Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Haute Ecole de Gestion
Hochschule für Wirtschaft

TRAVAIL DE BACHELOR 2021

Projet Fit-iN

Étudiant : Patrick Mettraux

Professeur : Jean-Pierre Rey

Mandant : Jérôme Nanchen

Date de dépôt : 13 août 2021

RÉSUMÉ

Mots-clés : Travail de Bachelor, Fit-iN, Application, Sport, Pandémie.

Ce rapport détaille le processus d'architecture et de création du projet Fit-In. Le projet Fit-In est une application destinée aux élèves du secondaire I et II en Valais. La situation due au Covid19 a confirmé un constat déjà tiré sur la condition physique et l'activité physique des adolescents, toutes deux en baisse (Gregor Starc, 2020). Cette application veut aborder d'une manière complémentaire l'éducation physique, en difficulté pendant l'enseignement à distance.

Fit-In veut redonner envie aux élèves de s'occuper de leur santé physique et mentale et également donner la possibilité aux professeurs d'évaluer les élèves sur des critères et domaines différents de l'approche actuelle qui a toujours voulu que le sport soit évalué sur la performance. Or, un enfant obèse n'aura jamais les même performances qu'un enfant en bonne santé, cela ne veut pas dire qu'il faut l'exclure du système d'éducation physique. Il faut trouver une autre manière de l'évaluer et de le guider sur la voie sportive. C'est le défi que se lance cette application. De manière plus générale, l'app veut encourager la pratique dans toutes les situations d'autonomie : semi-confinement, quarantaine, absence au cours, dispenses partielles ou totales, options, travail par chantiers, etc.

REMERCIEMENTS

Karin Hagemann, pour la relecture orthographique de ce document.

Jérôme Nanchen, pour sa disponibilité et son envie de créer quelque chose de nouveau dans l'enseignement du sport chez les jeunes.

Jean-Pierre Rey, pour ses nombreux retours et conseils pour que ce rapport ressemble à ce qu'il est aujourd'hui.

Mes amis, ainsi que toutes les personnes qui m'ont soutenu moralement pour cette reprise des études dans la 30aine.

La COVID-19, sans qui j'aurais certainement trouvé mieux à faire que de continuer les études et qui m'a également fourni le sujet de travail de bachelor.

GLOSSAIRE

API : Acronyme en Anglais signifiant Application Programming Interface ou en français, interface de programmation.

AWS: Amazon Web Service, c'est la partie spécialisée dans l'hébergement et les services cloud du grand groupe Amazon.

Backend: Correspond à la couche métier dans le modèle multi-tiers.

Backlog: Carnet de produits en français. Il contient la liste des fonctionnalités qui doivent être implémentées dans un produit.

CDN: Content Delivery Network. C'est un réseau de diffusion de contenu. Il permet entre autres de gérer la manière dont va être dirigé le trafic. Il a en général des technologies de caching ce qui permettent de limiter la charge infligée à la destination du trafic d'origine.

DNS: Domaine Name System. C'est en français le système de noms de domaine. Il permet de transformer un nom de domaine internet tel que www.google.com en une adresse IP pour que le protocole HTTP sache où envoyer la requête.

Endpoint: C'est le point d'entrée d'une API. On parle souvent d'endpoint en mentionnant l'URL permettant de contacter l'endpoint. Par exemple: https://mon-api.com/v1/ressources/12 Chaque endpoint est lié à une fonctionnalité de l'API.

Framework:	Ensemble de composants servant de fondation pour des projets informatiques. Il donne une ligne directrice et organisationnelle sur le code et sa structure.
Frontend:	Correspond à la couche de présentation dans le modèle multi-tiers.
ICTVS:	Centre de compétence valaisan du Service de l'enseignement pour tout ce qui est lié à l'information, la communication et la technologie. Il s'occupe de l'informatique des écoles et de l'intégration du numérique.
JavaScript:	Langage de programmation de type script principalement utilisé pour l'interaction utilisateur sur les applications web.
Mockups:	C'est un prototype d'interfaces de l'application. Il ne représente pas un design fini, mais les idées qui démontrent comment l'interface pourrait être utilisée par les utilisateurs de l'application.
Open Source:	Définit généralement un logiciel dont le code source est ouvert d'accès à tout le monde. Cela permet une plus grande transparence et permet à la communauté de participer à son développement. Attention Open Source ne veut pas dire gratuit.
PHP:	Langage de programmation principalement utilisé pour créer des applications web.

POC:	Acronyme en Anglais signifiant Proof Of Concept ou en français, preuve du concept. Il est souvent utilisé dans les processus de développement pour montrer la faisabilité d'un projet.
SASS:	Software As A Service, soit en français le logiciel en tant que service. C'est une manière d'exploiter commercialement un outil en facturant, par exemple, à l'utilisation.
Serveur SMTP:	Serveur qui permet l'envoi d'emails. SMTP est un acronyme anglais (Simple Mail Transfer Protocol) qui se traduit en Simple Protocol de Transfert d'Email.
TypeScript:	Langage de programmation libre et open source créé par Microsoft dans le but d'améliorer et sécuriser le code JavaScript. TypeScript est une sur-couche de JavaScript et est totalement compatible avec le JavaScript.
URL:	Uniform Resource Locator. Terme anglais qui désigne la syntaxe d'accès à une ressource généralement web. Par exemple http://www.google.ch est une URL. mailto:patrick@mettraux-web.net est également une URL.
YAML:	Type de fichier très utilisé dans beaucoup d'outils autour de l'infrastructure et des configurations. CircleCi utilise ce type de fichier pour sa configuration, des outils comme HELM également.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

Figure 21 source: Auteur, dessiné avec https://www.figma.com

Figure 22 source: Auteur, dessiné avec https://www.figma.com

Figure 23 source: Auteur, dessiné avec https://www.figma.com

Figure 24 source: Auteur, dessiné avec https://www.figma.com

Figure 25 source: Auteur, dessiné avec https://www.figma.com

Figure 26 source: Auteur, dessiné avec https://www.figma.com

Figure 27 source: Auteur, dessiné avec https://www.figma.com

Figure 28 source: Auteur, dessiné avec https://www.figma.com

Figure 29 source: Auteur, dessiné avec https://www.figma.com

Figure 30 source: Auteur, dessiné avec https://www.figma.com

Figure 31 source: Auteur, dessiné avec https://www.figma.com

Figure 32 source: Auteur, dessiné avec https://www.figma.com

Figure 33 source: Auteur, dessiné avec https://www.figma.com

Figure 34 source: Auteur, dessiné avec https://www.figma.com

Figure 35 source: Auteur, dessiné avec https://www.figma.com

Figure 36 source: Auteur, dessiné avec https://www.figma.com

Figure 37 source: Auteur, dessiné avec https://www.figma.com

Figure 38 source: Auteur, dessiné avec https://www.figma.com

Figure 39 source: Auteur, dessiné avec https://www.figma.com

Figure 40 source: Auteur, dessiné avec https://www.figma.com

Figure 41 source: Auteur, dessiné avec https://www.figma.com

Figure 42 source: Auteur, dessiné avec https://www.figma.com

Figure 43 source: Auteur, via https://dbschema.com/

Figure 44 source: Auteur, via https://dbschema.com/

Figure 45 source: Auteur, via https://dbschema.com/

Figure 46 source: Auteur, via https://dbschema.com/

Figure 47 source: Auteur, via https://dbschema.com/

Figure 48 source: https://github.com/mailhog/MailHog

Figure 49 source: Auteur, via https://circleci.com

Figure 50 source: Auteur, via https://circleci.com

Figure 51 source: Auteur, via https://circleci.com

Figure 52 source: Auteur, via https://sequencediagram.org

Figure 53 source: Auteur, via https://www.diagrams.net

Figure 54 source: Auteur, via https://poeditor.com

Figure 55 source: Auteur via, https://app.jpc.infomaniak.com

Figure 56 source: Auteur, via https://www.diagrams.net

Figure 57 source: Auteur via, https://app.jpc.infomaniak.com

Figure 58 source: Auteur, via https://www.diagrams.net

Figure 59 source: Auteur, via https://www.diagrams.net

Figure 60 source: Auteur via, https://app.jpc.infomaniak.com

Figure 61 source: Auteur via, https://manager.infomaniak.com

Figure 62 source: Auteur

Figure 63 source: Auteur

Figure 64 source: Auteur

Figure 65 source: Auteur

INTRODUCTION

L'application Fit-In dans son état actuel est un concept créé par Jérôme Nanchen. Ce rapport décrit les différentes étapes qui ont amené ce projet à aboutir à un POC, soit une première version utilisable et démontrable. Les processus de création et de réflexion sont détaillés dans les différents chapitres de ce document. C'est le résultat de multiples échanges avec le mandant du projet, Jérôme Nanchen ainsi que le centre de compétence ICTVS.

Avant propos

Les informations de cette section ont été trouvées grâce à la revue professionnelle en ligne "l'Éducation Physique en mouvement" de décembre 2020 (L'Education Physique en Mouvement, 2020) qui fut principalement dédiée à la présentation du projet Fit-In et aux problématiques qui l'entourent.

Historique

Jérôme Nanchen est un psychologue du sport, professeur d'éducation physique et didacticien secondaire II à la haute école de pédagogique du Valais (HEPVS). Suite à la pandémie de COVID-19 qui a changé notre monde et nos habitudes dès 2020, Jérôme a imaginé comment l'éducation physique pourrait se faire à distance. Cette réflexion a démarré lors du premier semi-confinement vécu dès le 13 mars 2020 et qui aura finalement duré 2 mois.

Problématique

Comment réussir à garder la motivation des élèves même à distance. Quelle solution permettra un engagement suffisant à la pratique sportive? Certes, les enseignants se sont adaptés à la situation sanitaire liée au coronavirus mais cela ne veut pas dire que ces adaptations ont permis de garder la motivation des élèves, et encore moins d'exploiter toutes les potentialités de l'enseignement de l'éducation physique, particulièrement dans les domaines du bien-être, des compétences mentales et de l'éducation du physique.

La revue analyse des outils et des pistes pédagogiques et recense des témoignages d'enseignants en formation, de conseillers en pédagogie et d'autres intervenants liés à l'éducation sportive ou à la pédagogie.

Direction prise

De manière globale tout semble montrer que l'enseignement à distance et cette nouvelle façon d'apprendre en autonomie est une opportunité pour les élèves. Encore faut-il trouver le bon fil rouge et les aspects pertinents liés à la condition physique et au bien-être. Le concept Fit-iN s'attaque à 3 aspects principaux, dans les domaines "My sport", "My well-being" et "My body".

My sport concerne tous les exercices de fitness et d'endurance. My well-being regroupe tout ce qui est lié au mental, aux exercices de relaxation. My body s'occupe des aspects liés à l'hygiène de vie et l'alimentation.

Ces aspects montrent comment cette application veut aborder l'éducation physique au sens large du thème et développer une éducation physique abordant tous les domaines de la littératie physique : moteur, psychologique, social et cognitif.

Mon Healthy score, c'est par ce nom que l'application veut montrer à l'élève son évolution et surtout lui donner envie de s'améliorer. Ce score est un graphique radar regroupant six points clés à définir. Chaque élève, sur la base de tests et questionnaires, aura un score de base, pour par la suite effectuer des exercices lui permettent d'améliorer ce score. Le but est principalement de renforcer l'engagement de l'étudiant sur Fit-iN.

Le nom

Jérôme a choisi le nom fit in car c'est un verbe à particule riche de sens (Jérôme Nanchen, s.d., p. 2). Fit-iN est un verbe issu de l'anglais qui signifie "s'intégrer". Il peut être utilisé de plusieurs manières mais il fait l'association entre le fait d'être en forme physique et le côté comportemental et environnemental des individus.

Evolution

Malheureusement, les noms de domaines relatifs à Fit-iN ne sont plus disponibles. C'est pourquoi Jérôme est en train de chercher une alternative disponible afin de pouvoir réserver au plus vite les noms de domaine nécessaires. Pour le moment, l'alternative qui a le plus convaincu est "Fit-Win".

1. MÉTHODOLOGIE DE TRAVAIL

Considérant la quantité d'inconnues dans ce projet, il sera très important de constamment faire le point avec le mandant afin de s'assurer que tout va toujours dans la bonne direction. Également s'il venait à y avoir un changement dû à un imprévu ou tout autre type de problème, il nous faut un moyen de pouvoir rapidement corriger le tir.

Pour ces raisons, une version allégée de la méthodologie Agile Scrum sera utilisée. Pourquoi allégée? Parce que la méthodologie scrum nécessite normalement trois membres au minimum. Un scrum master, un membre de la team de développement et le directeur de produit (Ken Schwaber and Jeff Sutherland, 2020, p. 5)

L'avantage de scrum est de fonctionner par itération. A la fin de chaque itération on fait le point avec le chef du produit et ensuite on planifie le suivant. Dans le cadre de ce projet, le point est fait avec le mandant. Les itérations s'étendent sur deux semaines.

Figure 1: Spirale Agile

Initialement, durant la phase de planification, un backlog fut mise en place. Il n'aura au final servi que de ligne de conduite durant ce projet et sera très peu utilisé hormis à la toute fin du projet pour vérifier que la ligne a été suivie, ce qui fut le cas.

Durant ce projet, un point fut effectué au maximum toutes les deux semaines avec le mandant. Cela a permi non seulement d'assurer au mandant que le projet avance comme il se doit, mais il me permet également de m'assurer que j'avançais toujours dans la bonne direction.

Lors de la phase de création des mockups, le rythme des rendez-vous avec le mandant était plus proche de un par semaine. Cela était essentiel car il fallait décrire et comprendre tout le fonctionnement de l'application qui se trouvait uniquement dans l'imagination du mandant.

Sans une méthodologie de travail collaborative, le résultat final n'aurait pas été aussi qualitatif que celui obtenu.

2. RÔLES

Qui va utiliser cette application? C'est la question à laquelle il faut répondre avant toute conception. Il est très important d'identifier les types de rôles de l'application.

Tout d'abord, un utilisateur peut avoir plusieurs rôles, mais dans la majorité des cas un utilisateur aura souvent accès qu'à un seul type de rôle. Il a été déterminé avec le mandant que les rôles suivant seront présents:

2.1. Les élèves

Ce sont les utilisateurs principaux de l'application, ils effectuent les exercices, consultent les conseils en nutrition ou encore lisent des conseils en lien avec la santé mentale. Ils envoient leur production à leur professeurs et ont la possibilité d'évaluer l'intérêt de la fiche utilisée.

2.2. Les professeurs

Ils créent les classes, assignent les fiches aux classes et/ou aux élèves. Ils évaluent les exercices effectués par les élèves.

2.3. Les contributeurs

Ils créent les fiches; en cas de sollicitation ou refus des validateurs, ils les modifient.

2.4. Les validateurs

Les validateurs relisent les fiches afin de les modifier, les valider ou les supprimer.

2.5. Les administrateurs

Les administrateurs peuvent créer des utilisateurs dans l'application et leur assigner des rôles. Ils créent, modifient ou encore suppriment le contenu de l'application qui ne change pas régulièrement tel que la liste des écoles.

3. FONCTIONNALITÉS

3.1. Introduction

Il est important de comprendre que l'application Fit-In est un concept incomplet à cette étape du processus. Beaucoup d'idées sont présentes mais il est désormais de mon ressort de travailler avec le mandant afin d'en faire une application fonctionnelle et cohérente. Ci-après sont listées les fonctionnalités principales ainsi que les réflexions autour de leur conception.

3.2. Comptes utilisateurs

Comment les comptes se retrouvent dans l'application Fit-In ? Tout d'abord cela dépend du type de compte. Les comptes administrateurs seront créés via la partie administration de l'application. Les comptes professeurs peuvent également être créés de la même manière. Cependant si une base de données existe déjà avec les listes des élèves et des professeurs cela éviterait des doublons. Fit-In pourrait donc utiliser ces données et les utiliser pour la connexion à l'application. Dans le cas contraire, il nous faut trouver le bon processus pour les professeurs ainsi que les élèves.

3.2.1. Existant

Si les comptes existent il faut que le centre cantonal ICTVS nous fournissent une API permettant l'identification des utilisateurs. Il faut ensuite espérer que des informations telles que l'école, le professeur et la classe puissent être récupérées via l'API. Ceci serait idéal mais dans le cadre de ce projet et de son état actuel cela semble peu réalisable dans le temps imparti. Cependant dans le futur il serait totalement envisageable de changer le système d'authentification par un système externe. Le reste du fonctionnement de l'application n'en sera pas altéré.

3.2.1.1. API disponible

Le centre cantonal ICTVS peut mettre une API de test à notre disposition qui permet aux élèves de s'identifier. Cette API utilise le Security assertion markup (SAML). Le SAML est un standard définissant un protocole servant à l'échange d'informations. Le SAML est utilisé pour effectuer l'authentification unique, Single Sign-On en anglais (SSO).

Lors de la connexion d'un utilisateur, un jeton contenant des données sur l'utilisateur nous est retourné. Il est possible d'associer des informations dans ce jeton; dans notre cas, il serait intéressant pour les élèves d'avoir leur classe et pour les enseignants la liste des classes qu'ils enseignent.

Cependant, cela voudrait dire que pour qu'un enseignant puisse assigner du travail à un élève et que le système Fit-In le connaisse, l'élève doit effectuer au moins une connexion pour que le système puisse le reconnaître.

Le centre cantonal ICT-VS n'a pour le moment aucune API permettant la récupération complète de listes de classes complètes, élèves et professeurs. Cette solution serait idéale dans le futur et éviterait la redondance de données.

3.2.2. Non existant

Dans le cas où les comptes ne seraient pas fournis, soit à temps ou que le système ne convient pas suffisamment à l'implémentation recherchée avec Fit-In, il est important de trouver une méthode efficace pour leur création. Plusieurs possibilités sont envisageables, cependant avant toute chose il nous faut lister les informations nécessaires à un compte étudiant:

- Un identifiant
- La classe dans laquelle il se trouve

Ces deux informations sont obligatoires, via la classe nous pouvons retrouver l'école et le professeur. Ensuite un mot de passe n'est peut-être pas nécessaire si l'on décide d'utiliser des liens magiques.

3.2.2.1. Lien magique

Un lien magique, plus communément connu sous le terme Anglais Magic Link, est une méthode d'authentification. Le but est d'utiliser comme identifiant d'une application un email afin d'y recevoir un lien contenant un jeton encodé grâce à une clé privée. Ceci permet via un simple clic sur ce lien d'identifier une personne sans lui demander de mot de passe, uniquement son email.

Une autre information qui a été clarifiée avec le mandant, c'est que chaque année les informations de l'étudiant liée à sa classe sont remises à zéro. Cette information est valable le temps d'une année scolaire.

3.2.2.2. Flux dirigé par les étudiants

Lors de la première connexion annuelle, l'étudiant devrait choisir sa classe dans une liste. Il verrait en premier lieu une liste des écoles, puis en sélectionnant l'école il verrait ensuite une liste des classes et devrait choisir la sienne. Le professeur devrait ensuite approuver l'élève afin d'éviter les éventuelles erreurs.

3.2.2.2.1. Avantages

- Le travail est du côté des étudiants, le professeur ne doit que créer ses classes et attendre les demandes d'intégration de la part des élèves pour les valider.

3.2.2.2.2. Inconvénients

- Les étudiants peuvent se tromper de classe et ils doivent attendre sur le professeur pour accéder au contenu de l'application
- Plus d'informations concernant les étudiants seront nécessaires comme le nom et prénom afin que le professeur puisse les identifier dans la liste.

3.2.2.3. Flux dirigé par les enseignants

Les enseignants dans leur interface d'administration, en plus de pouvoir créer leur classe dans leurs écoles, peuvent y ajouter des élèves. Un simple email pourrait également être nécessaire si celui-ci est utilisé comme identifiant.

3.2.2.3.1. Avantages

- Pour l'élève c'est très simple, pas besoin de longue explication de la classe à choisir, l'école, etc.
- Pas d'attente d'acceptation de la part du professeur, une fois qu'on est ajouté, cela fonctionne.

3.2.2.3.2. Inconvénients

- Plus de travail une fois par an pour les enseignants qui doivent entrer la liste complète des élèves pour chacune de leur classe.

3.2.3. Solution choisie

Dans cette première version, la création de compte se fera via invitation du professeur, et donc via le flux dirigé par les enseignants. Combinée aux liens magiques cela évite la création d'un compte de la part d'un étudiant et permet à l'enseignant de ne pas être dépendant d'un système ou de devoir attendre sur un étudiant pour commencer à utiliser l'application et assigner des exercices à ses élèves.

En ce qui concerne l'API SAML peut-être dans un second temps une solution hybride pourrait être envisagée, ou une solution s'appuyant entièrement sur des données externes mais cela implique du travail du côté du centre de compétence ICTVS, pour le moment cette solution n'est pas envisageable.

L'avantage de la solution choisie, c'est qu'elle ne se limite pas qu'au Valais. Il suffit d'une adresse email et elle devient utilisable. Cette application pourrait donc être présentée à d'autres cantons.

3.3. Voir des fiches d'exercices

Tous les utilisateurs de l'application doivent pouvoir voir les fiches d'exercices. Cependant le but de cette lecture diffère selon les rôles.

3.3.1. Les élèves

Ils veulent voir les fiches afin d'effectuer les exercices.

3.3.2. Les professeurs

Ils veulent s'assurer des exercices qu'ils vont assigner à leurs élèves. Vérifier ce que contient la fiche.

3.3.3. Les contributeurs

Ils souhaitent vérifier qu'une fiche concernant un sujet similaire n'existe pas déjà.

3.3.4. Les validateurs

Ils veulent revoir des fiches existantes, vérifier qu'elles sont toujours d'actualité et qu'aucune erreur n'est passée entre les mailles du filet.

3.3.5. Les administrateurs

Les administrateurs ont tous les droits sur l'application, du coup automatiquement ils peuvent accéder aux fiches.

3.4. Réaliser un exercice

Seuls les élèves doivent pouvoir valider les exercices. Il est important qu'ils puissent transmettre le résultat de leurs exercices à l'enseignant. Ce résultat peut prendre les formes suivantes: vidéo, photo, auto-évaluation critériée ou un texte libre.

Il est possible que l'exercice nécessite la tenue d'une forme de journal de bord. C'est pourquoi il doit-être possible d'ajouter plusieurs rendus et de pouvoir les éditer. De cette manière, il serait possible de rendre plusieurs fois un contenu de type texte libre et ainsi de créer un journal de bord avant la soumission finale à l'enseignant.

L'élève doit pouvoir indiquer que l'exercice est selon lui totalement complété afin que l'enseignant l'évalue.

3.5. Recevoir une évaluation et échanger

Le cœur de l'application est de pouvoir échanger avec les professeurs, Il est donc important de pouvoir recevoir un feedback sur un exercice effectué lorsque l'on est un étudiant. Ce feedback de la part de l'enseignant peut-être formatif ou sommatif. C'est au professeur d'en décider.

Une fois l'évaluation reçue, le professeur peut ouvrir un canal d'échange avec l'étudiant s'il le souhaite. Ce canal d'échange peut être fermé à tout moment par l'enseignant.

3.6. Assigner des fiches

Les enseignants doivent pouvoir assigner des exercices spécifiques. Bien que les étudiants peuvent consulter toutes les fiches à tout moment et effectuer un travail volontaire s'ils le souhaitent, c'est aux professeurs de donner les priorités. Ils doivent pouvoir assigner un exercice à une classe entière, mais également être capables d'en assigner directement à certains élèves. Le but est de pouvoir assigner des exercices adaptés à chaque élève, peut-être qu'un élève a une blessure au bas du dos, c'est pourquoi l'enseignant évitera de lui donner des fiches de gainage frontal.

3.7. Créer des fiches

Les contributeurs doivent pouvoir créer des fiches. Ce sont des spécialistes dans leur domaine. Par exemple, une fiche dans la catégorie mentale sera créée par des psychologues tandis que des exercices physiques seront créés par des spécialistes du sport.

Une fois la fiche complète, le contributeur doit être capable de la soumettre pour approbation. Ce n'est pas le contributeur qui décide si la fiche peut-être publiée sur l'application Fit-In.

3.8. Approuver des fiches

Les fiches doivent être approuvées par les validateurs. Cette étape est essentielle pour éviter que des fiches contenant des fautes ou un contenu non approprié pour les élèves soit disponible via l'application Fit-In. Cette étape permet de garantir une meilleure qualité sur le contenu proposé.

A tout moment, ces approbations peuvent être révoquées par les validateurs. Si une fiche révoquée était assignée à des étudiants, l'enseignant doit-être tenu informé par email, ainsi que le créateur de la fiche.

Lors d'une révocation ou d'un refus d'approbation, les validateurs peuvent ajouter un message explicatif.

3.9. Assigner des rôles

Les administrateurs sont les utilisateurs qui peuvent affecter et révoquer des rôles pour d'autres utilisateurs.

4. STOCKAGE DES DONNÉES

Afin de déterminer où les données de l'application pourront être situées géographiquement, il est important de déterminer les contraintes légales.

4.1. Droit Européen

En Europe, le règlement général sur la protection des données, abrégé RGPD, définit des règles strictes concernant le type de données qu'une application peut stocker. Une section concernant les données relatives à des mineurs impose des procédures relatives au consentement des parents (General Data Protection Regulation - GDPR, 2021, Art. 8). Ce règlement européen est plus strict que la loi suisse sur la protection des données.

Par exemple, sous la RGPD nous ne pourrions stocker aucune information de mineur sans avoir le consentement d'un parent, et il faut également pouvoir vérifier que le parent est bien un représentant légal.

Pour le moment, la Suisse n'est pas soumise à la RGPD. Cela concernerait les entreprises suisses qui auraient des succursales en Europe ou offriraient des services en Europe (Réglementation UE pour la protection des données, 2017).

Si la décision était prise d'utiliser un hébergeur et des serveurs européens pour ce projets nous serions donc soumis à la RGPD.

4.2. Droit Suisse

En Suisses c'est la loi sur la protection des données, abrégée LPD, qui définit les règles cadres. Il est possible de transmettre des données à l'étranger si le pays garanti possède des lois assurant un niveau de protection au minimum similaire à la Suisse (Préposé fédéral à la protection des données et à la transparence - PFPDT, 2017).

