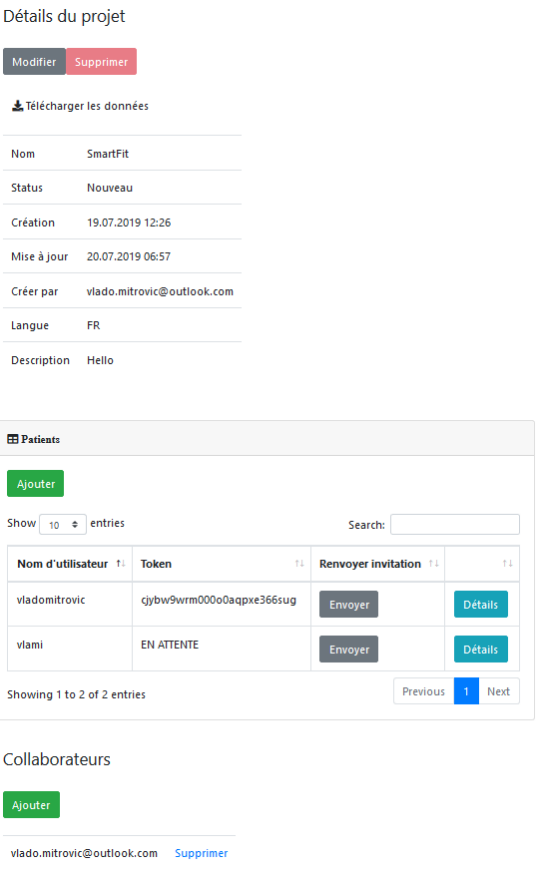
<table>
<tr><th colspan="3">Super administrateur</th></tr>
<tr><td>Étape 1</td><td>Allez à l'adresse https://pryv.p645.hevs.ch et cliquez sur « Pryv projects managment ».

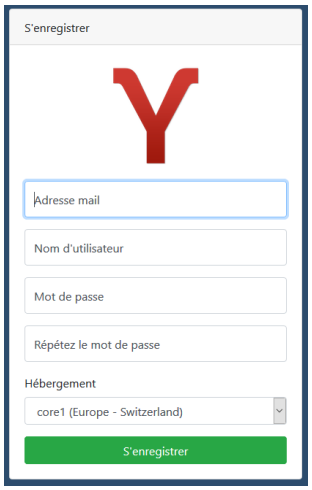
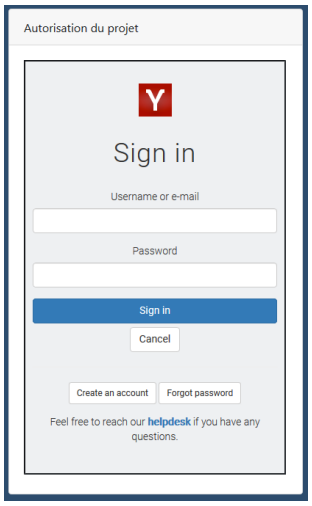
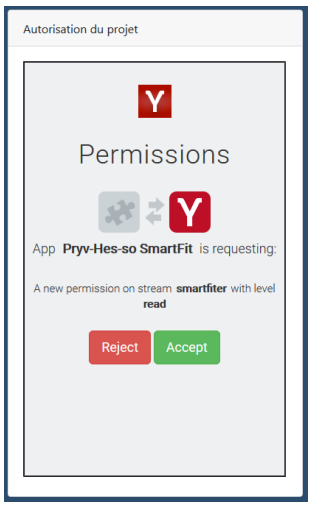
Entrer les accès super admin que vous trouverez dans le fichier acces.txt de la clé USB jointe à ce rapport</td><td> 
</td></tr>
<tr><td colspan="3">Ajouter un utilisateur</td></tr>
<tr><td>Étape 1</td><td>Allez à l'adresse https://pryv.p645.hevs.ch/admin/managment

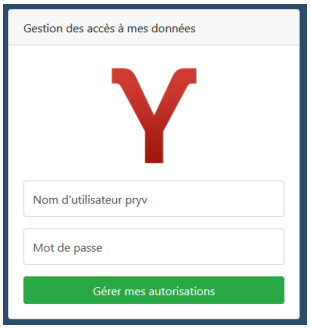
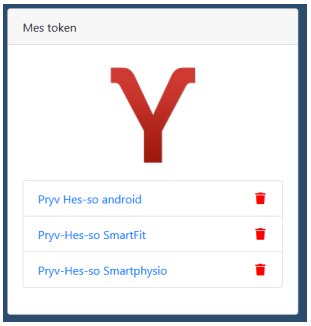
Saisissez l'adresse mail de l'utilisateur avec un rôle.

Un mot de passe lui sera transmis par email.</td><td> 
</td></tr>
</table>

Project manager		
Étape 1	Allez à l'adresse https://pryv.p645.hevs.ch et cliquez sur « Pryv projects managment ». Authentifiez-vous avec votre adresse mail et le mot de passe reçu par email.	
Création d'un projet		
Étape 1	Sur la barre latérale, cliquez sur projets puis nouveau.	
Étape 2	Entrez le nom, la description, la langue et les autorisations du projet. Sélectionnez les patients participants, puis enregistrer. Un email d'invitation sera automatiquement envoyé aux participants.	

Gestion du projet		
Étape 1	Sur la barre latérale, cliquez sur projets puis liste.	
Étape 2	Vous retrouvez la liste des projets auxquels vous avez accès. Cliquez sur détails pour ouvrir un projet.	
Étape 3	Depuis cette page, vous avez accès aux détails d'un projet et vous pouvez y effectuer les actions suivantes : • Modification • Suppression • Export des données • Ajout de participants • Ajout de collaborateurs • Suppression de collaborateur	

Patient		
S'enregistrer		
Étape 1	Allez à l'adresse https://pryv.p645.hevs.ch et cliquez sur « S'enregistrer ». Remplissez le formulaire avec vos informations.	
Autoriser ses données		
Étape 1	Ouvrez le lien reçu par email. Connectez-vous avec vos identifiants Pryv.	
Étape 2	Accepter les permissions requises par le projet. Vous êtes redirigé vers la page accueil avec un message de confirmation.	

Gérer ses autorisations		
Étape 1	Allez à l'adresse https://pryv.p645.hevs.ch et cliquez sur « Gestion des accès à mes données ». Authentifiez-vous avec votre nom d'utilisateur et le mot de passe Pryv.	
Étape 2	Cliquez sur un token pour afficher les détails ou sur la corbeille pour le supprimer.	

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autres aides que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- Michael Ignaz Schumacher, professeur HES et collaborateur à l'institut d'informatique de gestion.
- Jean-Paul Calbimonte Pérez, collaborateur à l'institut d'informatique de gestion

Sierre, le 31 juillet 2019

Vlado Mitrovic