Il est légalement possible d'utiliser des services européens pour l'hébergement de données de citoyens suisses si le pays cible atteint le niveau d'exigence requis en matière de protection des données.

4.2.1. Données médicales

Il se peut que l'application Fit-In sauvegarde des informations relatives à la santé des étudiants comme par exemple l'indice de masse corporel, abrégé IMC. L'IMC est considéré comme une donnée médicale étant donné qu'elle donne une information de l'état de santé d'une personne. C'est pourquoi il faut maintenant distinguer les données médicales dans l'équation de la gestion des données en Suisse.

La confédération a émis toute une série de mesures (Préposé fédéral à la protection des données et à la transparence - PFPDT, s.d.) à prendre en compte pour les données médicales. Ces données sont considérées comme nécessitant une protection toute particulière.

Les données médicales doivent être exclusivement stockées en Suisse afin de respecter le secret médical. Ce respect ne peut être garanti par des fournisseurs d'autres pays.

Le patient doit explicitement donner son consentement pour le stockage de ses données et doit garder le droit de rétracter ce droit en tout temps (Sébastien Fanti, 2019).

Selon un document de Sébastien Fanti (2017b, p. 92), avocat et notaire préposé à la protection des données et à la transparence du Canton du Valais, "Aucune donnée médicale ne devrait jamais être transmise à l'étranger ni sauvegardée dans un pays étranger".

4.3. Cadre scolaire

Dans le cadre scolaire c'est le secret de fonction (Guide Social Romand, s.d.) qui entre en compte lorsque l'on évoque les échanges entre les élèves et les enseignants. Un enseignant en tant qu'exécutant d'une tâche publique est tenu du secret de fonction. Il est important de noter que le secret de fonction n'est pas tenu face aux supérieurs hiérarchiques mais cependant il est présent d'un point de vue hiérarchique. Ce qui veut dire qu'un enseignant ne peut pas parler d'un secret confié par un élève à un autre enseignant.

La violation du secret de fonction est punie par la loi, d'une peine privative de liberté de trois ans au plus ou d'une peine pécuniaire. (CP art. 320)

La fin de l'emploi sous toute forme ne libère pas du secret de fonction. C'est pourquoi il est important d'année en année de supprimer les données des élèves ou lors de changement d'enseignant. (Code pénal suisse, s.d, Art. 320)

4.4. Décisions

Prenant en compte ces contraintes, il a été décidé avec le mandant de supprimer de l'application toutes les données médicales initialement prévues tel que l'IMC. L'application sera hébergée en Suisse car elle rentre dans le cadre scolaire, si possible au sein d'un serveur étatique.

Les autres données telles que les échanges entre les enseignants et les étudiants entrant dans le cadre du secret de fonction seront présentes sur l'application. Il sera important d'envisager un message de rappel à la loi aux enseignants lorsque ceux-ci consultent ces données.

5. RÉTENTION DES DONNÉES

Dans une politique de protection des données il ne suffit pas de savoir où et comment stocker ces données mais également combien de temps faut-il les conserver. Cette section répond à cette problématique.

Le choix est fait avec le mandant d'avoir la politique la plus stricte et irréprochable possible en termes de conservation de données.

5.1. Cycle de l'élève

Le cycle de l'élève reflète la durée de sa formation complète au secondaire, de son entrée à sa sortie. Il est partagé en 3 niveaux : 9-11H, secondaire I 1-3 et secondaire II 4-5. A l'intérieur d'un niveau, certaines données doivent être conservées d'une année à l'autre, tels que les exercices effectués, les points gagnés sur son "Heatlhy Score". Cependant, au terme d'un niveau, toutes les données doivent obligatoirement être supprimées.

5.2. Année scolaire

Chaque année scolaire est un nouveau commencement pour Fit-iN. Une grande partie des données doit être effacée. Principalement tous les échanges entre les étudiants et l'enseignant. Cependant, les progressions personnelles sur les exercices et les données accumulées via le "Healthy Score" doivent rester pendant tout le niveau.

5.3. Départ d'élève

Si un élève quitte une école, peu importe la raison, toutes les données doivent également être effacées. Il faut donc prévoir des mécanismes dans l'application qui, lors du retrait d'un élève d'une classe, supprime automatiquement toutes ses données.

5.4. Départ d'un enseignant

Si un enseignant part à la retraite, il change d'école ou autre, toutes les données échangées entre lui et ses élèves ne peuvent plus être conservées. Simplement parce qu'il ne doit plus être en capacité de les consulter.

5.5. Sur demande

Comme le précise la loi sur la protection des données (Préposé fédéral à la protection des données et à la transparence - PFPDT, 2013), toute personne peut exiger, sans fournir d'explication, que toute donnée relative à sa personne soit supprimée.

Pour notre application cela concerne principalement les échanges élèves/enseignants.

6. TECHNIQUE

Cette partie détaille les solutions techniques choisies et des explications qui ont mené aux décisions pour certains points jugés clés et intéressants pour ce rapport.

6.1. Architecture

L'application sera organisée de façon multi tiers, en l'occurrence plus précisément en modèle trois tiers soit:

1. **La couche de présentation**, qui contiendra la partie visuelle, c'est le niveau le plus haut de l'application il contient les interfaces qui reflète le fonctionnement du projet.

2. **La couche de traitement**, c'est la partie logique de l'application. Il effectue les demandes de la couche de présentation et s'occupe du transfert de données avec ses couches voisines (présentation et accès aux données).

3. **La couche d'accès aux données**, c'est la partie qui contient les données de l'application, le stockage. Via des requêtes de la part de la couche de traitement, ces données sont traitées et transférées si besoin à la couche supérieure.

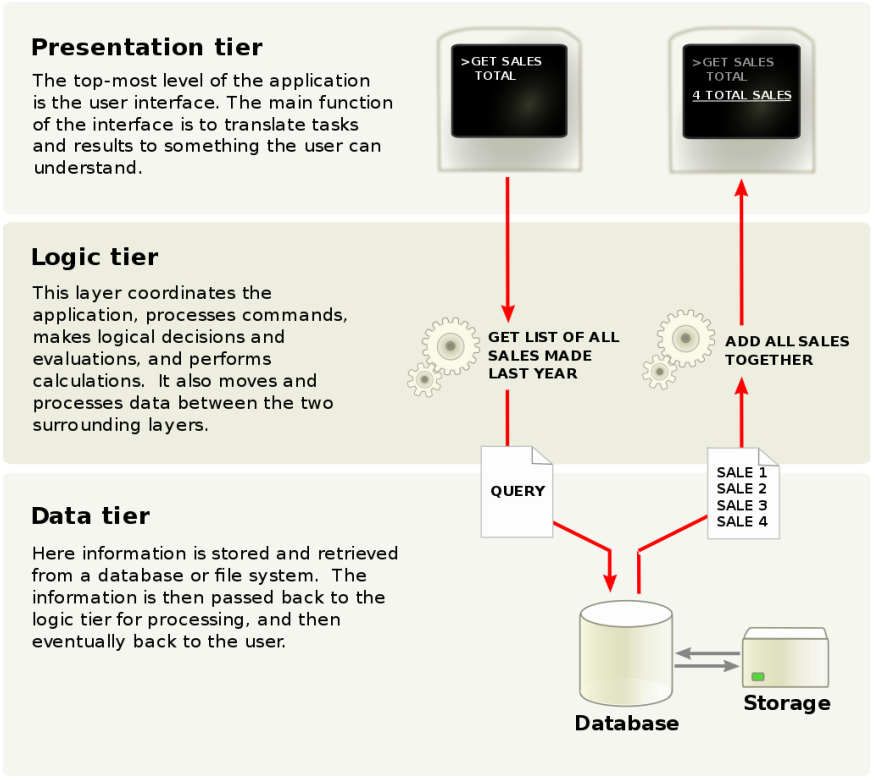

Figure 2: Architecture 3 tiers

Cette approche permet d'aborder le développement par étape, chaque partie a son utilité, on ne mélange pas les choses. Aussi en termes d'évolution, au fur des années s'il faut faire évoluer une technologie, il n'est donc pas nécessaire de tout changer, mais juste la couche concernée.

Le fait de séparer la couche frontend de la partie métier donne également la possibilité dans le futur de greffer une autre interface utilisateur comme par exemple une application mobile. Au même titre il serait simple de greffer à la partie métier une base de données externe ou un appel à une API externe si l'on décide un jour de se connecter avec un service du centre de compétence ICT-VS.

6.2. Technologies

6.2.1. Contraintes

Le centre de compétence ICT-VS n'a imposé qu'une seule contrainte technique, ne pas utiliser de PHP pour le développement de l'application.

6.2.2. Open Source

Tous les outils qui seront choisis pour le développement de cette application seront Open Source. Cette volonté est appuyée sur un article (Ben Balter, 2015) donnant six raisons qui devraient motiver ce choix. Dans le cas de ce projet, certaines des raisons nous concernent:

- **La transparence**, pour un outil qui touche à l'éducation, il est important de connaître les outils que l'on utilise, qu'il n'y ait pas de traceurs cachés par exemple.
- **Les meilleures pratiques**, souvent plus utilisées dans leur termes anglais "Best Practices" sont toutes les techniques considérées par la communauté comme étant les plus efficaces, justes et/ou sécurisées. Utiliser l'open source nous offre une meilleure garantie que les Best Practices soient suivies.
- **Réduction de l'effort**, en utilisant des outils adoptés par des millions de développeurs à travers le monde, il y a de plus grandes chances qu'il existe déjà un plugin ou module déjà codé qui fasse ce dont nous avons besoin.

6.2.3. Frontend

Pour la partie frontend plusieurs frameworks sont actuellement en tendance et très utilisés. Voici une liste non exhaustive:

- Angular
- React
- Vue

Les tendances Google nous donnent en général un bon indicateur des outils les plus utilisés dans une région. J'ai donc appliqué ces trois frameworks à ma recherche au niveau Suisse et obtenu le résultat suivant:

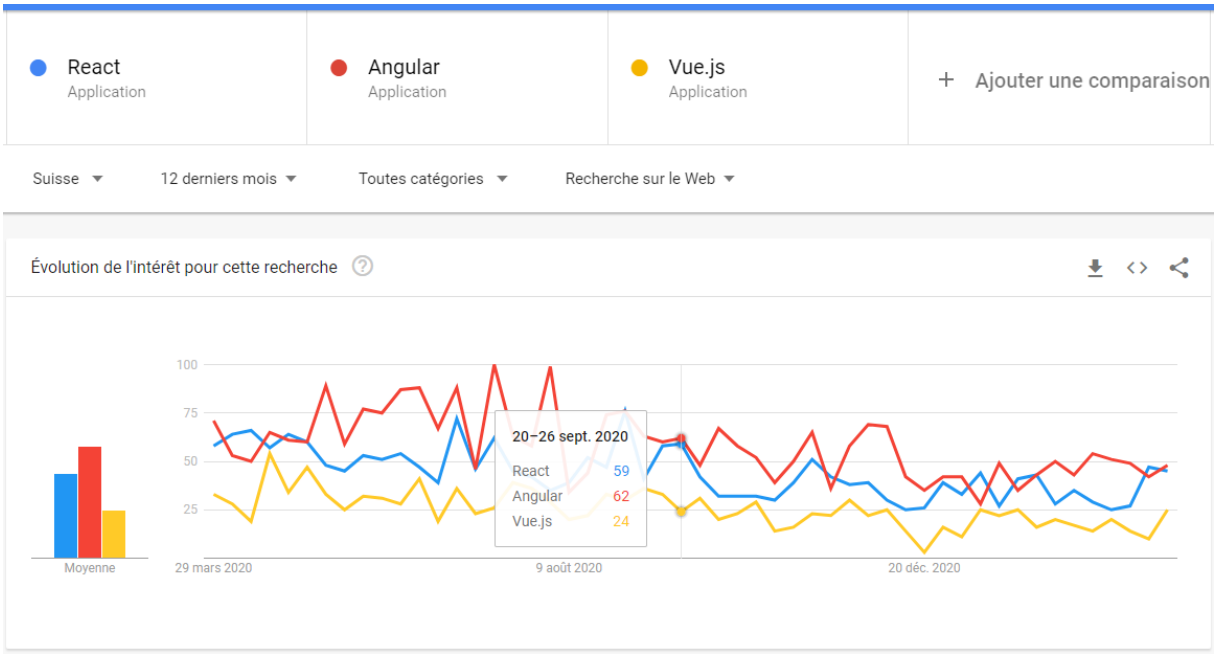

Figure 3: Trend frontend

Les courbes sont plutôt mitigées malgré une avance globale d'Angular, j'ai donc décidé d'approfondir ma recherche un peu plus. J'ai trouvé un comparatif plus récent (Shaumik Daityari, 2021) car datant du 15 mars 2021. Celui-ci arrive à la conclusion que Angular est clairement le plus mature des trois. Un de ses points négatifs cependant est sa courbe d'apprentissage. Il est également plus orienté TypeScript ce qui permet une meilleure rigidité niveau code. Cela permet également une cohérence de langage avec la partie backend.

Un autre article (formationjavascript.com, s.d.), qui parle aussi de vue.js mais très brièvement, résume bien la situation en expliquant qu'au final tout dépend du besoin mais que Angular est clairement décrit comme une solution plus complète, et qu'un projet avec React nécessite d'autres librairies pour arriver à une équivalence de structure telle que Angular.

Pour être certain du choix, j'ai également pris en considération une troisième analyse (Tomas Holas, 2017), qui explique qu'on ne peut pas comparer un chat et un chien. Angular et React sont deux choses totalement différentes. Cependant l'article parle de la combinaison React/Redux et du côté données immutables et unidirectionnelles qui est souvent mis en avant pour React. Or il est également possible de faire la même chose et Angular tend vers ceci. NGRX[1] est une librairie pour Angular qui permet justement de travailler en état réactif tout comme React.

Mon expérience personnelle est d'environ six ans sur Angular. En premier lieu appelé Angular.js puis Angular 2 en 2016. J'ai fait quelques projets sur React mais rien à l'échelle de ce que j'ai produit avec Angular. Je trouve Angular plus solide en termes de structure et de la manière dont un projet évolue avec le temps. Je n'ai aucune expérience avec vue.js, et si l'on prend en compte la maturité du FrameWork, bien que très prometteur, il me semble injustifié de le choisir.

React est un excellent outil également, souvent associé à Redux, un outil pour gérer des états réactifs. Cependant il existe NGRX, un outil qui permet d'utiliser les états réactifs dans Angular.

6.2.3.1. Décision

Pour le frontend le choix se portera sur Angular car c'est une solution complète qui structure le projet dès le début et comme les articles le mentionnent, il est clairement recommandé comme outil pour un nouveau projet. Ajoutant ceci à mon expérience personnelle, cela en fait un choix évident. Angular sera combiné avec la librairie NGRX afin de prendre avantage des états réactifs qui font la force d'un outil tel que React.

6.2.4. Backend

Pour la partie métier, une infinité de solutions s'offrent à nous. Cependant, je souhaite partir sur une solution légère. Je ne souhaite pas d'application nécessitant une compilation ni une application ayant besoin de beaucoup de ressources pour tourner. Pour moi, un projet tel que Fit-iN doit pouvoir tourner sur une machine avec le moins de ressources possibles.

Personnellement, cela fait plusieurs années que je développe professionnellement sur Node.js, dans un premier temps en JavaScript, puis en TypeScript lorsque celui-ci eût atteint sa maturité. Je travaille sur des applications de type SAAS à haute disponibilité. Je vois de manière journalière des applications développées en Node.js encaisser des trafics atteignant des milliards de requêtes par minute. Certes, l'infrastructure y est pour beaucoup mais le fait que le langage et la plateforme puissent s'étendre à de telles dimensions prouve son efficacité.

[1] https://ngrx.io/

Maintenant, il existe une multitude de Frameworks pour node.js, Express est le plus connu, beaucoup de Frameworks se basent sur lui. Meteor, Socket, tails, etc, il en existe énormément. Ceci montre à quel point node.js est devenu populaire ces dernières années.

Certes, des langages comme Java sont toujours plus utilisés, mais il faut prendre en compte le côté historique de certaines applications et la difficulté qu'ont certaines entreprises à passer le cap de la rupture technologique.

Cependant, on remarque depuis plus de quinze ans une tendance à la baisse pour des langages comme Java, ce qui ne le rend pas désuet, mais dans le cas de ce projet ce n'est pas le genre d'outil qui nous intéresse.

Figure 4: Trend Java

6.2.4.1. Décision

Afin d'avoir une cohérence entre le backend et le frontend, Node.js sera utilisé pour la partie backend avec le langage TypeScript. Pour le Framework c'est mon expérience professionnelle qui m'a fait pencher pour Nest.js, qui est selon moi le plus mature et robuste des FrameWorks présents sur le marché pour TypeScript sur Node.js. Nest.js guide également le développement selon la pratique appelée "Convention plutôt que configuration" (S M Asad Rahman, 2019). Cette pratique vise, par la mise en place de conventions, à réduire le temps passé à prendre des décisions par un développeur.

6.2.5. Base de données

Plusieurs critères doivent être pris en compte lors du choix de la base de données. Doit-elle être relationnelle? Quel type de requête allons-nous principalement gérer? Le type de charge? Quel niveau de sécurité existe-il? Pour ce faire nous allons nous appuyer sur plusieurs comparatifs.

Dans un premier temps, attaquons-nous aux plus populaires des bases de données de 2020 (Sadissa Babeni, 2020). Tout comme pour les langages, le but est de prendre uniquement des technologies Open-source. Très rapidement une liste limitée apparaît:

- MySQL
- MariaDB
- MongoDB
- Redis
- PostgresSQL

Dans cette liste il y a trois bases de données relationnelles (MySQL, MariaDB et PostgreSQL) et deux autres non relationnelles et très différentes l'une de l'autre.

6.2.5.1. Redis

Redis est une base de données qui fonctionne en mémoire. Cela fait d'elle probablement la plus rapide des bases de données existantes sur le marché. Redis est extrêmement utilisé dans le caching mais cela ne veut pas dire que c'est l'unique chose que cet outil peut faire. Cependant, le fait que Redis ne gère pas des jointures dans les requêtes et ne permet pas une structure aussi rigide qu'une base de données relationnelle fait que ce n'est pas le bon choix pour ce projet.

6.2.5.2. MongoDB

C'est la base de données qui a rendu populaire le non-relationnel (NoSQL, Not Only SQL). Sa rapidité et ses performances sont indéniables. Le NoSQL a de clairs avantages dans le développement moderne. Cependant MongoDB est aussi très gourmand en mémoire et ne permet pas une rigueur stricte au niveau des schémas comme le permettrait une base de donnée relationnelle.

L'analyse de ces deux bases de données permet donc rapidement de les éliminer de la sélection. Il faut donc a priori se pencher sur une solution orientée relationnelle.

6.2.5.3. MySQL

Sortie en 1995, elle représente certainement la base de données la plus populaire du web. Encore très largement la base de données utilisée sur des applications web, cependant de plus en

plus en déclin face à l'arrivée d'outils extrêmement performants et concurrentiels. Un coup d'œil sur Google trend permet de confirmer cette tendance à la baisse depuis plus de quinze ans désormais.

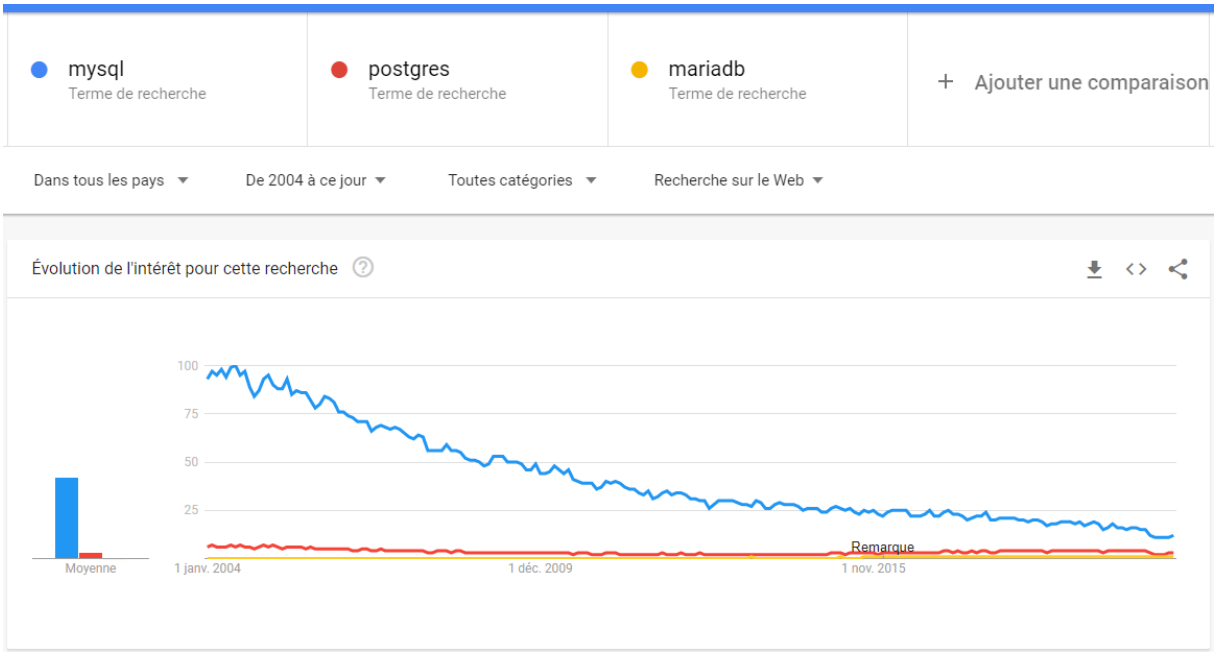

Figure 5: Trend Base de données

MySQL est très performant et permet la gestion de grandes bases de données. Il manque de support pour le XML ou OLAP mais, dans notre cas, cela n'est pas pertinent. MySQL peine un peu en termes de gestion de la concurrence des requêtes (Krasimir Hristozov, 2019).

Cependant MySQL est très simple de prise en main et compatible avec la plupart des hébergeurs. C'est donc un choix de force à considérer.

6.2.5.4. MariaDB

MariadDB fut créé par le fondateur de Mysql dans le but de remplacer ce dernier (Michael "Monty" Widenius, 2019). Ce système est donc très récent, il date de 2009. Très performant, cette base de données à un système qui très souvent corrige automatiquement les types de données envoyées, contrairement à un système comme PostgresSQL plus rigide de ce côté-là. Cependant MariaDB a des fonctionnalités telles que le support Master-Master qui est très intéressant pour des applications à très grande échelle (optimizdba.com, s.d.).

6.2.5.5. PostgresSQL

Créé en 1996, PostgreSQL est considéré aujourd'hui comme la base de données la plus performante du monde (Kirk Roybal, 2020). PostgreSQL est de type objet-relationnelle, ce qui lui permet grâce à son côté orienté objet d'avoir des systèmes d'héritage directement supportés via le schéma. Cet outil est hautement ajustable (scalable en anglais). Son plus grand défaut est sa

difficulté à configurer et préparer ses schémas. Il est moins facile à prendre en main que d'autres outils. Cependant l'installation par défaut de PostgresSQL est souvent qualifiée de meilleure que celle par défaut de MySQL. Le support de type de données est largement plus grand avec PostgresSQL.

6.2.5.6. Décision

Avant toute chose, le choix de la base de données n'est pas motivée par le choix du framework, Nest.js. Nest.Js est agnostic à la technologie de base de données utilisée (nestjs.com, s.d., database), il fonctionne aussi bien avec des bases de données relationnelles que son contraire. Donc le framework ne sera pas un critère de sélection.

Malgré les grandes qualités de MariaDB, le manque de communauté et de maturité de la technologie dû à son âge fait qu'il ne sera pas le bon choix pour ce projet. Le choix est donc entre MySQL et PostgreSQL. Les tendances et les articles tendent à montrer que les gens se tournent vers PostgresSQL.

La rigidité des schémas permet une très haute garantie en termes de qualité des données stockées. Pour moi, c'est un point essentiel à tout système d'information: si les données sont de qualité, il est toujours possible de reconstruire par-dessus si quelque chose ne va pas.

Un des points négatifs est sa difficulté d'apprentissage pour la configuration, mais de ce côté-là j'ai personnellement plusieurs années d'expérience avec PostgresSQL qui combleront ce point.

C'est donc **PostgreSQL** qui sera le système de base de données de Fit-iN, son vaste choix de types et la possibilité de pousser la création de schémas et les dépendances très loin me garantissent le fait que je ne serai pas limité lors de la création de ma structure de données.

6.3. Méthodologie et bonne pratiques

Le site web https://12factor.net/fr/ "The Twelve-Factore App" liste les douze facteurs à prendre en compte et à suivre lors du développement d'une application de service. Personnellement, j'applique professionnellement ces règles depuis plusieurs années. Le but de cette méthodologie est d'organiser son application d'une manière qui permettra de garantir un niveau de sécurité par désign, à ralentir le coût de la détérioration du logiciel avec les années, aussi appelé dette technique. Plusieurs points feront référence à ces douze facteurs dans ce document.

6.4. Déploiement continu

Le déploiement continu contient des pratiques telles que l'intégration continue. Cette dernière est une pratique qui permet d'automatiser une grande partie des processus permettant la garantie de la qualité. Par exemple s'assurer qu'il n'y a pas de régression dans le code en lançant tous les tests à chaque changement du code source.

Le déploiement continu va plus loin et s'occupe également de la construction et du déploiement de l'application sur les serveurs. Cette pratique vise à augmenter la vitesse à laquelle les développeurs mettent en place des mises à jour. Toutes ces pratiques font partie de ce que l'on appelle l'extrême programming, une méthodologie agile (extremeprogramming.org, 2013). Cela rentre donc en parfaite adéquation avec la manière de travailler sur ce projet. Cela garantit aussi une plus grande facilité à transmettre ce projet et les codes sources à une nouvelle équipe dans le futur. Ils sauront que tout est en place pour le déploiement et qu'aucun code non testé ne passera l'étape du déploiement continu si celui-ci échoue.

En ce qui concerne les outils il en existe plusieurs, en voici une liste non exhaustive:

- CircleCI
- Bamboo
- Jenkins

Beaucoup de plateformes s'occupant de la conservation et versionnage du code proposent leur propre outil d'intégration continue. Pour ce projet, je cherche un outil gratuit en libre service qui ne nécessite pas d'installation. C'est pourquoi je vais m'orienter vers CircleCi que j'utilise depuis plusieurs années. De plus, celui-ci est complétement compatible avec Docker et la configuration se fait à l'aide d'un fichier YAML (CircleCi Docs, 2021) dans le code source, ce qui garantit un versionnage et une flexibilité si un jour un autre outil devait être utilisé.

6.5. Stockage du code

C'est le facteur numéro un selon https://12factor.net/fr/codebase. Il est important d'avoir un suivi de version qui permette des retours en arrière et la possibilité de travailler à plusieurs d'un même code source. Malgré le fait que je sois le seul développeur du projet, il est important d'avoir un outil de cette sorte.

En équipe ce genre d'outil permet également la revue de code par les autres membres de l'équipe avant de procéder à son acceptation dans la base de code principale. Cette base de code doit être unique.

Une base unique ne veut pas dire un seul déploiement. Il sera possible de déployer ce code dans plusieurs environnements si nécessaire tel que test ou production par exemple.

Aujourd'hui un outil domine clairement le marché, il s'agit de GIT. Git est un outil de gestion de version. Plus jeune qu'un de ses anciens concurrents SVN, créé par apache, il a été créé par le fondateur du noyau Linux, Linus Torvalds en 2005. Il est l'outil de versions le plus utilisé dans le monde par douze millions de personnes (Marius Nestor, 2016).

Il est possible d'héberger soi-même sa solution GIT, mais à nouveau il est important de savoir externaliser certaines parties du projet. Cela évite d'avoir à se soucier de l'aspect sécurité entre autres et de la qualité du service. Le leader mondial est https://github.com/

GitHub permet d'avoir des projets privés, ce qui garantit que le code source n'est pas exposé. C'est l'outil qui sera utilisé pour ce projet.

6.6. Parité dev/prod

Le dixième facteur de la méthodologie https://12factor.net/fr/dev-prod-parity parle de la parité entre les différents environnements de l'application. Par environnement il faut entendre la production, ou pré-production ou encore l'environnement de test. Il peut y en avoir une infinité, mais en général il y en a au moins deux, production et pré-production souvent appelés staging.

Pendant longtemps les disparités entre les environnements étaient nombreuses. Entre les différences de configuration ou les modules manquants, nombreuses furent les raisons pour lesquelles une application fonctionnait sur un environnement mais pas un autre.

Docker, outil arrivé sur le marché en 2013, a résolu tous ces problèmes et a complètement révolutionné la façon d'aborder le développement logiciel. Avant toute chose, il faut savoir ce qu'est Docker. C'est un logiciel libre de conteneurisation, à ne pas confondre avec la virtualisation (William G. Wong, 2016), qui permet rapidement de déployer des applications. Docker permet la création de conteneurs contenant toute la configuration d'un système. Chaque conteneur est complètement isolé des autres et fonctionne par couche. Chaque couche est gardée en mémoire et tant que les couches inférieures ne changent pas (semaphoreci.com, s.d.), Docker ne changera que la couche qui contient une modification ainsi que celle supérieure à ceci. Prenons un exemple:

Pour un conteneur donné, j'ai les couches suivantes:

1. L'installation du système d'exploitation
2. L'installation des paquets liés au système
3. L'installation des modules de mon application
4. L'ajout du code source de mon application
5. Le démarrage de mon application

Les étapes 1 à 3 sont clairement les plus lentes du processus, cependant si je décide de changer le code source de mon application, Docker va uniquement reconstruire les couches 4 et 5 du conteneur.

Cet outil est très puissant et j'ai eu la chance de travailler avec certains développeurs actifs à son évolution dès 2015 moins de deux ans après sa sortie. Pendant longtemps la société pour laquelle je travaillais à l'époque à Londres fut l'entreprise Londonienne avec la plus grande architecture docker en production (Nicolas Trésegnie, 2019). J'ai pu suivre l'évolution également de tous les outils AWS (Amazon Web Service) et utiliser la puissance de docker. Aujourd'hui je ne pourrais concevoir le développement d'applications sans Docker: cela fait selon moi, tout comme un outil comme GIT, partie des choses minimales requises pour tout développeur moderne (David Currie, 2017).

6.6.1. Limitation d'hébergement

Il est possible que je fasse face à une certaine limitation de la part des hébergeurs potentiels pour ce projet. Par exemple, peut-être qu'il ne sera pas possible d'avoir une base PostgresSQL. Docker permet également de pallier ceci. Très souvent des images officielles de conteneur existent pour les grands outils du monde du développement. Il est donc possible de créer un conteneur Docker avec PostgresSQL à l'intérieur[2].

6.7. Hébergement

La question de l'hébergement est certainement une des plus complexes pour ce travail. Les plateformes cloud sont clairement ce vers quoi il faut se tourner pour des applications de service moderne. De plus, les hébergeurs modernes facturent les services cloud selon l'utilisation et il est donc possible de maximiser l'élasticité des applications au maximum afin de ne pas dépenser un sou pendant les heures creuses mais d'être capable d'encaisser de lourdes charges lors des pics.

Aujourd'hui il existe trois grands hébergeurs cloud:

- Amazon Web Services
- Google Cloud
- Microsoft Azure

Même si Azure propose aujourd'hui des hébergement linux, à ce jour il est surtout privilégié pour les technologies Microsoft telles que C# par exemple. Je pense qu'il faut le mettre de côté pour tout ce qui n'est pas lié aux technologies Microsoft, cependant il est possible qu'Azure dans le futur rattrape son léger retard et devienne concurrentiel face aux deux autres géants.

Aujourd'hui, le grand leader du marché est AWS, loin devant son concurrent le plus proche, en termes de services équivalents, Google Cloud (Felix Richter, 2021). En effet AWS, avec 32% des parts de marché contre 20% pour Azure et 9% pour Google Cloud, est leader du marché pour une

[2] https://hub.docker.com/_/postgres

bonne raison. Nombreux sont les articles en ligne comparant les trois (Scott Carey, 2020), et très souvent (Nick Underwood, 2021) la conclusion est simplement que AWS est juste trop gros et beaucoup trop en avance, autant en termes de service que de prix. AWS propose un free-tier qui permet quasiment d'avoir accès à la majorité des services, avec une certaine limitation, gratuitement pendant un an (aws.amazon.com, s.d.).

Cela fait environ sept ans que j'utilise exclusivement AWS professionnellement, même si je n'ai jamais passé de certification car je n'en ai jamais eu le besoin professionnellement, je pense avoir un niveau de connaissance plutôt confirmé. Je suis loin de tout connaître, je laisse ceci aux experts en infrastructure, mais pour un développeur j'estime avoir un niveau de connaissances suffisant pour tirer certaines conclusions.

A priori le choix pourrait paraître tout tracé, mais étant donné qu'il nous faut héberger en Suisse, tout se complique. En effet, AWS n'a pas de data center en Suisse. Ils ont prévu d'ouvrir une zone, celle-ci devrait être disponible au deuxième semestre 2022 et sera hébergée dans la région de Zurich (Werner Vogels, 2020).

L'application étant prévue pour la rentrée scolaire de 2022, il serait donc envisageable plus tard de choisir comme hébergement final AWS, mais dans tous les cas il faut une solution en attendant tout cela, et de préférence une solution gratuite.

Il existe une solution nommée Jelastic[3] qui permet de créer une plateforme de service Cloud. Cette solution est utilisée par plusieurs hébergeurs Suisse tels que Infomaniak (infomaniak.com, s.d., Jelastic Cloud) ou encore Hidora (Yulia Shevchenko, 2017). Il s'avère que j'ai la possibilité d'avoir un contact direct avec le fondateur d'Hidora, cependant le mandant a également besoin d'autres services tels que l'hébergement d'adresses email entre autres, et infomaniak est très complet à ce niveau. De plus, Infomaniak est moins cher: pour la configuration minimale, infomaniak est à 1.40CHF par mois, contre 2.20 CHF pour la même configuration chez Hidora. Sinon, d'un point de vue technologie, Hidora supporte le stack requis, soit Docker et Postgres. Infomaniak propose également les mêmes technologies et nous permet de supporter complètement notre stack technique. Dans le cas où l'hébergeur ne supporterait pas Postgres il serait toujours possible de créer un conteneur Docker Postgres, sachant qu'avoir Postgres dans un conteneur plutôt que de l'avoir en natif ne veut pas forcément dire de moins bonnes performances (Petr Jahoda, 2021).

Le choix se porte donc sur infomaniak, la configuration de jelastic fait l'objet d'un chapitre dédié.

[3] https://jelastic.com/

6.8. Configuration

Un des douze facteurs touche la configuration https://12factor.net/fr/config. Il est important de préciser ce que cela implique. La configuration est tout ce qui n'implique pas le code, mais qui est une valeur qui change d'un environnement à un autre. Quelques exemples: une clef d'API, une URL ou encore les accès à une base de données.

Il est très fréquent de voir ce genre de configurations stockées dans des fichiers. Par exemple pour le système WordPress c'est un simple fichier à la racine du serveur (wordpress.org, s.d.) qui contient les informations de connexions à la base de données. Les douze facteurs sont contre cette pratique car elle comporte plusieurs risques. En effet, souvent ces valeurs se retrouvent sur les systèmes de contrôle de version tel que GIT. Si par mégarde des accès étaient changés et le code rendu public, c'est tous les accès qui tomberaient en même temps.

La méthodologie demande une séparation stricte entre la configuration et le code, la configuration doit être stockée dans des variables d'environnement. Il est conseillé de se demander si l'on pourrait rendre le code source de l'application publique afin de savoir si la séparation entre la configuration et le code a bien été effectuée.

Jelastic, la technologie derrière le système d'hébergement cloud pour gérer et déployer des images docker, permet très facilement d'ajouter et de modifier des variables d'environnement comme l'explique leur documentation (docs.jelastic.com, s.d.).

6.9. Dépendances

Encore un point traité par la méthodologie https://12factor.net/fr/dependencies. Il est important de déclarer de manière claire les dépendances d'une application. L'avantage avec le fait que le langage pour le frontend et le backend soit basé sur les mêmes technologies fait que nous avons accès à NPM (Node Package Manager). Celui-ci permet très facilement d'installer des dépendances qui pourront être utilisées dans notre projet. Ces dépendances sont ensuite stockées dans un fichier à la racine du projet appelé package.json (docs.npmjs.com, s.d.).

Un des avantages à avoir une déclaration explicite des dépendances est qu'il est très simple pour un développeur découvrant le projet de voir la liste de celle-ci. Cela évite énormément de temps de recherche et de découverte du projet.

6.10. Conventions de codages

Afin de garantir une structure saine dans le code, il est important de définir des conventions de codage et de nommage claires. Afin de s'assurer qu'elles soient appliquées, ESLint [4]sera utilisé.

[4] https://eslint.org/

ESLint (eslint.org, s.d.) est un outil qui parcourt le code source et vérifie, selon des règles prédéfinies, s'il contient des erreurs. Par exemple, le fait de ne pas autoriser les doubles guillemets par exemple générerait une erreur s'il en trouve. ESLint permet également de fixer automatiquement la plupart de ces erreurs.

Un outil tel que Husky[5] permet de lancer des commandes lors de certaines opérations liées à GIT. Dans notre cas, lors d'un commit avant de pouvoir pousser le code sur le système de version, on souhaite s'assurer que ESLint n'a détecté aucune erreur. Ceci est mis en place grâce à quelques lignes ajoutées dans le fichier package.json qui définit notre paquet NPM à la racine des projets.

Voici les quelques lignes:

```
{
  "husky": {
    "hooks": {
      "pre-commit": "lint-staged"
    }
  },
  "lint-staged": {
    "*.ts": "eslint --cache --fix --max-warnings 0"
  }
}
```

6.10.1. Liste des règles

Le projet utilise par défaut les règles recommandées par ESLint trouvables ici: https://eslint.org/docs/rules/. Cependant certaines règles spécifiques à notre projet sont ajoutées, en voici quelques exemples:

```
"max-len": ["error", { "code" : 120 }]
```

Limite la longueur d'une ligne à 120 caractères. Cela permet une meilleure lisibilité dans l'éditeur et lors des relectures de code d'autres développeurs sur GitHub.[6]

```
"new-cap": 0
```

[5] https://typicode.github.io/husky/

[6] https://eslint.org/docs/rules/max-len#enforce-a-maximum-line-length-max-len

Désactivation de la règle forçant les noms des constructeurs à commencer par une majuscule.[7]

```
"comma-dangle": ["error", "always-multiline"]
```

Oblige l'ajout d'une virgule à la fin du dernier élément dans un objet ou un tableau. Dans notre cas uniquement si celui-ci est écrit sur plusieurs lignes. Cela permet, lors de l'affichage de la différence entre deux versions de code, de n'avoir qu'une ligne changée au lieu de deux si la dernière venait à être modifiée.[8]

Il y environ une vingtaines de règles personnalisées pour ce projet, en plus des règles d'ESLint par défaut. Ces règles ne sont pas créées au hasard, elles sont là pour refléter les bonnes pratiques en termes de codage et suivre les lignes de conduite de nest.js ainsi que d'angular. Ces lignes de conduite sont basées sur le style et les exemples fournis dans la doc respective de ces deux outils. https://docs.nestjs.com et https://angular.io/guide/styleguide

[7] https://eslint.org/docs/rules/new-cap

[8] https://eslint.org/docs/user-guide/migrating-to-6.0.0#the-comma-dangle-rule-is-now-more-strict-by-default

7. MOCKUPS

Afin de mieux comprendre le processus et de permettre un meilleur échange avec le mandant, une série de mockups a été proposée afin de démontrer les processus d'utilisation de l'application. L'accent a été mis sur la partie étudiante ainsi qu'un petit peu sur la partie enseignante qui est le cœur de l'application. Ces mockups ne représentent pas ce que sera l'application définitive mais ils ont permis de clarifier plein de fonctionnalités ainsi que le type d'information à afficher.

Cette étape a été essentielle autant pour moi que pour le mandant, elle a permis d'avoir une première image de l'application, d'assurer au mandant que sa vision était réalisable et elle m'a permis de comprendre ce qu'il souhaitait que je réalise.

Il a été décidé, après les versions présentées ci-après qu'il n'y aurait pas plus de travail effectué sur ces mockups, qu'il y avait suffisamment de matière pour démarrer le projet et qu'il sera possible d'orienter et d'améliorer les interfaces au fur et à mesures des itérations.

L'outil utilisé pour créer ces mocks est https://www.figma.com/. Il est gratuit et fonctionne entièrement via un navigateur internet, ne nécessite aucune installation et tous les fichiers sont directement sauvegardés sur votre compte. Il est possible de collaborer à plusieurs sur cet outil, ce qui m'a permis de transmettre des accès au mandant lors de nos réunions et d'échanger sur les mockups avec lui directement.

7.1. Identification des utilisateurs

Cette partie est commune à tous les utilisateurs, elle démontre comment les liens magiques fonctionnent.

7.1.1. Vue globale

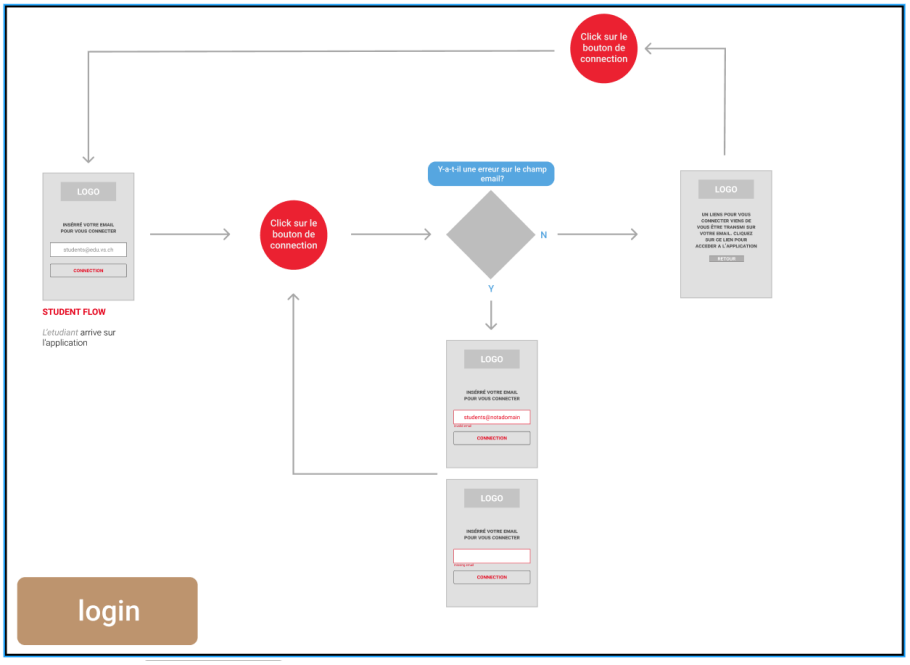

Figure 6: Flux de connexion

7.1.2. Écran de connexion

Grâce aux liens magiques un simple champ pour l'email et un bouton de connexion suffisent.

Figure 7: Ecran de connexion

7.1.3. Erreur de saisie à la connexion

Si le champ contient une erreur ou s'il est laissé vide, voici des exemples de messages d'erreurs de saisie.

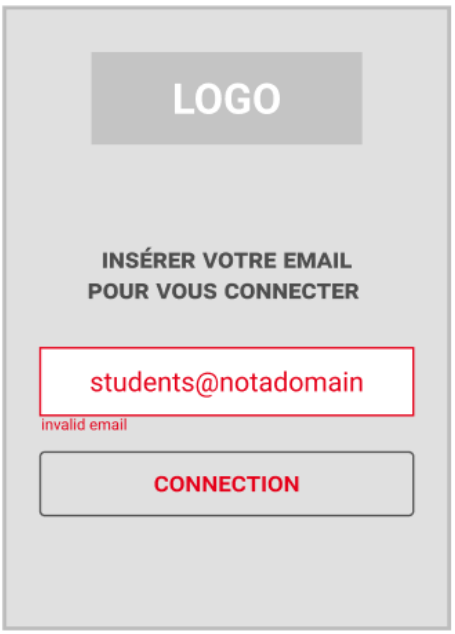

Figure 8: Erreur de connexion

Figure 9: Connexion email manquant

7.1.4. Écran de confirmation

Ici juste un message pour expliquer à l'utilisateur d'aller lire ses emails afin de cliquer sur le lien lui permettant de se connecter à l'application.

Figure 10: Confirmation de connexion

7.2. Mémoire de l'utilisateur

Lorsque l'utilisateur se connecte pour la première fois, il a la possibilité de laisser l'application se rappeler de lui, cela lui évitera de devoir refaire le processus précédent pendant une période de temps qui sera précisée en temps voulu.

Figure 11: Se rappeler de moi

7.3. Navigation étudiante

7.3.1. Vue globale

Cette partie décrit le processus d'utilisation de l'application du point de vue d'un étudiant.

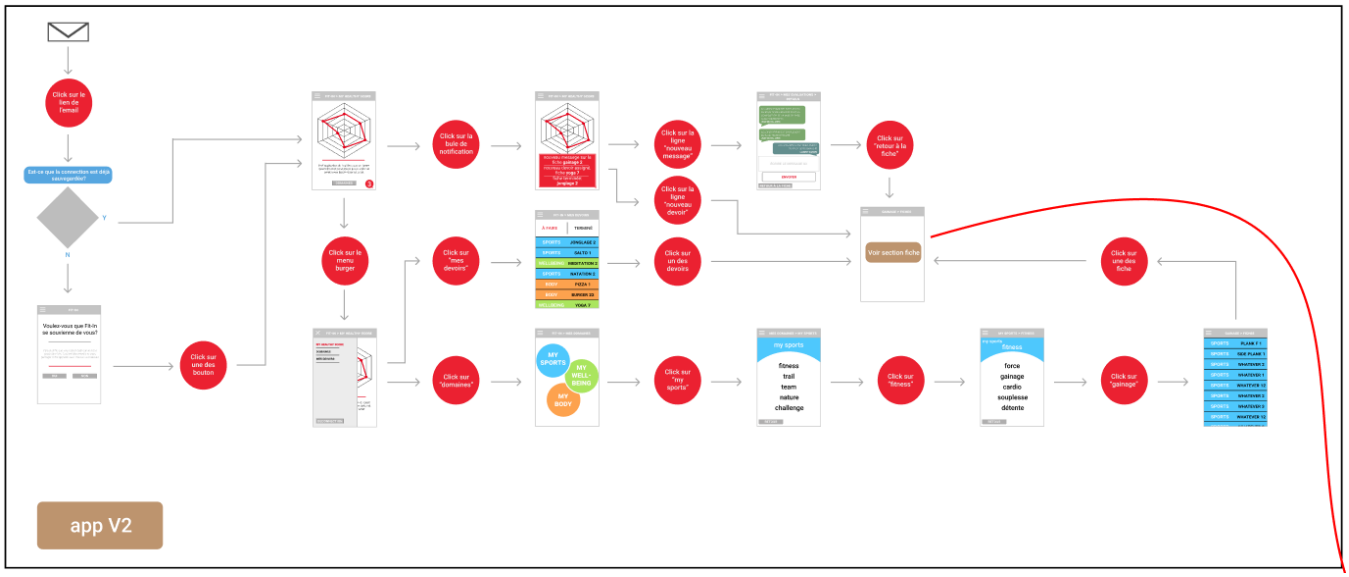

Figure 12: Navigation étudiante

7.3.2. Écran d'accueil

L'accueil contiendra un graphique représentant la progression de l'élève avec son "Healthy score". Les dernières notifications ainsi que les domaines sont directement accessibles sans avoir à cliquer sur le menu.

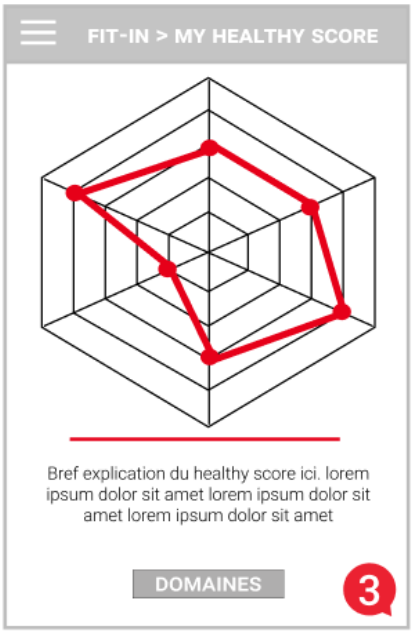

Figure 13: Score de bien-être

7.3.3. Menu

Un menu latéral sera disponible au clic de l'icône ou au mouvement en glissant avec le doigt comme une application mobile. Pour l'élève, trois sections sont visibles.

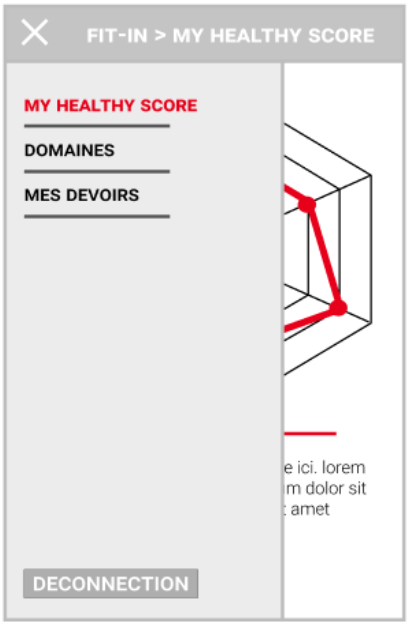

Figure 14: Menu latéral

7.3.4. Notifications

Lorsque l' élève clique sur l'icône avec les notifications, il peut directement voir les derniers événements qui se sont déroulés. Il a la possibilité de cliquer dessus pour accéder aux éléments concernés directement.

Figure 15: Notifications

7.3.5. Section devoirs

Un étudiant doit pouvoir à tout moment identifier les exercices qui lui sont assignés. Il peut trier par ceux qui sont terminés ou ceux qui sont à faire.

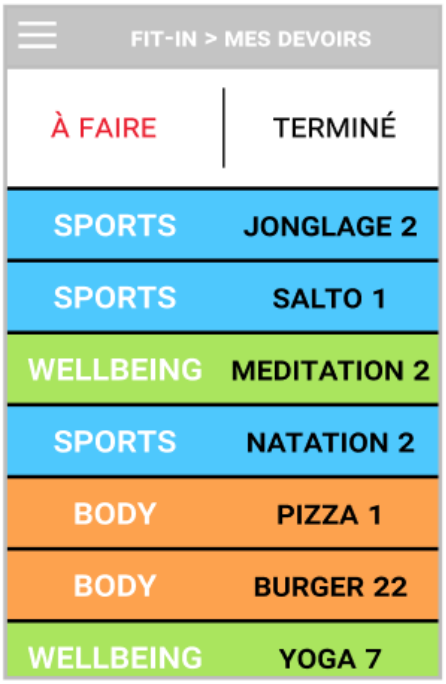

Figure 16: Devoirs

7.3.6. Section domaines

L'application contient trois domaines principaux. Ceci est l'écran permettant de choisir l'un d'entre eux.

Figure 17: Mes domaines

7.3.7. Catégories

Une fois arrivé dans un domaine, il nous faut choisir la catégorie désirée.

Figure 18: Mes thèmes

7.3.8. Sous catégories

Une fois la catégorie choisie, il reste à sélectionner la sous-catégorie.

Figure 19: Mes sous-thèmes

7.3.9. Liste des fiches

Les fiches sont toutes accessibles une fois que l'utilisateur est dans la sous-catégorie correspondante.

Figure 20: Mes exercices

7.3.10. Discussion élève-enseignant

Les professeurs ont la possibilité pour chacune des fiches d'ouvrir un canal de discussion avec l'élève afin d'échanger sur l'exercice. Il est donc possible d'accéder à un écran comme celui-ci.

Figure 21: Discussion

7.4. Fiches

Le point central de l'application est constitué par les fiches. Celles-ci ont eu droit à leur propre analyse de processus et leurs propres mockups. Après plusieurs itérations, une idée de fonctionnement fut trouvée et servira de base de départ pour la création de l'application. De nombreuses itérations auront un effet sur les fiches d'ici la fin du projet, mais ces mockups ont permis d'établir toute la logique et les données nécessaires au démarrage de la partie technique.

Les fiches utilisent une terminologie anglophone selon le choix du mandant.

7.4.1. Vue globale

Malheureusement pas très visible sur ce document, l'on peut néanmoins voir que plusieurs étapes et sous-étapes forment ensemble le processus nécessaire au fonctionnement des fiches

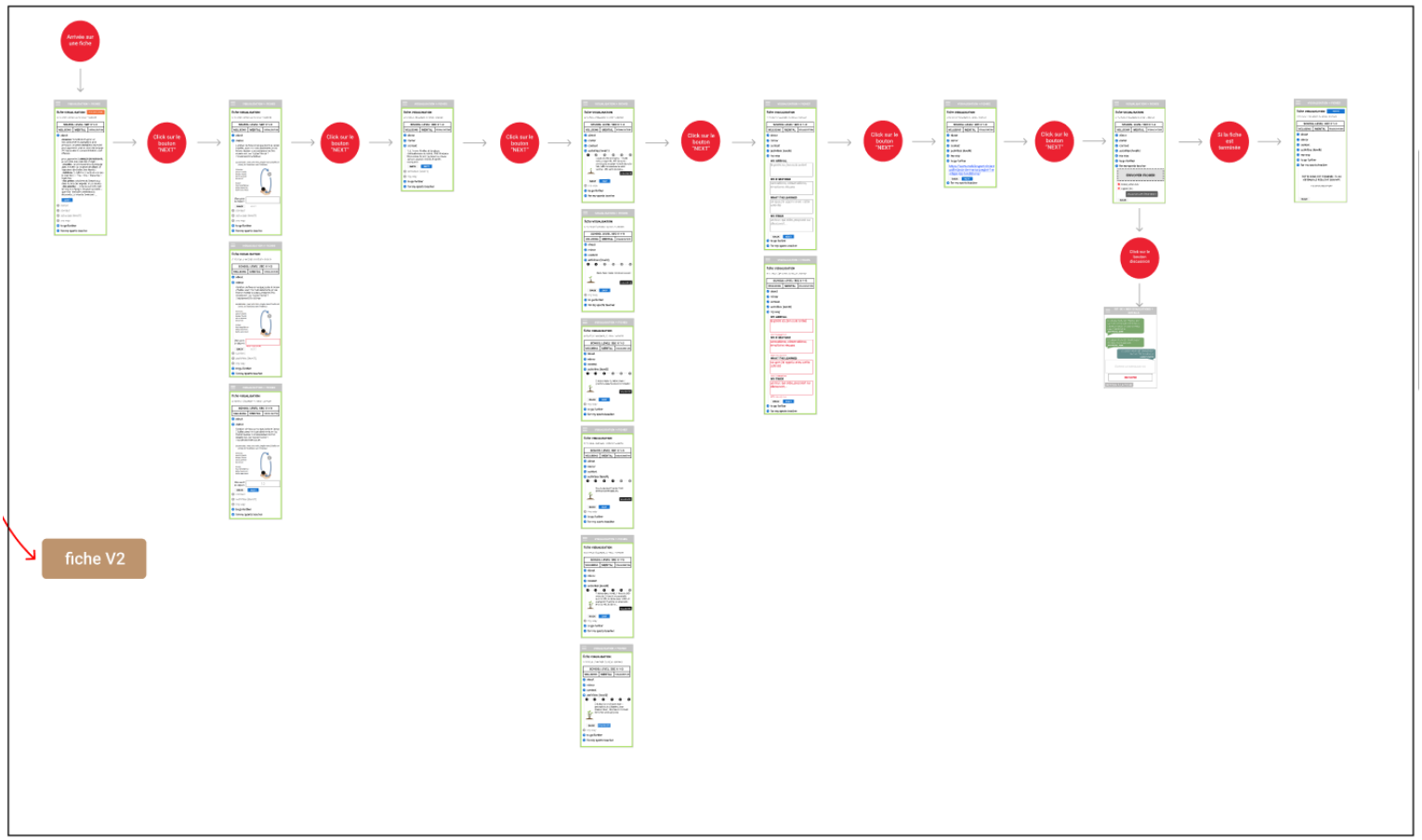

Figure 22: Flux d'une fiche

Les fiches sont accessibles par plusieurs utilisateurs, mais ce sont principalement les élèves qui auront la plus grande interaction avec elles. Les enseignants pourront en tout temps visionner les fiches afin de pouvoir assister les étudiants de la meilleure des manières.

7.4.2. About

Cette section est un simple affichage de contenu, mais c'est la section d'arrivée lorsque l'on visite une fiche. Cela permet de voir la structure de la fiche ainsi que son contenu qui varie selon les options, mais la grande majorité de la fiche reste identique. Le délai restant affiché n'apparaît que si la fiche est assignée en exercice par un professeur.

Figure 23: Fiche section about

7.4.3. Mirror

Cette partie nécessite une entrée de la part de l'élève. Elle est donc bloquante si l'élève ne donne pas son score de référence, son point de départ. L'on peut voir trois variations des mockups.

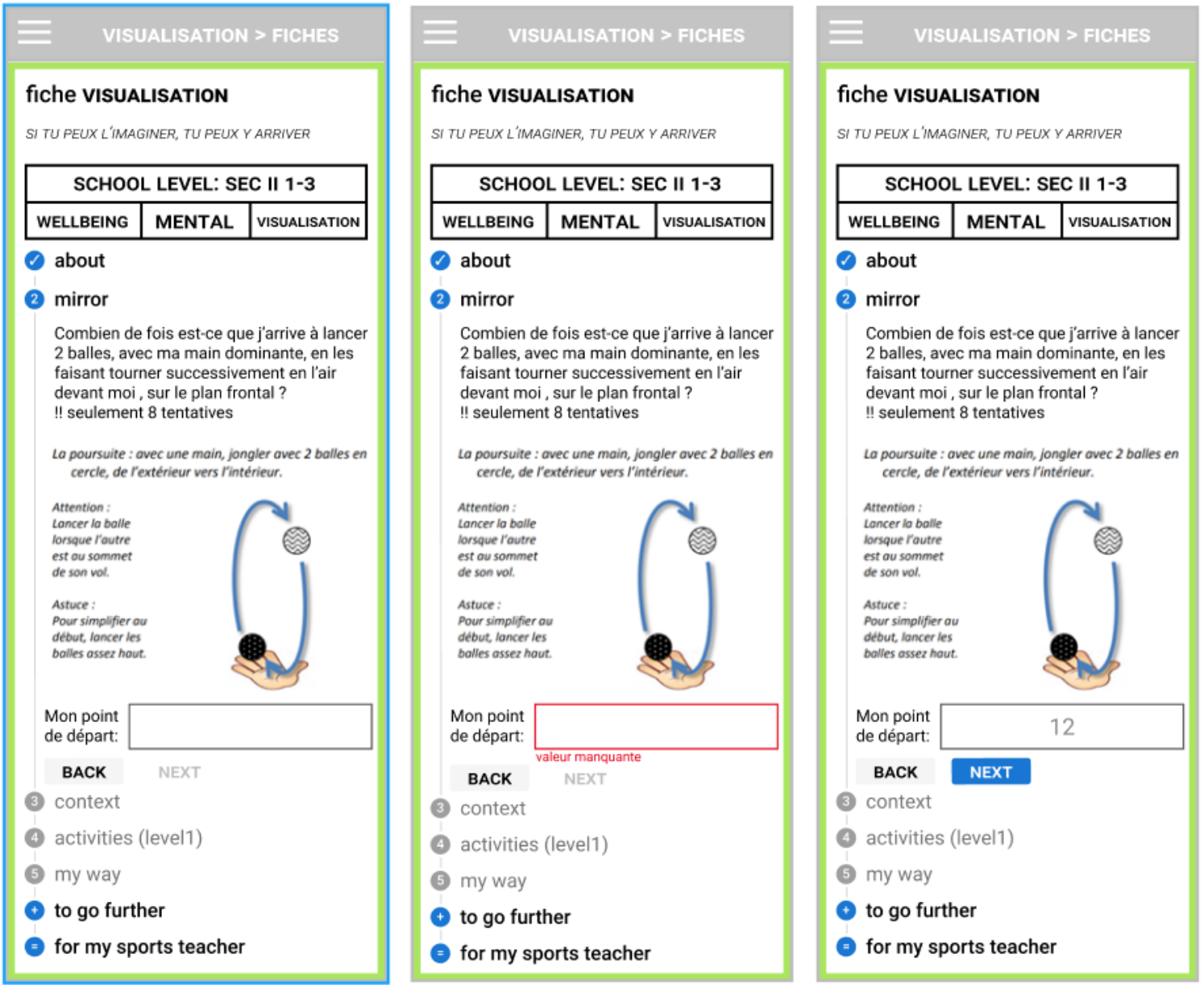

Figure 24: Fiche section mirror

7.4.4. Context

Cette section contient uniquement du contenu libre que l'étudiant doit lire.

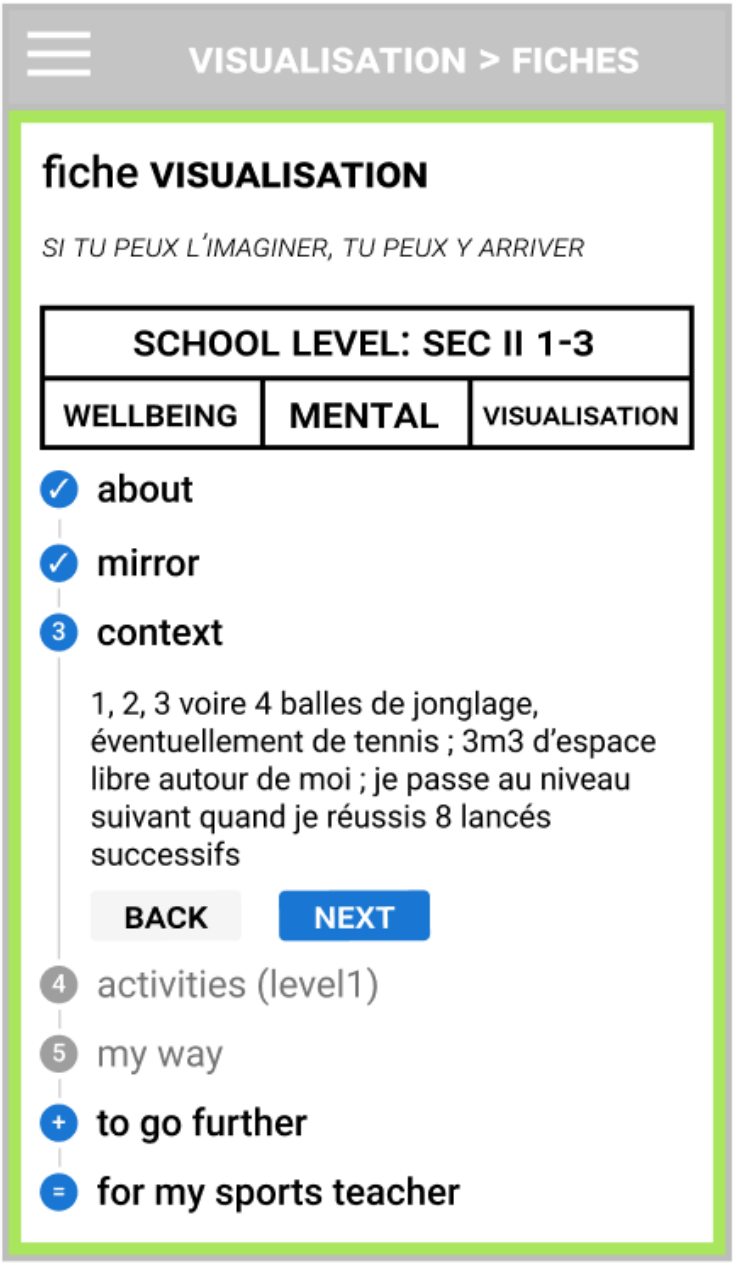

Figure 25: Fiche section context

7.4.5. Activities

La partie activités contient les différents niveaux de l'exercice que les étudiants doivent réussir à effectuer. Au fur et à mesure de leurs progrès, une image évolue; dans les mockup, c'est représenté par une plante. Il leur est possible de passer cette étape dès le moment où ils n'arrivent plus à passer le niveau suivant.

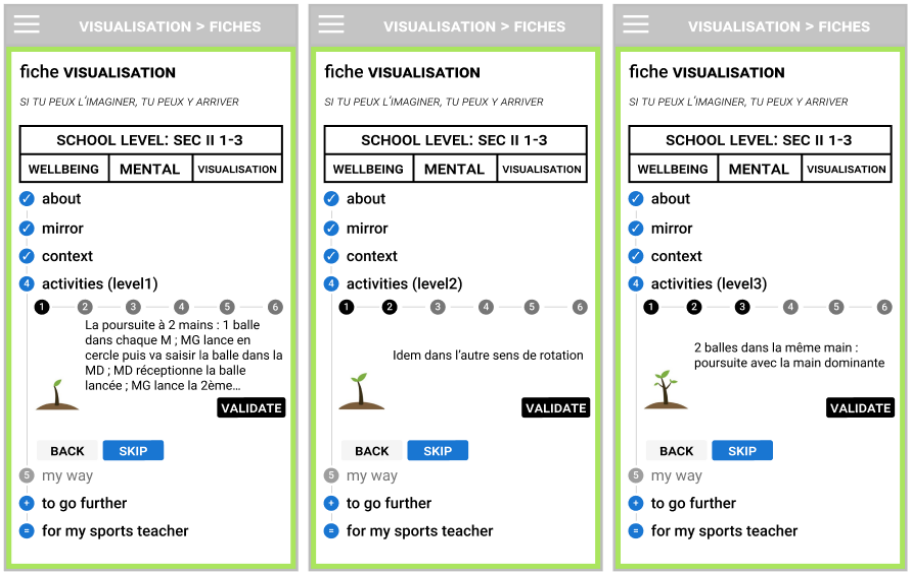

Figure 26: Fiche section activities

Grâce à des bulles d'information ou peut-être un tutoriel, cette partie du processus nécessitera sûrement plus d'explications pour les élèves utilisant l'application pour la première fois.

Lorsque l'étudiant est à la dernière étape, le bouton "next" se transforme en "complete" pour lui indiquer que tous les niveaux sont désormais complets.

Les textes des mockups ne correspondent pas à la version finale de l'application.

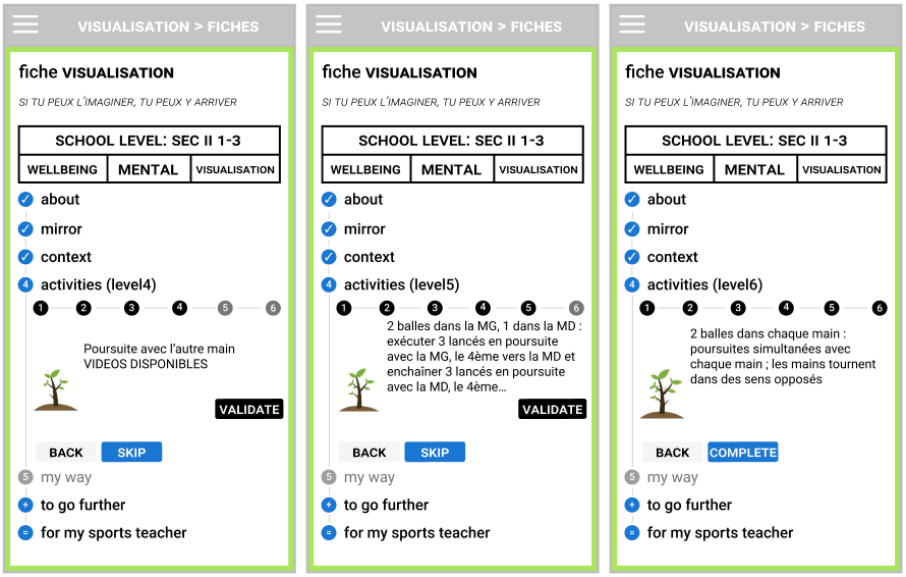

Figure 27: Fiche section activities bis

7.4.6. My way

Cette partie permet à l'élève une activité métacognitive sur son processus d'apprentissage et sa progression, ainsi que sur son ressenti pendant et au terme des activités proposées par la fiche. Il y aura aussi par la suite une section pour qu'il donne son point de vue sur la fiche elle-même, par le biais d'un questionnaire portant sur l'intérêt en situation.

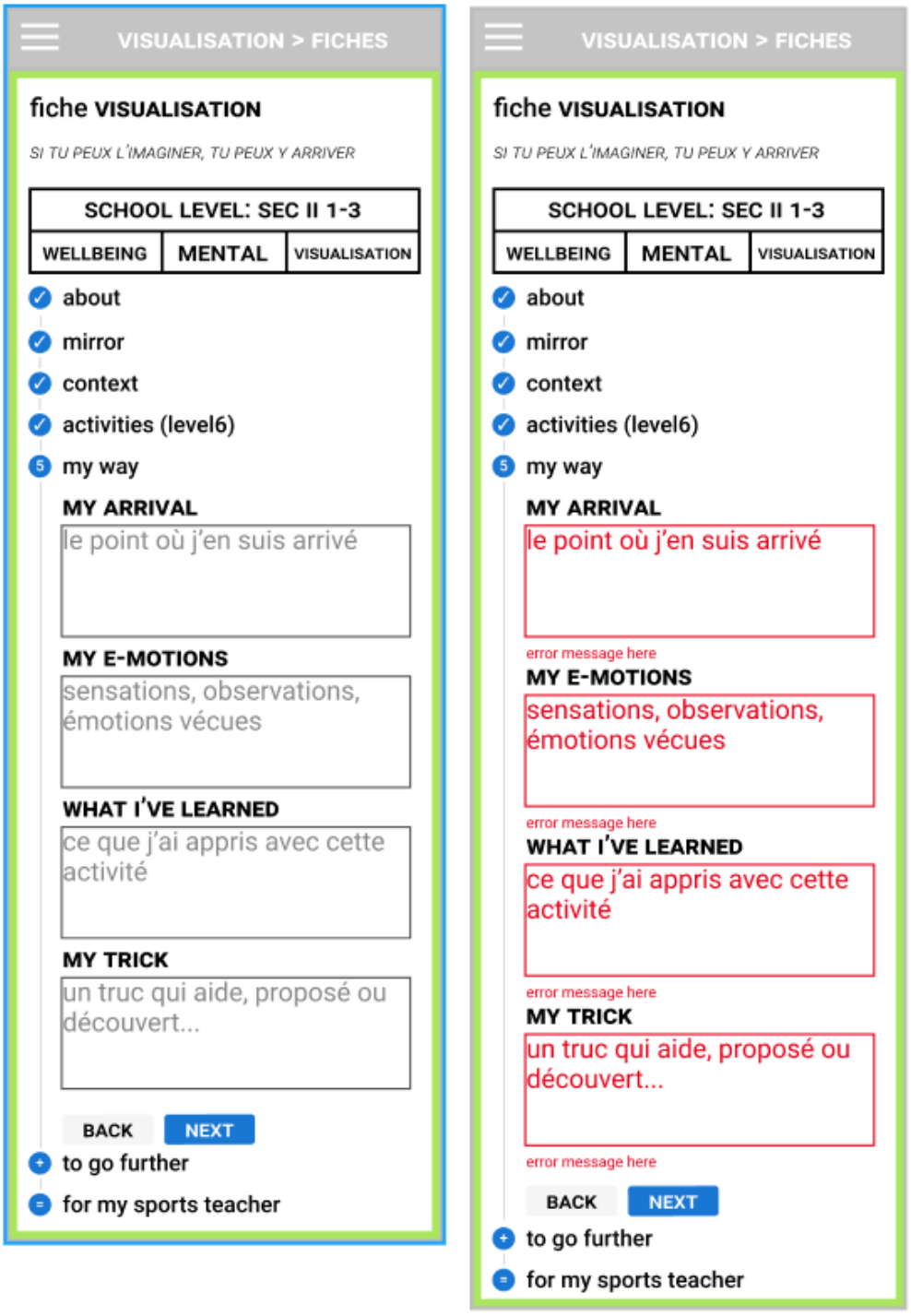

Figure 28: Fiche section my way

7.4.7. To go further

Section qui contient du contenu libre qui permet à l'élève d'approfondir la matière s'il le souhaite. Cette section est disponible en tout temps.

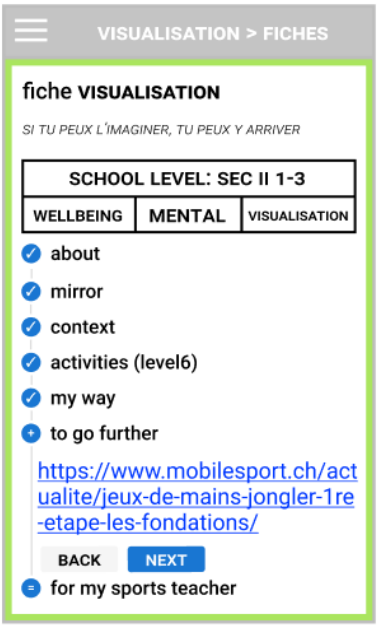

Figure 29: Fiche section to go further

7.4.8. For my sports teacher

Section qui permet à l'élève de contacter le professeur si l'option est activée et d'envoyer des fichiers contenant le résultat de l'exercice pour que l'enseignant l'évalue

Figure 30: Fiche section for my sports teacher

Une fois l'exercice terminé et évalué par le professeur, voici un exemple de contenu que l'élève pourra voir.

Figure 31: Evaluation du travail réalisé sur la fiche

7.5. Navigation des professeurs

7.5.1. Vue globale

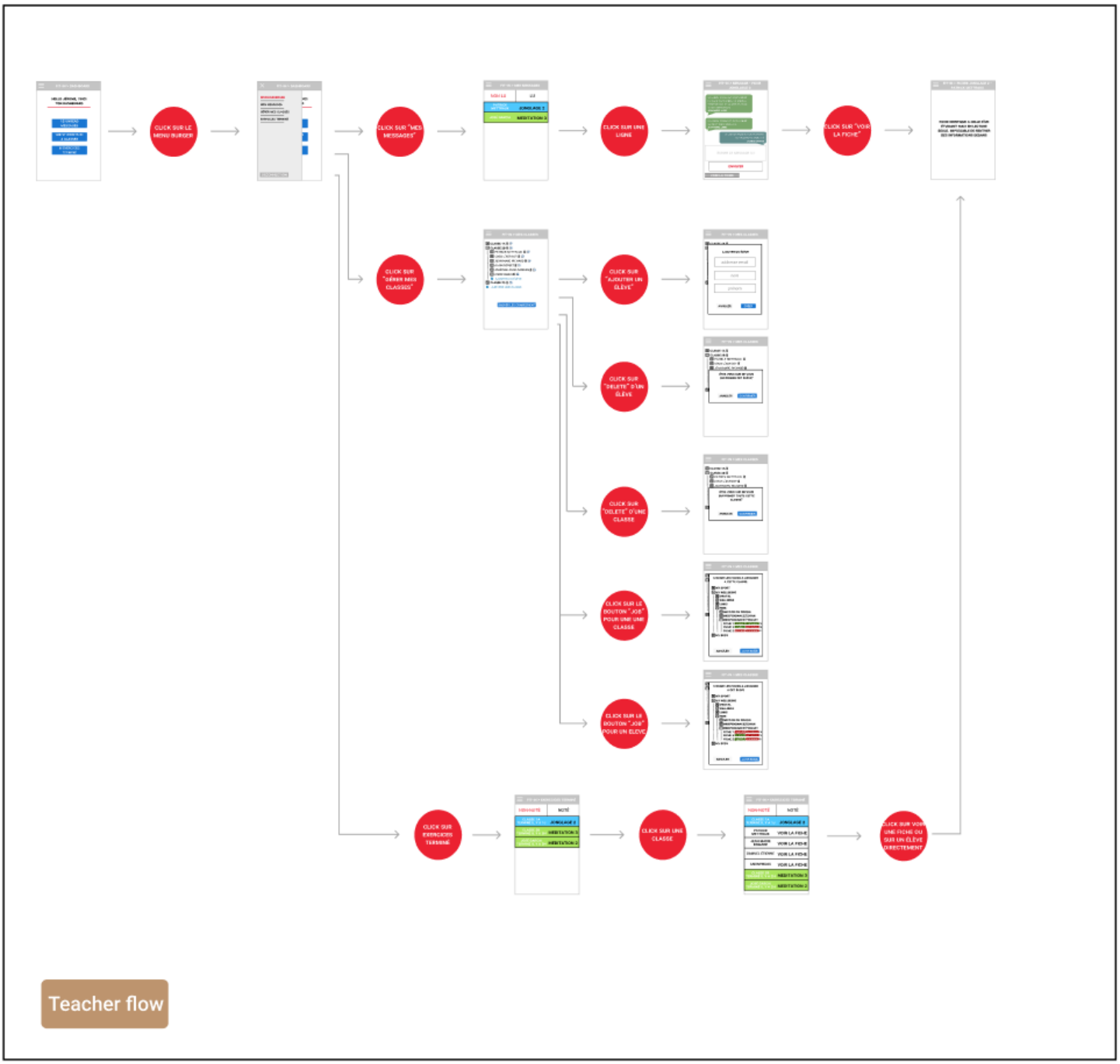

Figure 32: Flux enseignant

7.5.2. Accueil

L'écran d'accueil pour les enseignants contient quelques nombres concernant les points clés.

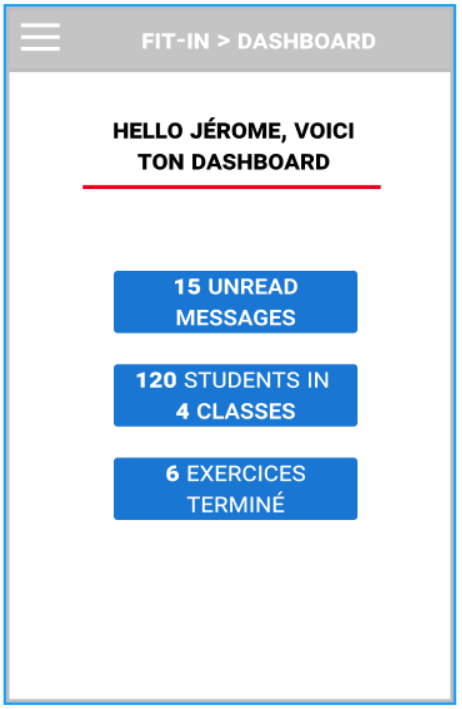

Figure 33: Dashboard enseignant

7.5.3. Menu latéral

Le menu latéral fonctionne de la même manière pour tous les utilisateurs, sa seule différence est son contenu. Ici il rappelle les points visibles sur l'écran d'accueil.

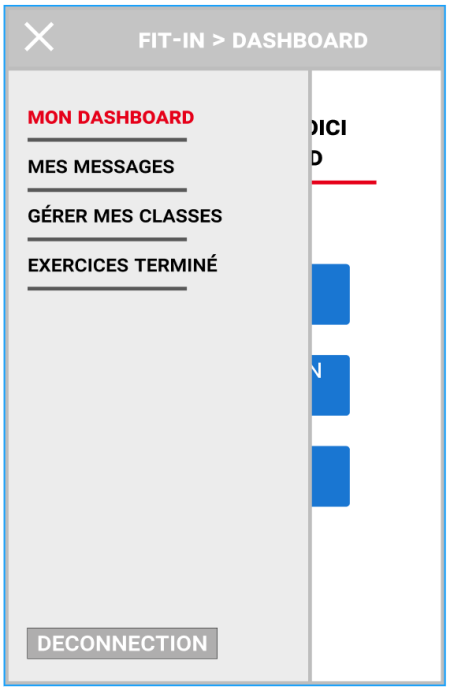

Figure 34: Menu enseignant

7.5.4. Messages

Cette partie permet au professeur de voir les nouveaux messages du/des étudiants, la fiche concernée et également les messages précédemment lus. Lorsqu'il clique sur une des lignes, il retrouve la même vue de discussion que celle de l'élève vue précédemment.

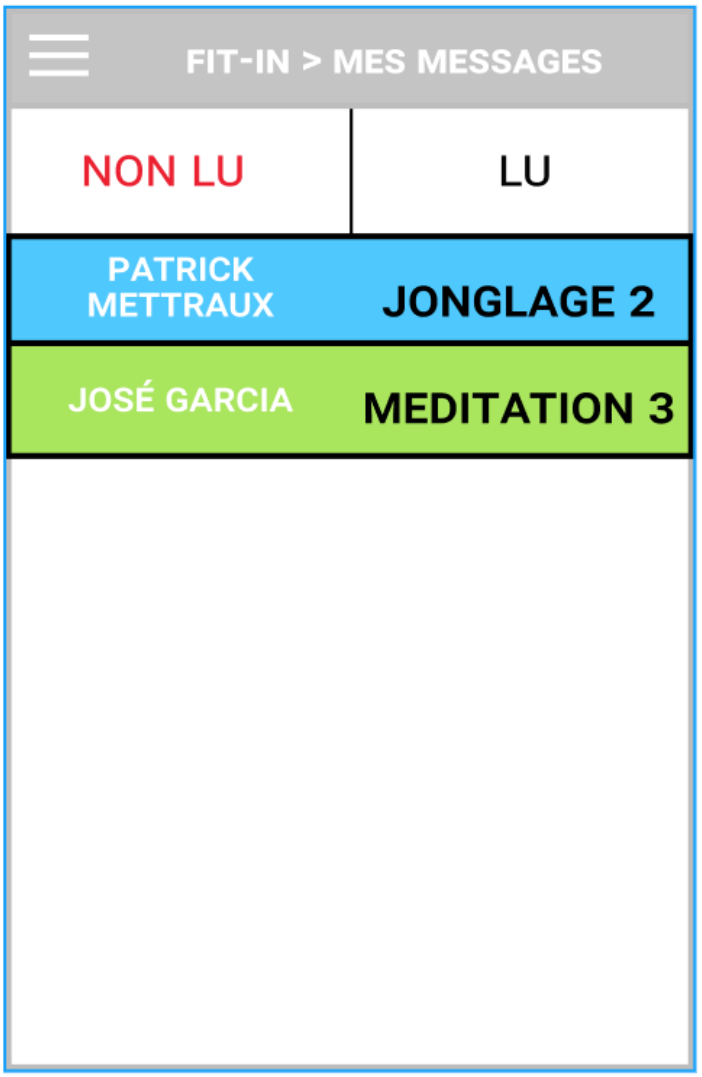

Figure 35: Gestion des messages enseignant

7.5.5. Gestion des classes

Cette partie consiste en une hiérarchie déroulante comme on le ferait pour un système de dossiers et de fichiers. Dans cet exemple, les dossiers seraient les classes et les fichiers seraient les élèves.

Le bouton “Sauver les changements” apparaît uniquement si des changements ont été effectués dans les classes et/ou les élèves. Étant donné que certaines actions peuvent engendrer l’envoi de notifications aux élèves concernés, le but est de pouvoir les regrouper avec un bouton permettant de sauver les changements quand tous ceux-ci sont terminés.

Sur cet écran, c'est ici que les enseignants vont créer leurs classes et y ajouter leurs élèves en début d’année. Il est également possible de supprimer une classe et/ou un élève.

Un bouton permet de passer à la gestion des exercices pour un étudiant ou une classe.

Figure 36: Gestion des classes

7.5.6. Ajout d'un élève

Une interface simple avec juste les infos nécessaires. Après cet ajout dans le système, l'étudiant sera capable de se connecter à Fit-In.

Figure 37: Ajout d'élève

7.5.7. Suppression d'un élève

Avant chaque suppression, une confirmation est toujours demandée.

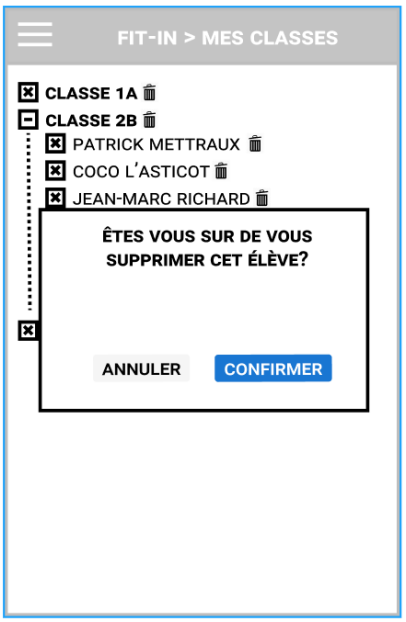

Figure 38: Suppression d'élève

7.5.8. Suppression d'une classe

Avant la suppression d'une classe, il est important de préciser que cela engendrera la suppression de tous les élèves.

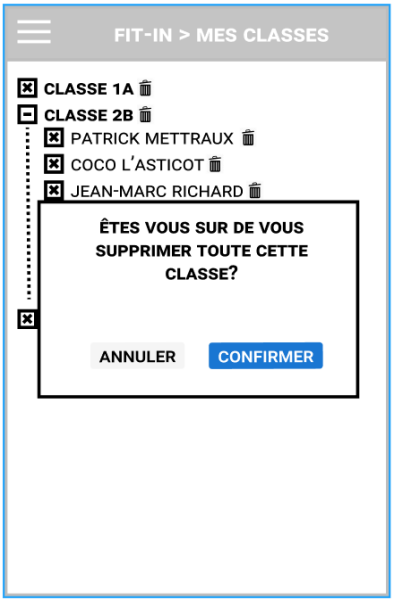

Figure 39: Suppression de classe

7.5.9. Gestion des exercices d'une classe

Cette partie permet d'assigner des fiches à toute une classe. C'est également ici que l'enseignant va activer ou non la discussion avec l'élève et affecter une date de fin pour l'exercice.

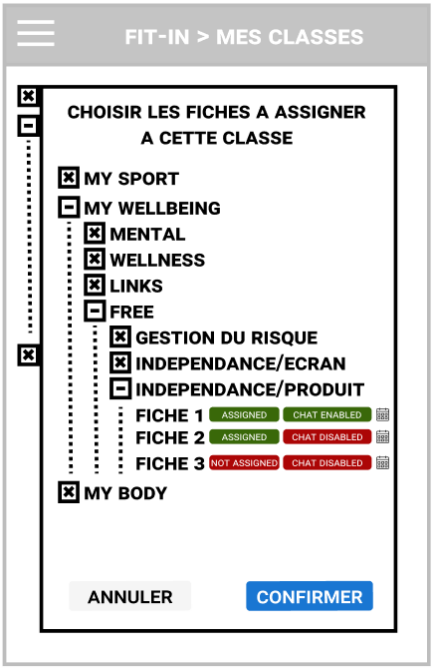

Figure 40: Assignation devoirs à une classe

7.5.10. Gestion des exercices d'un élève

Tout comme la gestion au niveau de la classe, il est possible ici de changer la valeur assignée au niveau de la classe et de la préciser au niveau de l'élève. Il est donc possible de ne pas assigner un exercice déjà assigné à la classe de l'élève ou l'inverse, il est possible d'assigner à l'élève un exercice qui n'était pas assigné à la classe. Le but ici est de permettre aux enseignants de s'adapter à une éventuelle blessure ou tout autre raison qui ferait qu'un étudiant ne puisse pas suivre le programme de la classe.

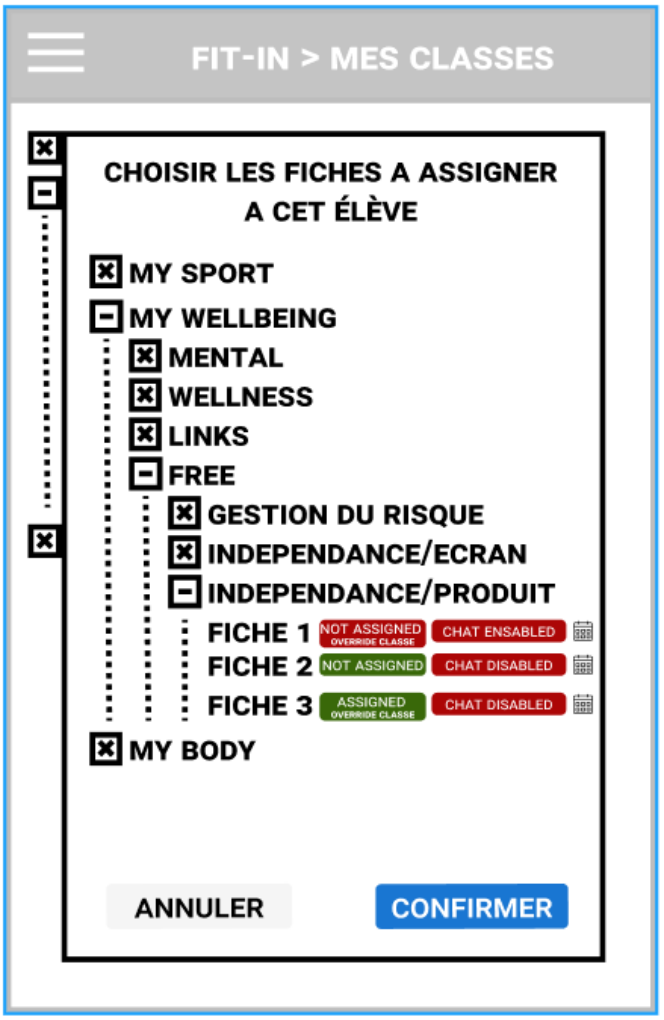

Figure 41: Assignation devoirs à un élève

7.5.11. Exercices terminés

Cette section permet à l'enseignant de voir les exercices terminés qu'il n'a pas encore notés. Il peut voir si cela concerne une classe ou un élève. Si cela concerne une classe, il peut étendre la section afin de voir chaque élève concerné.

Lorsqu'un professeur clique sur "voir la fiche" il arrive sur la même vue que l'élève a d'une fiche, sauf qu'il ne peut pas modifier ce que l'élève voit. Mais une fois la fiche terminée, il a la possibilité de la noter et l'étudiant sera informé de ceci.

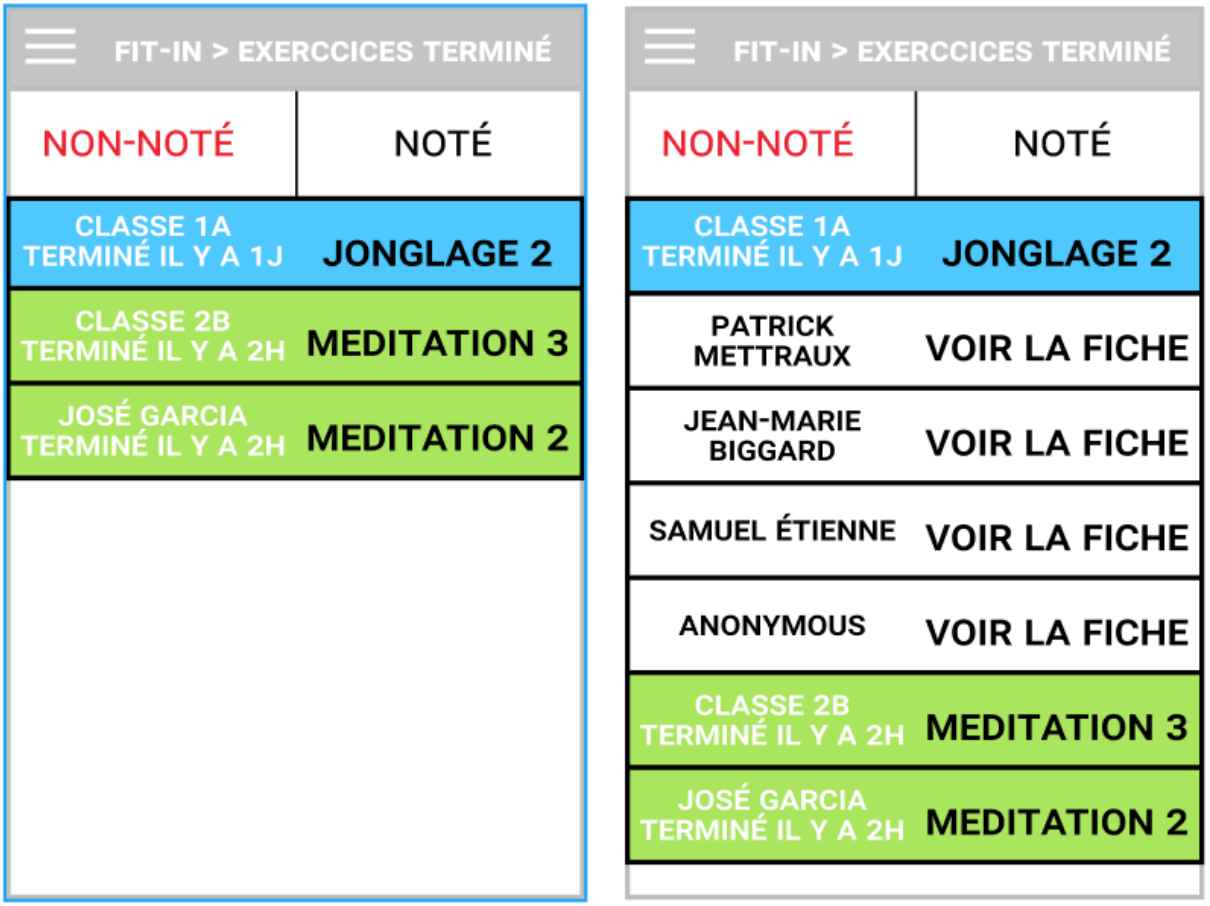

Figure 42: Exercices terminés

8. IMPLÉMENTATION

8.1. Repository de code

Deux repository ou dépôts en français, ont été créés. Un pour la partie backend et un autre pour la partie frontend. Ils ne sont pas accessibles au public et sont stockés sur mon compte personnel qui est un compte pro et me permet donc d'avoir des projets privés illimités. Ils seront donc transférables à tout moment au mandant lorsqu'il le désirera.

Le backend: https://github.com/pmettraux/fitin-backend

Le frontend: https://github.com/pmettraux/fitin-frontend

8.2. Schéma de données

Voici le schéma actuel de la base de données. Dans le futur il faudra y ajouter les tables concernant les fonctionnalités manquantes telles que le chat entre l'élève et l'enseignant, le processus des contributeurs et des validateurs.

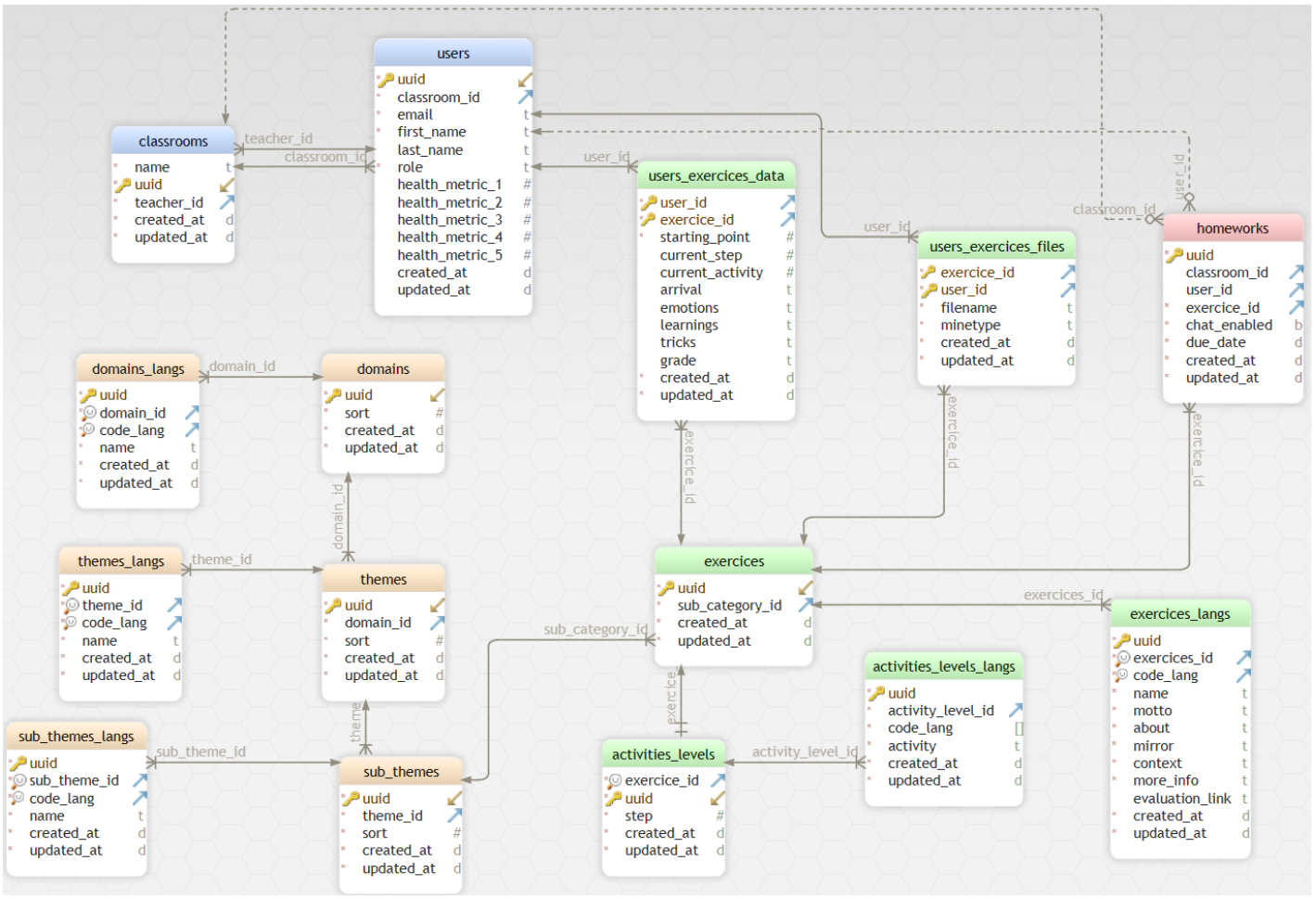

Figure 43: Schéma de la base de données

8.2.1. Identifiant Unique Universel

UUID, acronyme anglais qui signifie Universally Unique Identifier, est très souvent utilisé pour identifier des ressources dans un système de données. Le choix pour ce projet est d'utiliser des UUID plutôt qu'un simple nombre numérique incrémenté à chaque nouvel ajout dans une base de données. Même s'il est préférable de ne jamais exposer des identifiants, la réalité est que l'on en retrouve toujours soit dans des URL, soit dans un jeton ou ailleurs. Un identifiant incrémenté permet d'obtenir des informations telles que le nombre possible d'éléments dans la base de données. Ou de deviner les identifiants suivants. Un UUID supprime totalement le facteur permettant de deviner et supposer ce genre d'informations vis à vis de notre système.

8.2.2. Schéma en détail

8.2.2.1. Utilisateurs

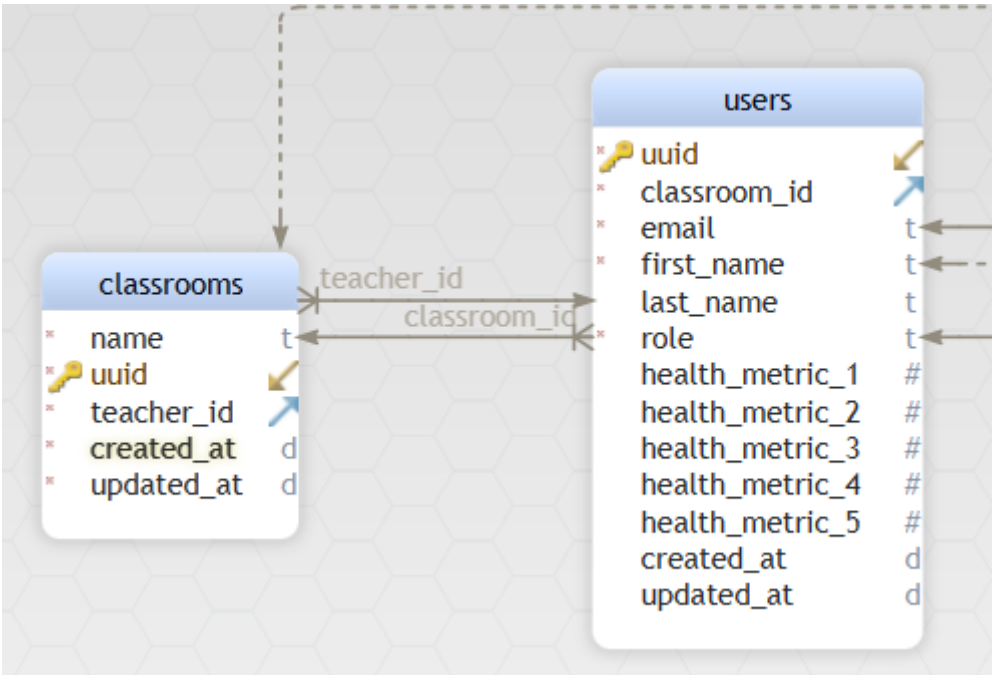

Figure 44: Schéma BDD, users

Une seule table "users" pour tous les types de "rôle". Ce qui permet de la lier directement à une classe et vice-versa. Cela permet de connaître qui est l'enseignant d'une classe et qui sont ses élèves.

Dans cette version de l'application, chaque élève ne peut appartenir qu'à une seule classe. Mais un enseignant peut avoir plusieurs classes.

8.2.2.2. Catégories

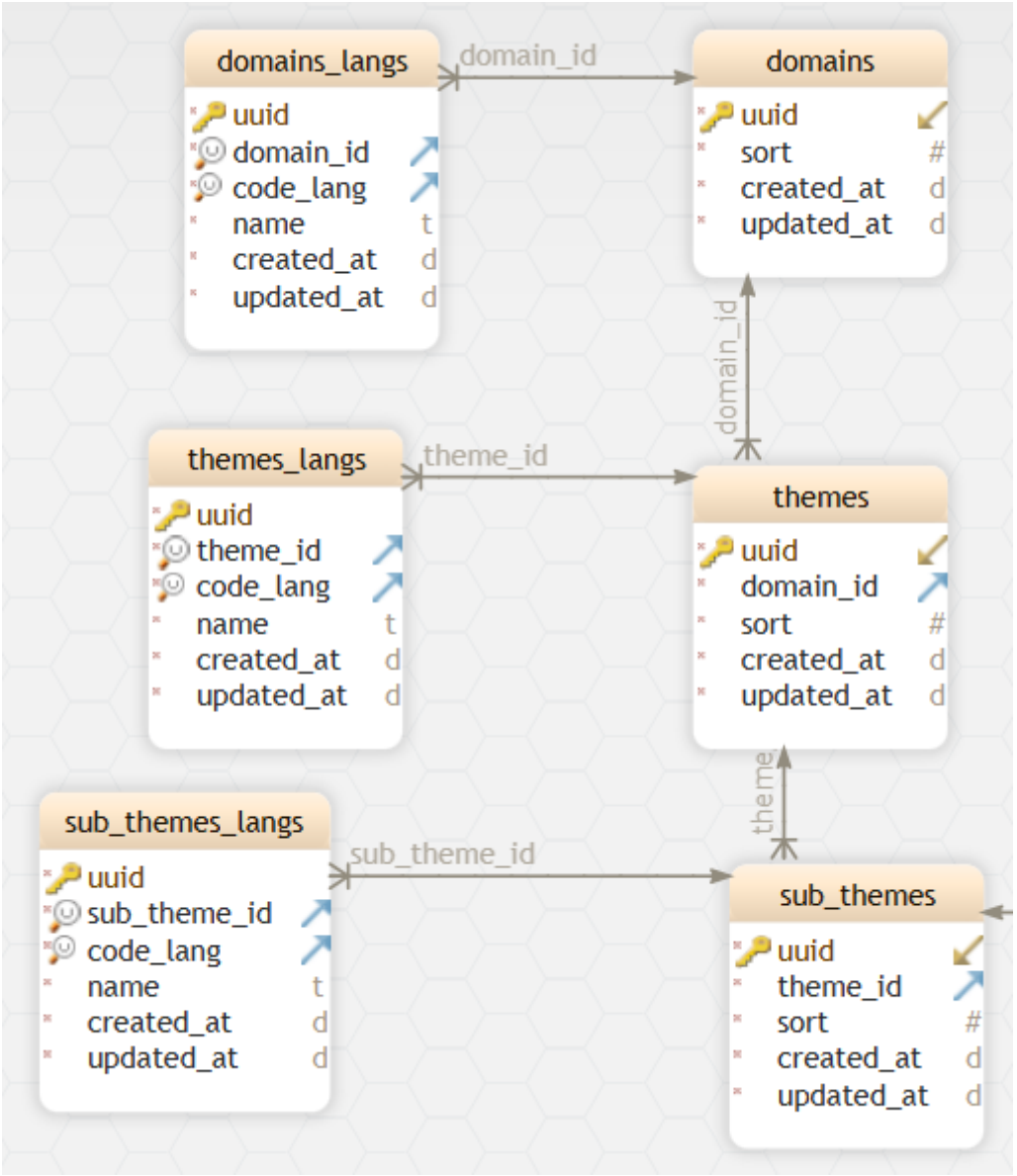

Figure 45: Schéma BDD, catégories

Ce qui ressemble ici à une duplication des tables permet de gérer le fait que ces contenus doivent pouvoir être traduits dans n'importe quelle langue. C'est pourquoi les tables "domains", "themes" et "sub_themes" gèrent la structure, permettent de définir le parent et l'ordre d'affichage. Les champs "code_lang" de chaque table sont liés à un type enum déclaré via la librairie TypeORM qui lie automatiquement le champ à un type enum créé automatiquement dans Postgres.

8.2.2.3. Exercices

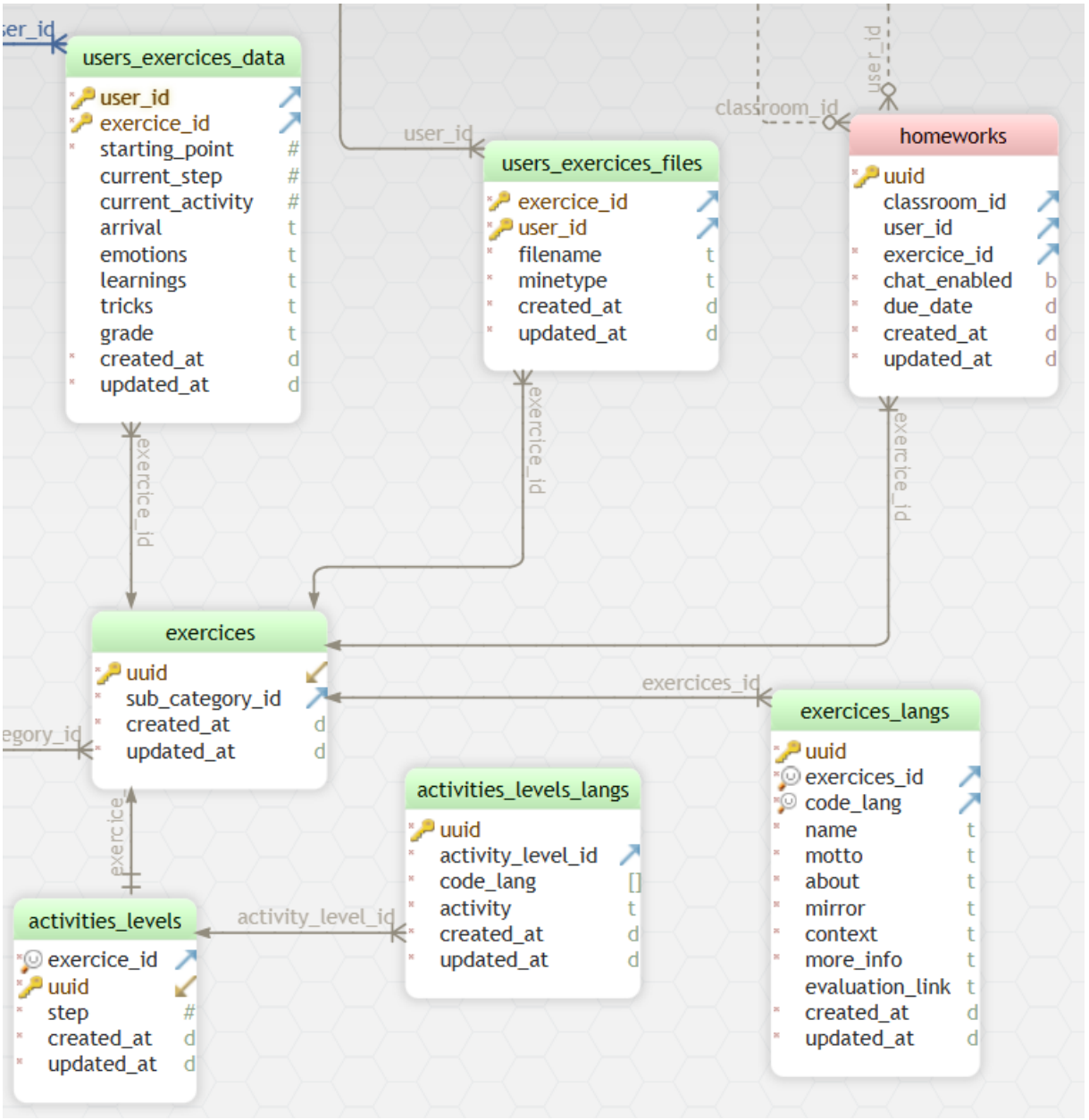

Figure 46: Schéma BDD, exercices

La partie exercice est certainement la plus complexe du système. Hormis les deux tables de langues qui permettent de traduire les textes pour un exercice et pour les activités d'un exercice, il y a les données d'un exercice lié à un élève. C'est la table "users_exercices_data" qui fait ce lien. C'est pareil pour la table "users_exercices_files", elle permet d'avoir une infinité de fichiers liés entre un exercice et un élève.

8.2.2.4. Devoirs

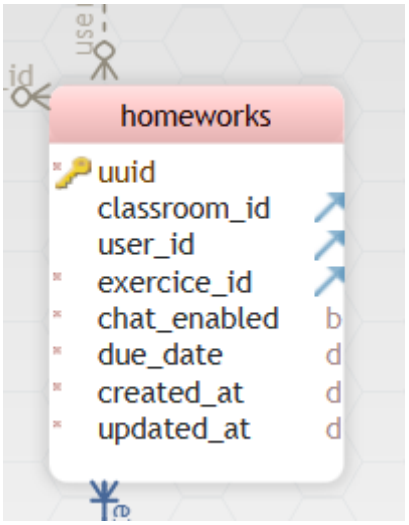

Figure 47: Schéma BDD, devoirs

Grâce à cette table “homeworks”, il a été possible de gérer l'assignation d’exercices aux élèves soit pour toute une classe via “classroom_id” soit juste pour un élève via “user_id”.

Cela a permis de mettre en place le système de priorité, par défaut la classe est prise en compte à moins qu’il y ait une donnée directe pour l’utilisateur qui surcharge la valeur de la classe.

8.2.3. Migrations

Toutes les fonctionnalités n’ont pas été développées dans ce POC. La base de données actuelle reflète l’état actuel de l’application. Des outils tels que le chat ou encore le système de soumission et d’évaluation des contributions pour les fiches devra être mis en place en termes de données également.

Le système est basé sur TypeORM qui intègre le concept de migration. Une migration permet de faire évoluer une base de données tout en gardant un historique complet de ses évolutions.

TypeORM permet de générer automatiquement ces migrations en se basant sur les modèles et donc le code. Chaque changement effectué au niveau des modèles ou chaque nouveau modèle créé permet automatiquement de générer une migration et va mettre à jour la base de données pour refléter l'état des modèles.

Une commande est mise en place et permet de gérer tout cela. Il suffit de l’utiliser comme suit dans un terminal.

```
./scripts/docker_npm_run.sh migration:generate
```

Ceci va générer le fichier qui se trouvera dans le dossier “migration”. Il suffira de le renommer afin qu’il reflète ce que la migration représente.

8.3. Environnement de développement

L'environnement de développement est, comme la mise en production, entièrement basé sur Docker. Afin de simuler certains éléments de la production en local, docker-compose est utilisé. Cela nous permet d’avoir une base de données Postgres avec exactement la même version que la production et également d’utiliser Mailhog. Mailhog est un “Email catcher” ou un attrapeur d’email en français. Il nous permet de récupérer les emails envoyés par le système sans avoir à réellement utiliser un vrai serveur SMTP .

Il suffit donc de se trouver à la racine d’un des projets, frontend ou backend et de taper `docker-compose up` pour que tout fonctionne.

Le backend a également une seconde base de données à sa disposition, dédiée aux tests “end to end”. Le but est de pouvoir lancer les tests sans affecter la base de données utilisée en local.

Ci-dessous les deux fichiers docker-compose pour le frontend et le backend.

8.3.1. Configuration Frontend

```
version: '3.2'

networks:
  fitin_net:
    driver: bridge
    ipam:
      driver: default

services:
  frontend:
    build:
      context: .
      dockerfile: Dockerfile.dev
    environment:
      NODE_ENV: develop
      BACKEND_URL: 'testbackend.com'
    ports:
      - '4200:80'
    volumes:
      - './config:/srv/config'
      - './src:/srv/src'
      - './coverage:/srv/coverage'
    networks:
      - fitin_net
```

La section "networks" avec le driver en “bridge” permet lorsque docker-compose est lancé d'accéder au frontend via l’URL http://localhost:4200

8.3.2. Configuration Backend

```
version: '3'

networks:
  fitin_net:
    driver: bridge
    ipam:
      driver: default

services:
  backend:
    build:
      context: .
    environment:
      NODE_ENV: develop
      DATABASE_URL: postgres://postgres:dev@postgres:5432/postgres
      TEST_DATABASE_URL: postgres://postgres:test@test-postgres:5432/postgres
      SMTP_PORT: 1025
      SMTP_HOST: mailhog
      SMTP_USER: mailhog
      SMTP_PASS: mailhog
      FRONTEND_URL: http://localhost:4200
    ports:
      - '3000:80'
    volumes:
      - './config:/srv/config'
      - './migration:/srv/migration'
      - './src:/srv/src'
      - './coverage:/srv/coverage'
    links:
      - mailhog
      - postgres
      - test-postgres
    networks:
      - fitin_net

  postgres:
    image: postgres:13.2
    environment:
      POSTGRES_PASSWORD: dev
    ports:
      - 15432:5432
    networks:
```

```
    - fitin_net

test-postgres:
  image: postgres:13.2
  environment:
    POSTGRES_PASSWORD: test
  ports:
    - 15433:5432
  networks:
    - fitin_net

mailhog:
  image: mailhog/mailhog
  ports:
    - 1025:1025
    - 8025:8025
  networks:
    - fitin_net
```

Dans cette configuration pour le backend, on remarque comment il est très simple d'attacher des containers docker à un projet existant et de les lier ensemble.

Tout comme le frontend grâce au network, Mailhog est accessible via l'URL: http://localhost:8025. L'interface de mailhog ressemble à l'image qui suit.

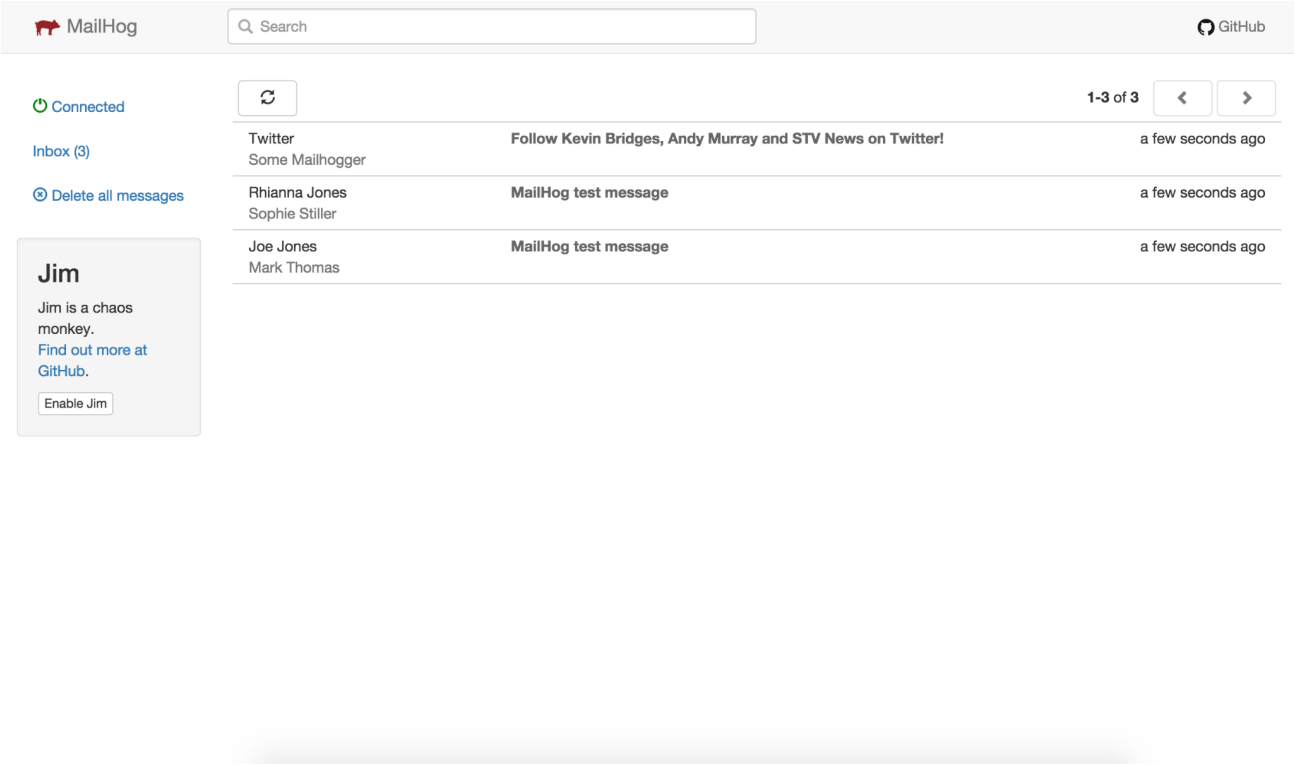

Figure 48: Mailhog

C’est donc une solution idéale pour développer des solutions utilisant l’envoi d’emails et de pouvoir tester le tout dans un environnement 100% fonctionnel malgré le fait que ce soit en local pour du développement.

8.4. Tests

Plusieurs catégories de test sont utilisées dans ce projet afin de garantir une qualité de produit et surtout d’éviter toute régression dans le code et les fonctionnalités.

8.4.1. Unitaires

Ces tests ont pour but de cibler le code directement, par exemple de vérifier qu’une fonction avec des paramètres spécifiques retournera le résultat escompté. Il est important d’avoir une couverture de code aussi grande que possible, pour les tests unitaires s’approcher du 100% si possible. Les tests unitaires dans ce projet sont utilisés dans la partie backend.

8.4.2. Bout à bout

Plus connus sous leurs terminaisons anglophones “End to end”, ces tests permettent de tester un service de A à Z. En l'occurrence ces tests sont utilisés pour le backend et permettent de vérifier que les accès à la base de données fonctionnent, que la sécurité liée aux endpoints fonctionne correctement également. L’avantage est que tous ces tests tournent via docker-compose ce qui nous permet de tester le service avec une vraie base de données et de vraies données. Ils sont un point essentiel au bon fonctionnement et à la non régression liée à la sécurité.

8.4.3. Interface Utilisateur

Ces tests sont directement liés au frontend. Ils sont réalisés via un navigateur web. En utilisant Cypress.io (cypress.io, s.d.), un outil qui permet d’écrire en typescript des tests utilisateur. Ces tests grâce à docker-compose permettent d’être exécutés sur une version totalement fonctionnelle et identique à la production de l’application angular.

Il serait donc même possible de faire tourner l’application backend et frontend et d'exécuter des tests permettant de complètement valider l’application de bout en bout. Mais au vu de la nature du projet, qui est de créer un POC, pour cette version le backend et le frontend seront testés individuellement. Rien n’empêchera l'intégration de ce genre de tests dans le futur étant donné que la structure docker est déjà entièrement faite.

Cypress.io est donc implémenté dans ce projet. L’automatisation des tests est mise en place dans le processus d’intégration continue qui sera vu au chapitre suivant. Cependant il est possible de lancer Cypress.io sur sa machine lors du développement de test afin de pouvoir tester le bon fonctionnement. Il suffit d’utiliser la commande suivante dans un terminal.

```
npm run cy:open
```

8.5. Configuration CircleCi

Pour le déploiement continu, il a été décidé d'utiliser CircleCi. Nous avons donc un fichier dans chaque projet qui contient la configuration requise.

La version gratuite de CircleCi est utilisée. Elle ne permet pas le layer caching de Docker, ce qui fait que pour chaque job le container est entièrement reconstruit à chaque fois, et il ne permet d'exécuter qu'une étape à la fois. Ce qui fait que la parallélisation des étapes n'est en réalité pas effective même si elle est déjà mise en place. Malgré ceci, les temps d'exécution sont extrêmement rapides et prouvent la qualité de cet outil.

8.5.1. Frontend

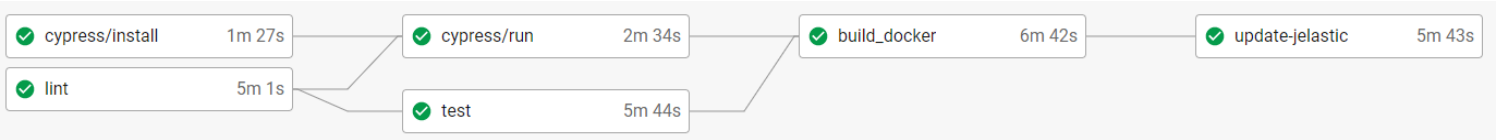

Figure 49: CircleCi Frontend

Le flux de déploiement démarre toujours par la qualité du code, avec le linter puis les tests. Si ceux-ci passent, on construit l'image docker une dernière fois et on la déploie sur https://hub.docker.com/ afin qu'elle soit disponible pour la dernière étape qui est le déploiement sur jelastic.

Pour la partie test cypress une orbe est utilisée. Une orbe est un bout de programme qui permet d'utiliser des fonctions pré écrites dans un flux CircleCi; dans ce cas-ci, l'exécution des tests cypress.

Il faut préciser que le déploiement n'est effectué que pour les commits sur la branche main. Cela permet de travailler sur sa branche sans déployer une fonctionnalité non terminée. Une fois fusionné sur la branche principale, le déploiement sera effectué.

8.5.2. Backend

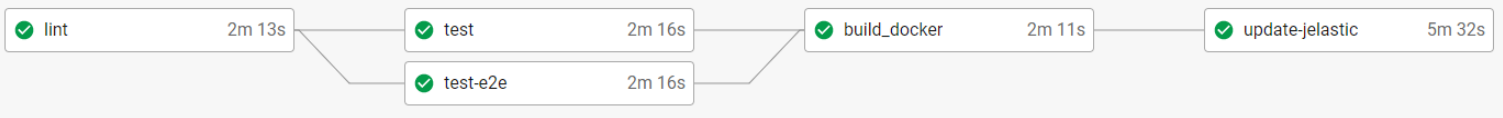

Figure 50: CircleCi Backend

Très ressemblant au frontend, le backend suit la même logique. Lint, puis tests puis l'image docker pour finir sur le déploiement.

Tout comme pour le frontend, le déploiement n'est effectué que pour les commit sur la branche principale afin de pouvoir travailler sur une autre branche le temps qu'une fonctionnalité soit prête.

8.5.3. Variables d'environnement

Comme expliqué précédemment dans la section dédiée à la configuration, aucune donnée sensible n'est disponible via le code. C'est pourquoi les informations nécessaires au script de déploiement sur Jelastic sont stockées en tant que variable d'environnement sur CircleCi.

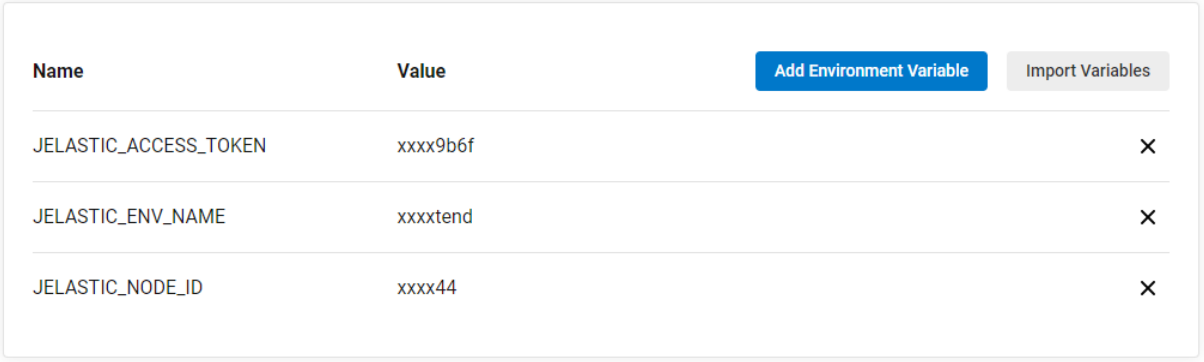

Figure 51: CircleCi variables d'environnement

Cela permet également d'avoir un seul script de déploiement sur Jelastic pour le backend et le frontend. Idéalement ce script pourrait être dans une librairie partagée. Pour le moment, il est dupliqué dans chaque repository de code.

8.5.4. Flux d'authentification

Voici sous la forme d'un diagramme de séquences, comment les utilisateurs s'identifient dans l'application.

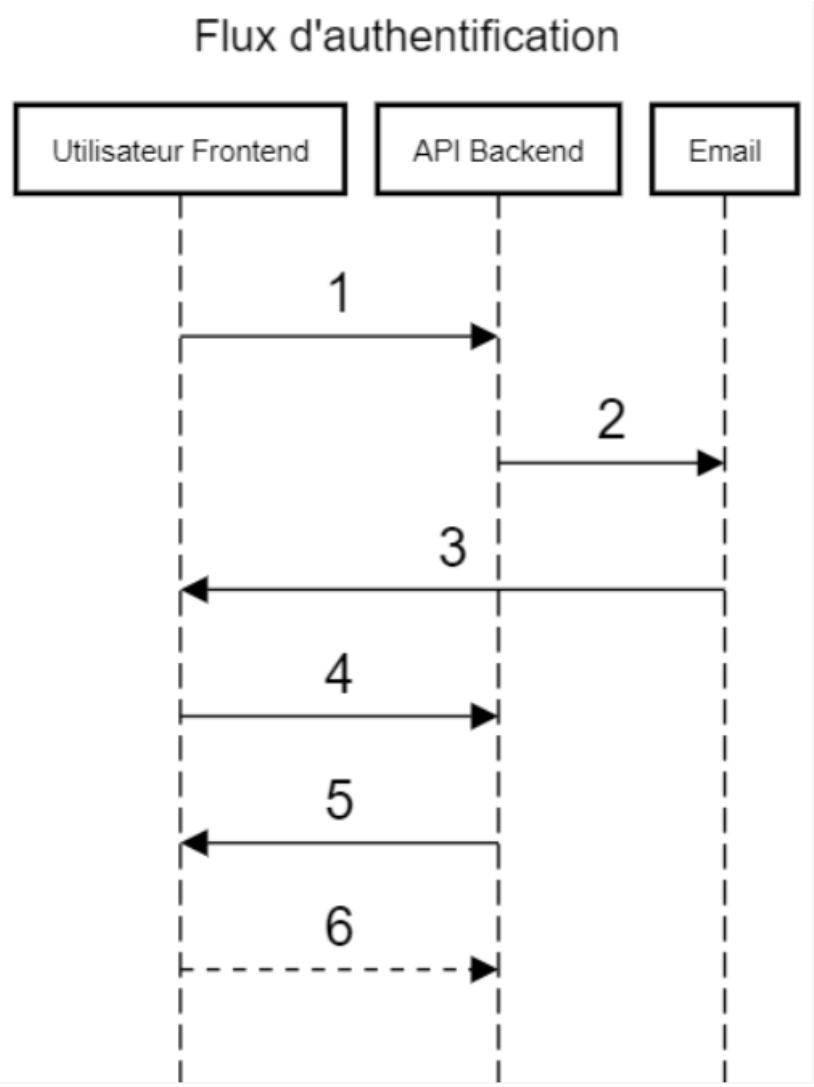

Figure 52: Flux d'authentification

1. Demande d'accès via l'envoi de l'email de l'utilisateur
2. Si l'email est valide, envoie un email avec le magic token dans un lien
3. Récupère le lien avec le magic token dedans
4. Envoie le magic token au serveur pour vérification
5. Si le magic token est valide, génère un Bearer token et l'envoie
6. L'utilisateur peut désormais utiliser le Bearer token pour toutes les requêtes aux autres endpoints de l'API, il est identifié

8.6. Sécurité

La sécurité est un point clé dans cette application. Il est essentiel de s'assurer qu'aucune donnée ne soit accessible à d'autres personnes que celles qui y sont éligibles.

8.6.1. Bearer Token

Tous les endpoints à l'exception de deux, génération et récupération du Magic Token, seront privés. Cela veut dire qu'il ne sera pas possible de les contacter dans un Bearer Token Valide. Qu'est-ce qu'un Bearer Token ? C'est un jeton d'accès en francais, qui transmet dans l'en-tête de la requête HTTP permettra à l'API backend de vérifier si l'utilisateur peut accéder à la donnée ou non.

Le contenu du Bearer Token n'est pas secret, il contiendra l'identifiant de l'utilisateur ainsi que son rôle et des méta données concernant sa durée de vie et sa génération.

Pour ce faire, JWT, Json Web Tokens (M. Jones, J. Bradley & N. Sakimura, 2015) sera employé. Un JWT est encodé en Base64, il est donc parfaitement décodable et lisible par n'importe qui, encore une fois, son contenu n'est pas secret. Mais un JWT est également signé par une clé privée et stockée au niveau du backend. Cela permet de vérifier l'authenticité du jeton d'accès. Si le jeton est modifié, la signature ne fonctionnera plus et il ne sera donc pas valide. Cette manière de faire évite l'accès à une base de données lors de chaque requête et permet très rapidement de faire un premier tri sur les utilisateurs qui peuvent ou non accéder à un endpoint du backend.

8.6.2. Éviter l'énumération

Certaines techniques pour obtenir des données consistent à chercher les endpoints d'application qui permettent de s'identifier. Plusieurs attaques existent mais dans notre cas nous n'avons ni mot de passe, ni moyen de s'enregistrer directement sur l'application. A première vue, cela semble être un système plutôt compliqué à attaquer. Il nous faut cependant nous méfier de plusieurs types d'attaques. La première est liée à ce qui est appelé les timing attacks (Selva Karthik, 2020). Ces attaques sont normalement liées à la cryptographie. Elles consistent à jouer sur le fait que chiffrer une donnée prend un certain temps. Un exemple de problème commun est le suivant. Nous avons un système de connexion avec identifiant et mot de passe.

Lorsqu'un utilisateur entre des données, le serveur va en premier lieu rechercher l'utilisateur dans la base de données. Très souvent si l'utilisateur n'est pas trouvé, le serveur va directement retourner une erreur. Le serveur va comparer les mots de passe uniquement si l'utilisateur se trouve dans la base de données. Cela peut paraître anodin mais c'est en réalité une erreur, il est en effet possible en envoyant plein de requêtes de comparer les temps de réponse du serveur pour en déduire les utilisateurs existants ou non dans la base de données.

Dans notre cas nous n'avons pas de cryptographie. Cependant nous avons l'envoi d'un email. C'est pourquoi l'endpoint s'occupant d'envoyer l'e-mail n'attend pas la réponse du serveur pour

fournir une réponse à l'utilisateur, cela ne permet donc pas l'énumération de comptes existants sur notre système. D'autant plus que nos comptes sont en réalité des adresses email et contiennent très souvent des noms et prénoms de personnes réelles. Il est important de les protéger.

8.6.3. Eviter la force brute

La force brute est l'attaque la plus basique possible. C'est celle qui consiste à essayer différentes combinaisons et ce jusqu'à trouver la bonne. Il est donc possible en théorie de lancer une attaque sur l'endpoint qui s'occupe de la validation du jeton magique afin d'obtenir le jeton d'identification utilisé ensuite pour chaque appel au serveur.

Un rate-limiter a été mis en place. Cela consiste en un système qui va limiter le nombre de requêtes en se basant sur l'adresse IP.

Le système va donc bloquer une adresse IP pendant un certain laps de temps si celle-ci a engendré un certain nombre d'erreurs. Cela permet de rendre une attaque par force brute totalement obsolète, car s'il n'est pas possible de lancer des milliers de requêtes par secondes mais uniquement trois requêtes toutes les dix minutes; il faudrait une éternité pour passer ce système.

8.6.3.1. Failles potentielles

Le système est loin d'être parfait et contient encore des failles potentielles. La principale est une attaque via l'entête de requête HTTP "x-forwarded-for". Cet en-tête permet de faire suivre une adresse IP, très utilisé lorsqu'un réseau passe par plusieurs proxy ou des load balancer (gestionnaire de charge en français).

Le système actuel accepte les adresses venant du "x-forwarded-for", il est donc possible de changer son adresse IP et de rendre obsolète le rate-limiter étant donné que ce dernier se base sur l'adresse IP. Il faut donc limiter les adresses IP du x-forwarded-for à celles de notre infrastructure et interdire toutes les autres afin de sécuriser ce type d'attaque.

Il faut également limiter le temps de validité du jeton de connexion. Pour ce POC il est spécialement élevé. Dans l'idéal, je conseillerais 24h de validité. Si l'on implémente un système de jeton de rafraîchissement (refresh token en anglais), cela serait suffisant pour sécuriser ce genre d'application.

Un point qui pourrait être sensible, c'est le fait qu'un jeton magique n'est jamais invalidé. Ce qui veut dire qu'une fois utilisé, il pourrait être réutilisé plusieurs fois. Cependant, l'intérêt des jetons signés est de n'avoir aucune donnée en base de données avant d'éviter des requêtes coûteuses et non nécessaires. Pour invalider un jeton, il faut soit garder une table qui liste les jetons invalidés, mais cela revient à ajouter une complexité coûteuse à notre base de données postgres. Il faudrait également mettre en place un nettoyage de cette table, ce qui ne semble clairement pas la meilleure idée. La solution que je suggère est de mettre en place un serveur Redis

et de stocker les jetons invalides dedans avec un TTL (Time to live, temps de vie en français) de la durée de vie du jeton magique. L'avantage de Redis est que c'est une base de données entièrement gérée en mémoire, extrêmement rapide (redis.io, s.d.) et certainement la mieux adaptée à cette solution. J'ai dans d'autres projets déjà utilisé Redis pour des solutions très similaires à celle-ci. De plus, la solution de TTL fait partie intégrante de Redis, lorsque l'on sauve une clé dans Redis il faut spécifier une durée de vie. Donc Redis se nettoie automatiquement.

Plus concrètement voici les deux commandes Redis qui permettent de solutionner notre problème. La première, le stockage du jeton invalidé dans Redis:

```
SET TOKEN_VALUE 1 EX 86400
```

Cette ligne va créer une entrée dans Redis avec la valeur TOKEN_VALUE qu'il faudra remplacer par la valeur du jeton. La durée de vie de cette entrée est d'une journée soit 86400 minutes.

Maintenant il nous faut une commande pour savoir si le jeton que nous allons utiliser a été invalidé les dernières 24 heures. Pour ceci, il suffit d'utiliser la ligne suivante dans Redis:

```
MGET TOKEN_VALUE
```

Voici donc comment grâce à Redis il est possible de solutionner le problème de validité des jetons magiques.

8.6.4. Sécuriser un endpoint

D'un point de vue du code, tout est en place pour sécuriser les endpoints. La sécurité est donc basée sur le rôle d'un utilisateur et en utilisant un simple décorateur, tout sera sécurisé. Voici un exemple d'endpoint public:

```
@Get('/public')
@Public()
async publicEndpoint(): Promise<void> {
  return;
}
```

Et voici un exemple d'endpoint sécurisé et limité aux enseignants et aux élèves:

```
@Get('/private')
@Roles(ERoles.Student, ERoles.Teacher)
async privateEndpoint(): Promise<void> {
  return;
}
```

8.6.5. Validation des données

Afin de protéger les endpoints il est essentiel de pouvoir valider les données qui sont autorisées à être transmises. Pour cela NestJS a un système de validation intégré (nestjs.com, s.d., validation) qui fonctionne autour de la librairie class-validator[9] et class-transformer[10].

L'idée est de définir des DTO (Data Transfer Object), Objet de transfert de données en français. Cela permet d'avoir un typage strict au niveau du code et de permettre une validation qui donne une excellente indication à l'utilisateur.

Par exemple pour notre endpoint qui envoie un email à l'utilisateur avec son lien de connexion, le DTO est très simple, il se définit comme suit:

```
export class MagicTokenCreateInputDTO {
  @IsEmail()
  email: string;
}
```

Le DTO doit être une classe, et chaque propriété de la classe représente un paramètre qui peut être transmis. Par défaut les paramètres sont obligatoires, il est possible de les rendre optionnels. Si ce paramètre est manquant, l'endpoint va nous retourner l'erreur suivante:

```
{
  "statusCode": 400,
  "error": "Bad Request",
  "message": ["email must be an email"]
}
```

Le code HTTP est 400, qui représente une erreur de validation et l'erreur explique que l'email est un champ requis et qu'il doit être d'un format de type email. Il est possible de pousser la

[9] https://github.com/typestack/class-validator#readme

[10] https://github.com/typestack/class-transformer#readme

validation très loin, ce système intégré à NestJs nous permet donc d'avoir une bonne structure et de contrôler l'information qui transite dans notre système.

8.7. Multilingue

La gestion des langues est centralisée autour de www.poeditor.com. Cet outil permet le stockage de clés et permet aussi au besoin d'exporter les traductions sous différents formats. Cela permet d'avoir un endroit centralisé pour traduire nos textes et également les soumettre à des traducteurs au besoin. Poeditor gère plein de formats en entrée, mais pour nous uniquement le format GetText (gnu.org, s.d.) qui emploie des fichier.pot comme fichier de référence sera utilisé. En sortie, cela dépendra entre le frontend et le backend. Le processus se déroule en trois étapes:

- Extraction des clés sur Poeditor
- Traduction des clés via le site Poedit
- Importation des traductions sur le backend ou le frontend au format voulu

8.7.1. Processus

Voici un aperçu du processus de traduction via poeditor. Ci-après les étapes importantes sont reprises dans le détail.

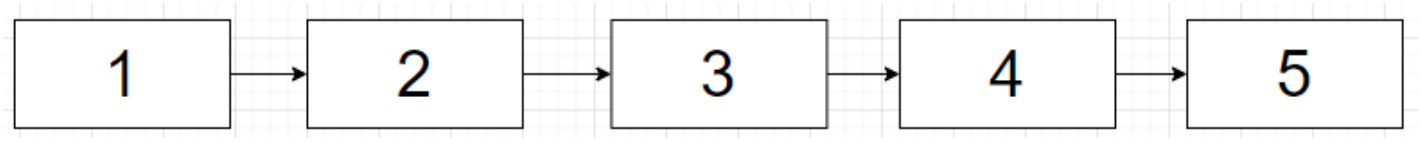

Figure 53: Flux de traduction

1. Extraction des clés
2. Envoi du fichier .pot à poeditor.com
3. Traduction des clés sur le site poeditor.com
4. Excecution du script pour récupérer les traductions via poeditor.com
5. Commit du code sur github

8.7.2. Extraction

La première étape est l'extraction des clés à traduire. Par exemple le frontend utilise la librairie ngx-translate-extract[11] pour l'extraction. Cet outil va parcourir le code source pour trouver les clés à traduire qui sont indiquées de différentes manières, et il va en générer un fichier .pot comme l'exemple ci-dessous.

[11] https://github.com/aamirvohra/ngx-translate-extract

```
msgid ""
msgstr ""
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=utf-8\n"
"Content-Transfer-Encoding: 8bit\n"

msgid "Email is required"
msgstr ""

msgid "Submit"
msgstr ""

msgid "Success! Please check your email"
msgstr ""
```

8.7.3. Traduction

Ce fichier est ensuite envoyé sur Poeditor et désormais il est possible de le traduire. Voici à quoi ressemble l'interface une fois traduit.

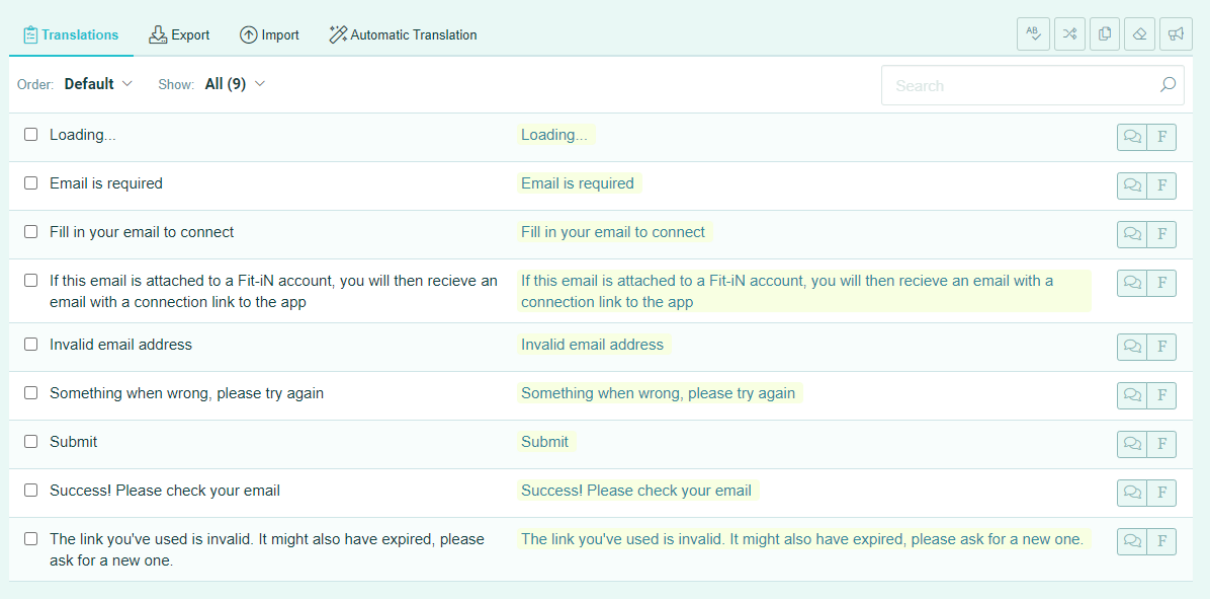

Figure 54: Traduction

Poeditor permet aussi de payer pour avoir des traducteurs professionnels (poeditor.com, s.d., How to get human translation services) qui s'occupent des traductions. Sa version gratuite (poeditor.com, s.d., Pricing plans that scale with your business) est celle utilisée; elle permet le stockage de 1000 chaînes de caractères. Cela est largement suffisant pour ce projet.

8.7.4. Importation

Une fois nos textes traduits il faut les importer dans notre projet. Poeditor possède une API (poeditor.com, s.d., API v2) qui va permettre d'automatiser ce processus. Le script suivant est celui utilisé pour le frontend. Il récupère toutes les langues traduites et les sauvegarde au format .json qui est celui utilisé dans notre cas.

```
 #!/usr/bin/env bash

if [ -z "$1" ]; then
    echo """Invalid usage, API token must be supplied. API token can be found on
poeditor.com under Account -> API access - https://poeditor.com/account/api

    Usage: $0 <API_TOKEN>
"""
exit 1
fi

TOKEN=$1
API=https://api.poeditor.com/v2

# project ID of this frontend
PROJECT=442383

LANGUAGES=$(curl -s -X POST $API/languages/list -d api_token=$TOKEN -d
id=$PROJECT | jq ".result.languages[].code" -r)
echo $LANGUAGES
for lang in $LANGUAGES; do
  echo $lang
  output_file=./src/locale/${lang}.json
  po_file=$(curl -s -X POST $API/projects/export -d api_token=$TOKEN -d
id=$PROJECT -d type=key_value_json -d language=$lang | jq -r .result.url)
  echo $po_file
  echo $output_file
  wget -q -O "${output_file}" $po_file
  echo "Exported $lang as ${output_file}"
done
```

8.8. Infrastructure

Comme évoqué lors du chapitre parlant du choix de l'hébergement, le choix s'est porté sur Infomaniak. La solution Jelastic aurait été la même dans n'importe quel cas.

Cette solution est mise en place pour la partie appelée "staging". C'est l'environnement pré-production. Dans notre cas nous n'avons pas d'environnement de production, le seul environnement utilisé est celui de "staging" et il a servi à avoir une version disponible en ligne de

l'application pour faire le point avec le mandant à la fin de chaque quinzaine. C'est également l'environnement qui sera utilisé pour la démo lors de la défense de ce projet.

L'infrastructure est séparée en deux parties. La partie frontend et la partie backend. Elles sont indépendantes l'une de l'autre.

Avec Jelastic plusieurs options sont disponibles. Dans le cas de ce projet, le but est de se baser entièrement sur les containers docker automatiquement créés via le déploiement continu et le processus CircleCi. Pour ce genre de configuration, Jelastic oblige l'utilisation d'un load-balancer (gestionnaire de trafic en Francais) pour chaque environnement. Il est possible d'utiliser une image docker ou de choisir parmi des load-balancer existant dont NGINX[12], c'est le choix que j'ai fait. L'avantage est qu'il n'était pas nécessaire d'avoir de configuration complémentaire pour le faire fonctionner. Jelastic s'occupe de tout.

[12] https://www.nginx.com/

8.8.1. Backend

Hormis le load-balancer qui est obligatoire, cet environnement contient un container docker ainsi qu'une base de données PostgreSQL. La base de données est native, elle ne tourne pas dans un container, même si cela aurait été possible.

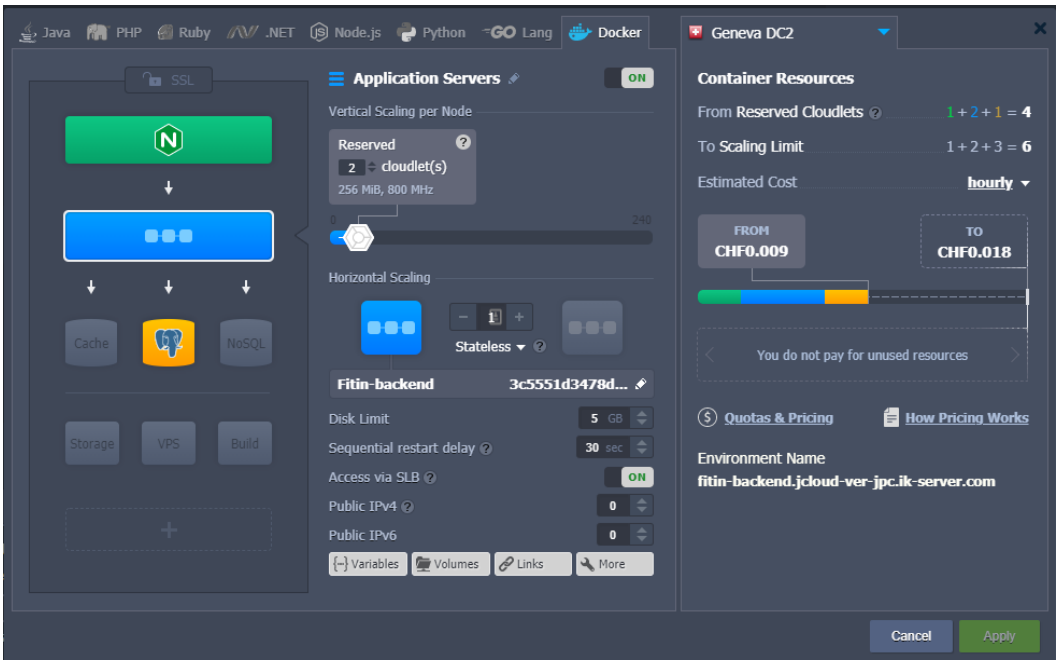

Figure 55: Jelastic Backend

En termes de nombre de nodes, tout est à une node, sauf la partie container de notre code qui nécessitait un peu plus de mémoire pour tourner de manière fluide. Avec un seul container, le processeur était utilisé au-delà des 80% en permanence. Avec deux nodes il se trouve désormais autour des 30%.

Il est possible de définir une fourchette minimale et maximale du nombre de nodes autorisés et de configurer Jelastic pour, qu'en cas de charges, de nouvelles nodes soient automatiquement créées pour supporter la charge. Dans notre cas, étant donné que cet environnement n'est pas celui qui sera utilisé pour la production, aucune fourchette n'est spécifiée. Le but est aussi de réduire les coûts au maximum.

Toute cette configuration se fait via l'interface graphique via un navigateur web. L'interface est très bien détaillée, il est possible de pousser la configuration très loin.

Le schéma de notre infrastructure backend se dessine donc comme illustré dans l'image suivante:

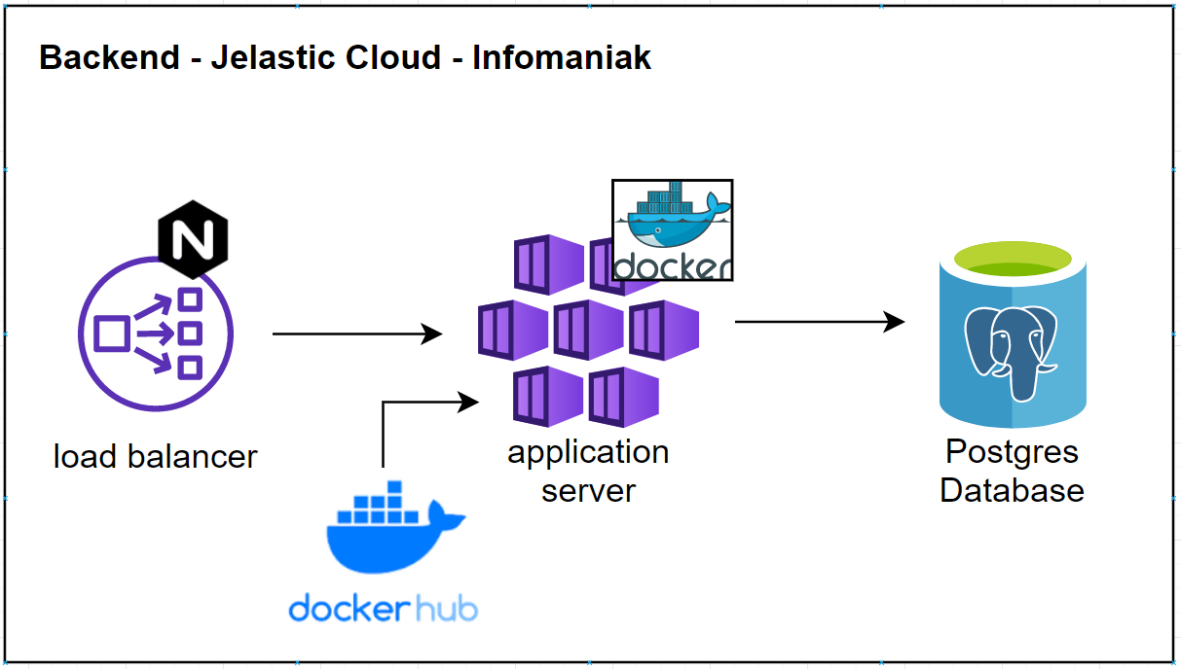

Figure 56: Infrastructure Backend

8.8.2. Frontend

Tout comme pour la partie backend, hormis le load balancer qui est obligatoire, cet environnement ne contient que notre container docker avec le code pour l'application Angular frontend.

Toutes les configurations sont au minimum. L'avantage est que l'application Angular ne fait que de servir des fichiers statiques qui sont générés lors du processus de déploiement continu via CircleCi. Cela veut dire qu'en termes de CPU aucune utilisation excessive ne sera jamais faite.

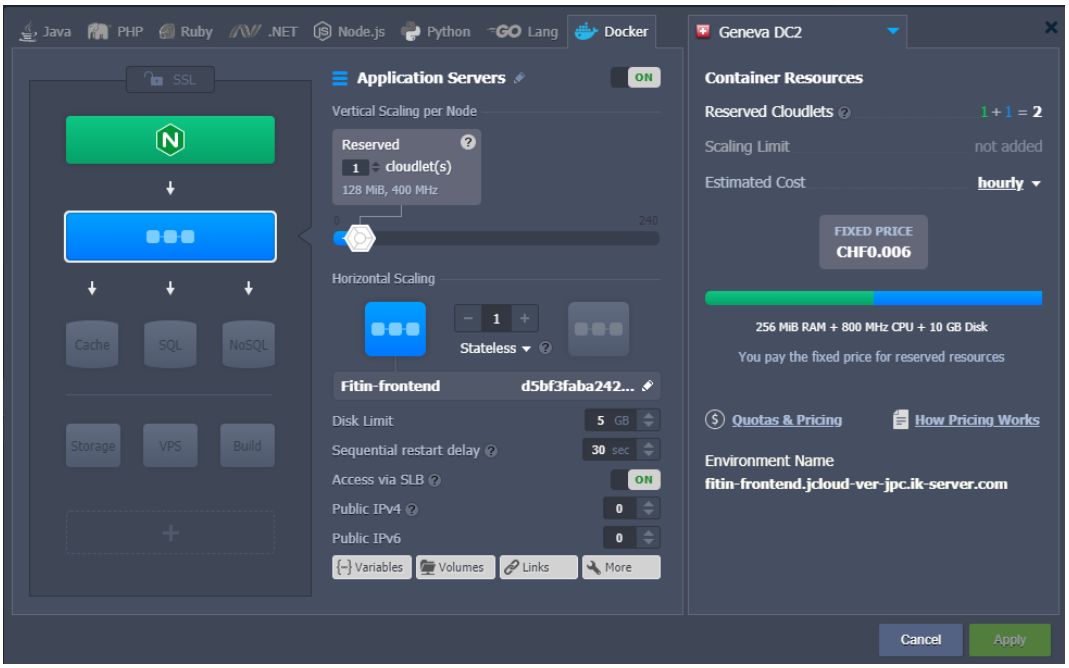

Figure 57: Jelastic Frontend

Tout comme le backend, aucune fourchette n'est définie en cas d'augmentation de charge.

L'idéal pour le frontend serait de stocker les fichiers générés sur un serveur de fichier static et de faire pointer le domaine sur le fichier index. Si tout cela est derrière un CDN, il est possible de cacher entièrement et indéfiniment l'application en se basant sur l'URL des fichiers javascript générés lors de déploiement d'Angular. Car Angular génère un hash différent pour chaque version. Ce qui forcerait automatiquement le cache à se renouveler lors du déploiement d'une autre version.

Même si pas parfaite, la solution docker actuelle a été choisie dans le but de simplifier et uniformiser le frontend avec le backend. Cette solution est entièrement suffisante pour ce POC.

Le schéma de l'infrastructure frontend se dessine comme suit:

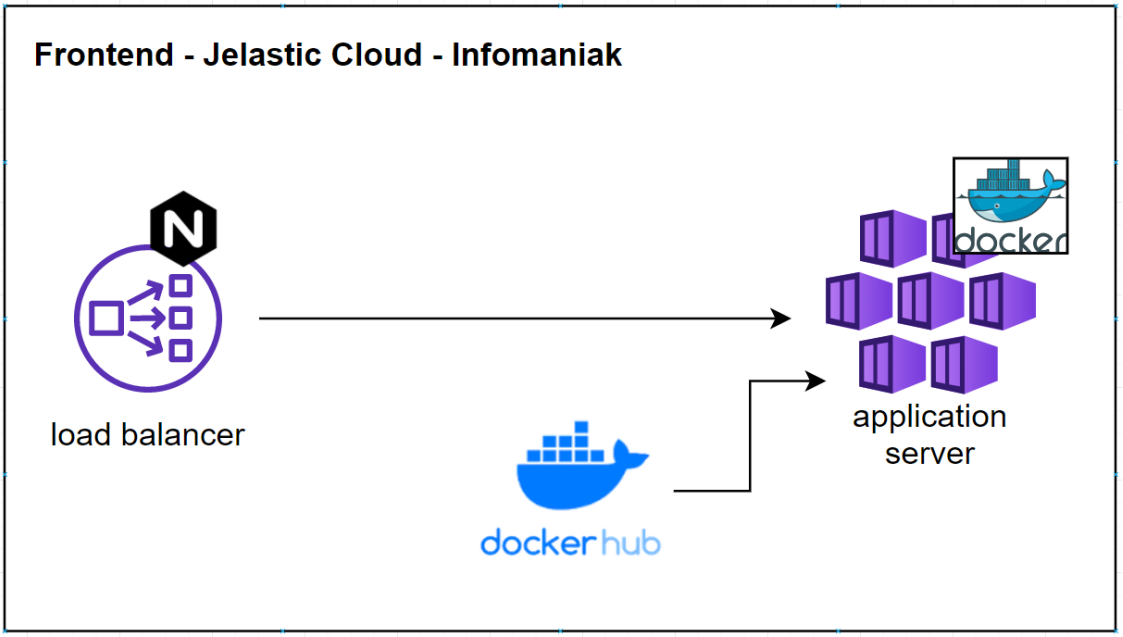

Figure 58: Infrastructure Frontend

8.8.3. Déploiement

Jelastic fournit une API (docs.jelastic.com, s.d., Jelastic Cloud API) qui permet de gérer l'entièreté de son infrastructure via des endpoints. Cependant, j'ai utilisé l'interface pour mettre en place la configuration initiale. L'API est utilisée pour mettre à jour le container docker lorsqu'une nouvelle version est disponible.

A chaque fois que du code est envoyé sur la branche principale, celui-ci génère la création du nouveau container docker automatiquement via CircleCi. Ci-dessous une représentation de la vie du code source.

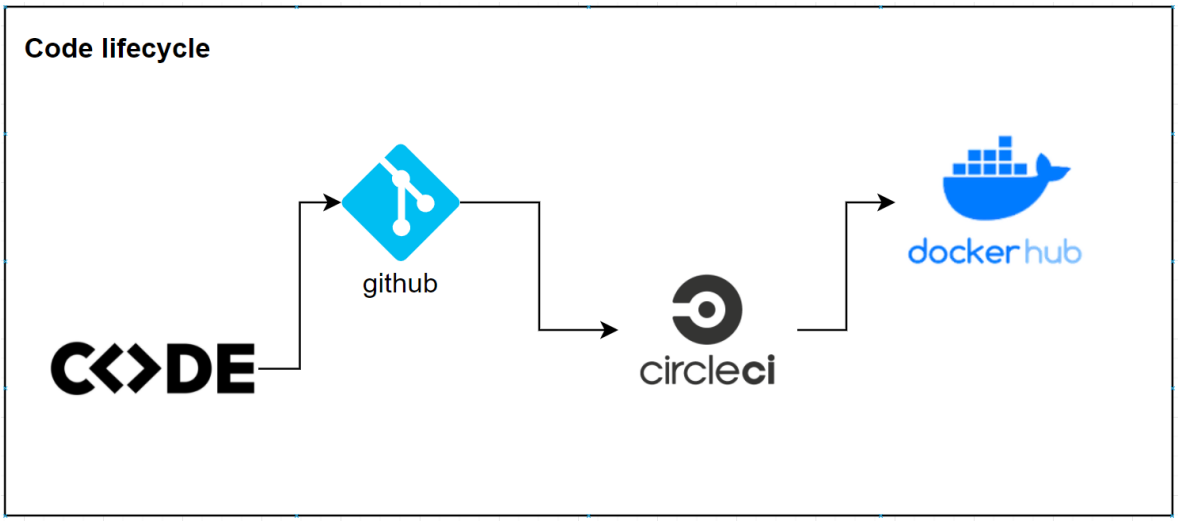

Figure 59: Code lifecycle

Une fois le container docker créé, il suffit d'une commande pour dire à Jelastic de changer le container actuellement utilisé. Expliqué en détail ici: https://docs.jelastic.com/api/#!/api/environment.Control-method-RedeployContainers

Le résultat est résumé en une ligne de code:

```
curl -X POST
"https://${JELASTIC_API_HOST}/1.0/environment/control/rest/redeploycontaine
rs?nodeId=${JELASTIC_NODE_ID}&session=${JELASTIC_ACCESS_TOKEN}&tag=${CIRCLE
_SHA1}&envName=${JELASTIC_ENV_NAME}"
```

C'est donc une requête HTTP POST sur un endpoint avec en paramètre un jeton d'accès, le tag utilisé pour le container docker et l'environnement ciblé.

8.8.4. Configuration DNS

Jelastic assigne chaque environnement à un domaine et nous fournit l'adresse de ce dernier comme on peut le voir dans l'image qui suit:

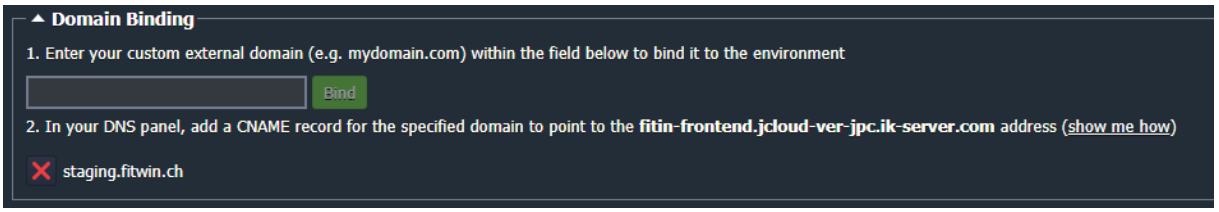

Figure 60: Jelastic DNS

Il nous suffit donc de créer un enregistrement CNAME (Canonical Name) pour le DNS. Un CNAME a pour but d'agir comme alias pour un nom de domaine existant. Il a donc fallu deux CNAME, un pour le frontend et un pour le backend. Comme suit:

SOURCE	TYPE	CIBLE	TTL
api.staging.fitwin.ch	CNAME	fitin-backend.jcloud-ver-jpc.ik-server.com	1 h.
staging.fitwin.ch	CNAME	fitin-frontend.jcloud-ver-jpc.ik-server.com	1 h.

Figure 61: Infomaniak CNAME

8.8.5. Budget

Les coûts étant disponibles en temps et mis à jour chaque heure, il est très facile de suivre ses dépenses. Dans notre cas voici nos frais:

Environnement	Coût par jour	Coût par mois (30 jours)
Backend	0,22 CHF	6,60 CHF
Frontend	0,14 CHF	4,20 CHF
Total	0,36 CHF	10,80 CHF

Tableau 1: Coûts

9. RÉSULTATS

9.1. En fonctionnalités

Ci-après une liste non exhaustive des fonctionnalités principales mise en place pour ce POC.

- Gestion des différents rôles de l'application
- Connexion/déconnexion de l'application
- Administrateurs
    - Inviter des utilisateurs sur l'application et leur assigner un rôle
    - Supprimer des comptes utilisateurs
- Enseignants
    - Création de classe
    - Suppression de classe
    - Ajout d'élèves à une classe
    - Suppression d'élève(s) d'une classe
    - Assignation de devoirs à une classe
    - Assignation et surcharge de devoirs directement à un élève
    - Consultation des exercices terminés des élèves
    - Notation des exercices terminés
- Étudiants
    - Accès aux domaines, thèmes et sous-thèmes afin de voir les exercices disponibles
    - Voir les exercices assignés en devoirs
    - Effectuer les exercices, modifier sa progression
    - Fournir des fichiers aux professeurs (photos, documents etc.)
    - Supprimer les fichiers fournis si nécessaire
    - Voir les exercices qui ont reçu une correction de la part de l'enseignant

9.2. En chiffres

Quelques chiffres pour démontrer le travail effectué durant ce POC.

9.2.1. Backend

Nombre de lignes de code: **25646**

Nombre d'endpoints: **37**

Nombre de tests unitaires: **134**

Nombre de tests end to end (bout à bout): **157**

9.2.2. Frontend

Nombre de lignes de code: **27762**

Nombre de tests GUI: **1 seul pour l'exemple**

9.3. En images

Quelques images pour montrer le résultat visuel de l'application. Bien entendu, cela ne représente pas toute l'application mais donne une petite idée du rendu de l'interface.

L'écran de connexion:

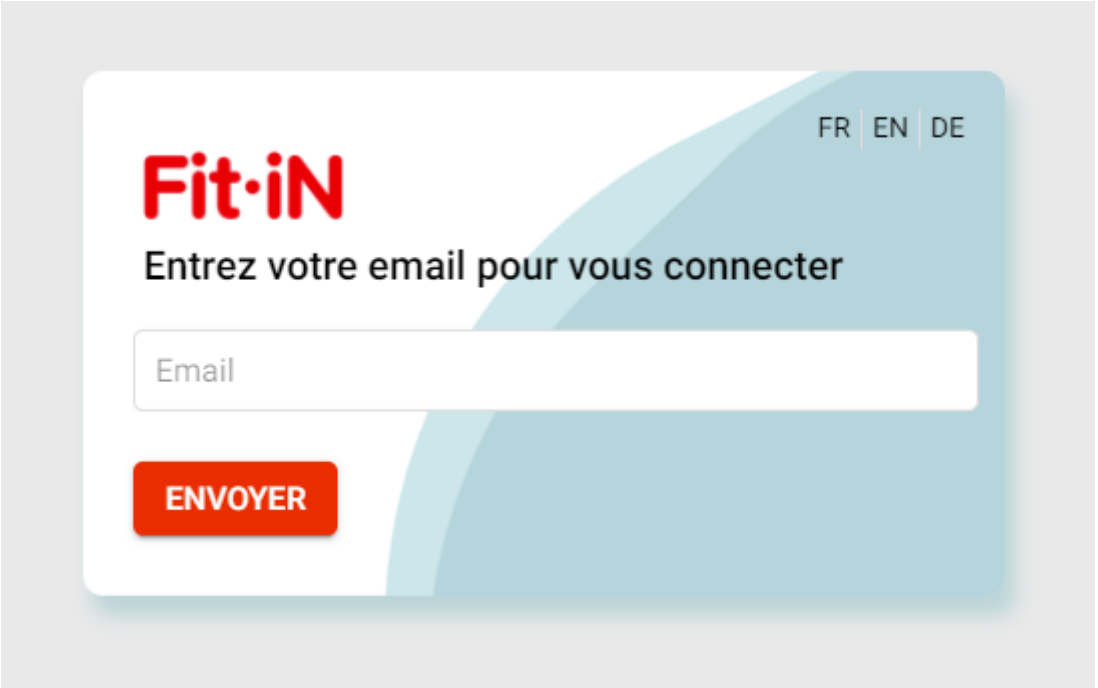

Figure 62: Fit-iN connexion

Gestion des étudiants dans une classe:

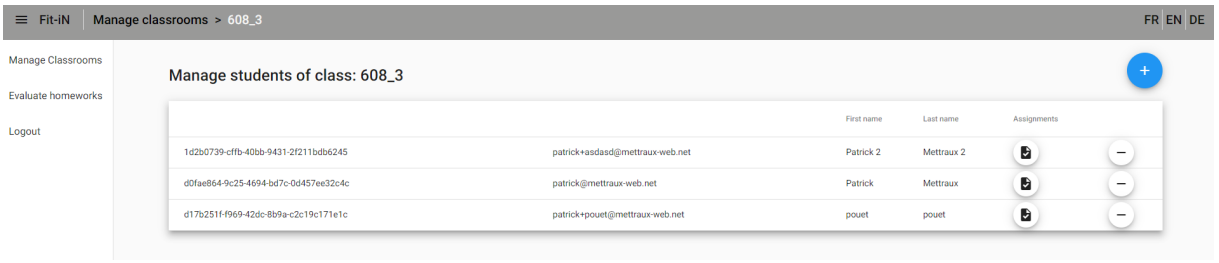

Figure 63: Fit-iN classes

Sélection des domaines pour les étudiants:

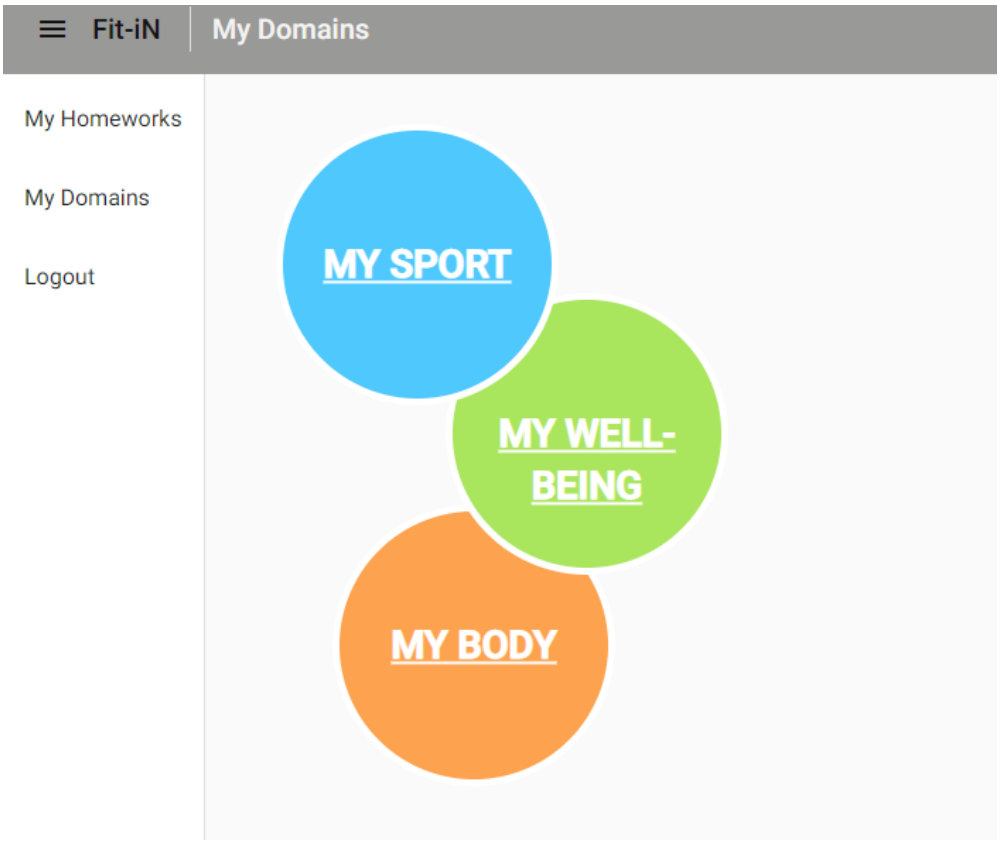

Figure 64: Domaines

Affichage d'une fiche d'exercices

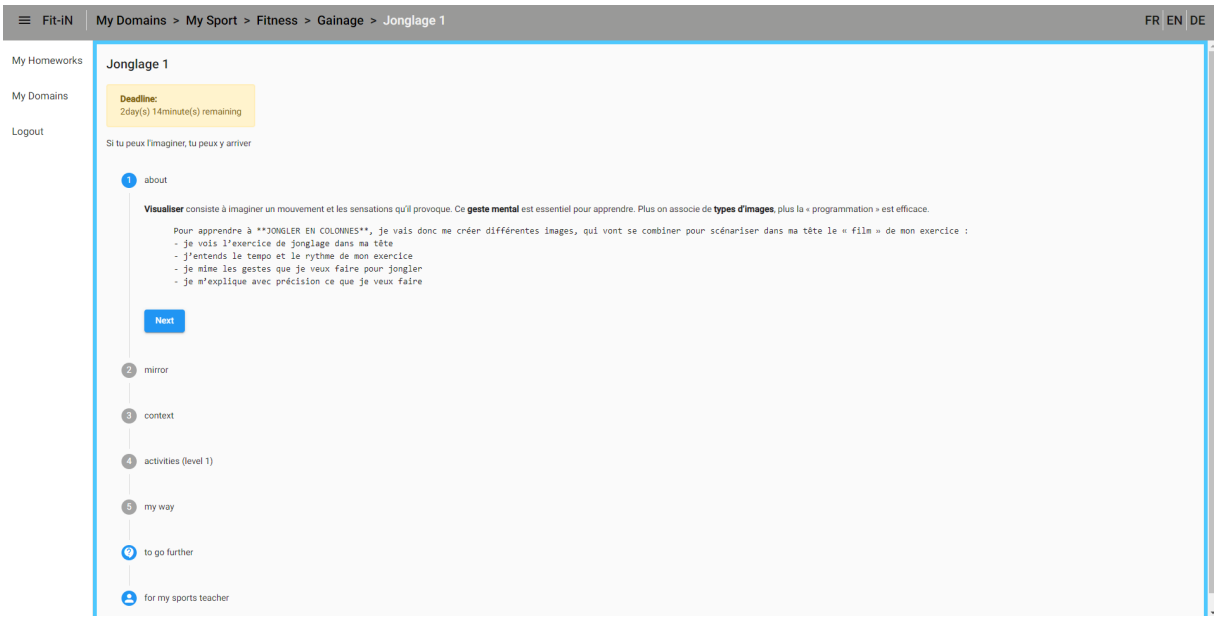

Figure 65: Fiche

10. AMÉLIORATIONS FUTURES

Le but de ce projet étant de fournir un POC, les améliorations possibles sont nombreuses.

10.1. Fonctionnalités:

- Les parties métier pour les contributeurs et les validateurs ne sont pas implémentées. Dans un premier temps, c'est un processus qui peut être fait en dehors de l'application donc clairement non prioritaire.
- Le canal de discussion entre un élève et un enseignant, l'option pour activer ou non le chat est déjà mis en place.
- Le Healthy score. Au fur et à mesure que l'élève effectue des exercices, son healthy score doit changer.
- Tutoriel d'utilisation lors de la première utilisation. Il serait bien pour les utilisateurs non initiés de mettre en place un tutoriel expliquant ce qui leur est demandé et quel est le but de la méthode d'apprentissage.

Tout plein d'autres fonctionnalités sont envisageables, comme par exemple, la mise en place de notifications lorsqu'un événement clé se passe, ou la possibilité pour un élève d'envoyer un défi à des camarades.

10.2. Sécurité

Il faut terminer la mise en place des mesures de sécurité entreprises. Réduire la durée de validité d'un jeton de connexion et mettre en place un système qui permet à l'utilisateur de rester connecté plusieurs jours uniquement s'il en fait la demande.

10.3. Tests

La stratégie de tests mise en place pour le backend est solide, il faudrait maintenant mettre en place des tests pour la partie frontend. Utiliser la structure pour les tests GUI mis en place. Grâce à la structure docker en place, il est même possible d'imaginer des tests sur tout le stack entier et encouragerait même la mise en place de tests par contrat.

10.4. Expérience utilisateur

Cette partie est selon moi celle qui nécessite le plus de travail à l'avenir. C'est un projet qui est entièrement axé sur l'utilisateur, il faut qu'un expert en expérience utilisateur se penche sur ce projet avec le mandant afin de déterminer quelles pourraient être les interfaces idéales et quel flux l'utilisateur devrait suivre afin d'atteindre son but.

10.5. Infrastructure

Jelastic est une bonne solution temporaire, mais étant donné la nature de ce projet et son lien direct avec l'état, il est fort probable que son hébergement finisse sur des serveurs étatiques. Je pense qu'il est important de penser à comment de tels serveurs devront s'auto-adapter selon la charge reçue. Comme évoqué dans le rapport, le frontend pourrait être servi de manière entièrement statique avec un CDN configuré avec un cache total qui permettrait de limiter énormément le trafic reçu sur ses serveurs. Cependant, s'il est possible d'éviter la case "serveur de l'état", je prendrais en considération des solutions telles que AWS qui va ouvrir en 2022 à Zurich.

CONCLUSION

Le projet est terminé et le POC est disponible en ligne. Selon mes premiers retours, le mandant est très satisfait du travail réalisé ; celui-ci concrétise sa vision. En tant que concepteur et coordinateur du projet, il va pouvoir aller de l'avant pour les prochaines démarches d'élaboration du projet.

De mon côté, je suis très content d'avoir pu participer à un projet qui non seulement pousse à la pratique du sport et à son bien-être sous tous ses aspects, mais également à un projet qui remet en question les codes et les manières d'enseigner d'aujourd'hui.

Tout s'est vraiment déroulé sans accroc, la disponibilité du mandant et ses retours lorsque je lui présentais la progression du projet ont vraiment permis d'obtenir une efficacité à toute épreuve lors de la réalisation. Ses retours, selon moi, ont été la clé du succès, dans un projet ou l'on est seul, sans revue de code, avoir les retours au moins sur le résultat au maximum toutes les deux semaines est essentiel. Une preuve de plus sur l'efficacité et le choix d'utiliser des méthodologies Agile de nos jours.

Je ne pense pas que j'aurais pu trouver un meilleur sujet de projet et être mieux encadré qu'avec Fit-iN. J'espère que mon travail servira de base, j'ai conscience qu'il y aura très certainement des changements mais j'espère que la manière de concevoir une application en suivant la méthodologie de https://12factor.net/fr restera.

D'un point de vue personnel, c'est pour moi non seulement la fin de ce projet mais la fin de trois ans de reprise d'étude passé la trentaine. Une expérience bien plus éprouvante que je ne l'aurais pensé, mais néanmoins enrichissante.

J'aimerais finir sur une note écologique: ce projet prône le changement et la remise en question de pratiques anciennes, il serait donc temps d'arrêter d'exiger d'imprimer sur papier en double exemplaire ces travaux de bachelor. Une école se doit d'être exemplaire à tous les niveaux, nous sommes en filière informatique à l'air du numérique, montrons l'exemple.

"Une personne qui n'a jamais commis d'erreurs n'a jamais tenté d'innover."

Albert Einstein

RÉFÉRENCES

Gregor Starc, (2020, 9 novembre) - Physical fitness of slovenian children after the covid-19.
Récupéré sur:
http://avmep.ch/wordpress/wp-content/uploads/2020/12/Presentation-Gregor-Starc.pdf

L'Education Physique en Mouvement, (2020, 4 décembre). Revue professionnelle en ligne.
Récupéré sur:
https://www.hepl.ch/files/live/sites/files-site/files/uer-ep/Publications/%c3%a9ducation-physique-en-mouvement-4-2020-hep-vaud.pdf

Jérôme Nanchen, (s.d.). «FitiN»: mon coach sport et bien-être en autonomie.
Récupéré sur:
http://avmep.ch/wordpress/wp-content/uploads/2021/04/Presentation-Concept-FitIN.pdf

Ken Schwaber and Jeff Sutherland, (2020, novembre), The Scrum Guide.
Récupéré sur:
https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

GDPR, Conditions applicable to child's consent in relation to information society services, (2021)
Récupéré sur:
https://gdpr-info.eu/art-8-gdpr

Réglementation UE pour la protection des données, (2017)
Récupéré sur:
https://www.kmu.admin.ch/kmu/fr/home/savoir-pratique/gestion-pme/e-commerce/reglementation-ue-pour-la-protection-des-donnees.html

Préposé fédéral à la protection des données et à la transparence - PFPDT, (2017, janvier)
Récupéré sur:
https://www.edoeb.admin.ch/dam/edoeb/fr/dokumente/2017/03/die_datenuebermittlunginsauslandkurzerklaert.pdf.download.pdf/la_communicationdedonneesaletrangeren24questions.pdf

Préposé fédéral à la protection des données et à la transparence - PFPDT, (s.d.)
Récupéré sur:
https://www.edoeb.admin.ch/edoeb/fr/home/protection-des-donnees/gesundheit/la-protection-des-donnees-au-cabinet-medical.html

Sébastien Fanti, (2019, 27 mai) - Numérisation du dossier patient et cloud
Récupéré sur:
https://lexing.ch/numerisation-du-dossier-patient-et-cloud

Sébastien Fanti, (2019b) - Big data & protection des données dans le domaine de la santé
Récupéré sur:
https://lexing.ch/wp-content/uploads/2017/10/31.pdf

Guide Social Romand, (s.d.) - Secret professionnel et de fonction
Récupéré sur:
https://www.guidesocial.ch/recherche/fiche/secret-professionnel-et-de-fonction-127

Code pénal suisse, (s.d.) - Violation du secret de fonction
Récupéré sur:
https://www.fedlex.admin.ch/eli/cc/54/757_781_799/fr#art_320

Préposé fédéral à la protection des données et à la transparence - PFPDT, (2013, mai) - Le droit d'accès
Récupéré sur:
https://www.edoeb.admin.ch/edoeb/fr/home/protection-des-donnees/generalites/le-droit-d_acces.html

Ben Balter, (2015, 9 décembre) - 6 motivations for consuming or publishing open source software
Récupéré sur:
https://opensource.com/life/15/12/why-open-source

Shaumik Daityari, (2021, 15 mars) - Angular vs React vs Vue: Which Framework to Choose in 2021
Récupéré sur:
https://www.codeinwp.com/blog/angular-vs-vue-vs-react

formationjavascript.com, (s.d.) - Angular, React ou Vue ?
Récupéré sur:
https://formationjavascript.com/angular2-vs-react

Tomas Holas, (2017) - Angular vs. React: Which Is Better for Web Development?
Récupéré sur:
https://www.toptal.com/front-end/angular-vs-react-for-web-development

S M Asad Rahman, (2019, 20 août) - Why I choose NestJS over other Node JS frameworks
Récupéré sur:
https://medium.com/monstar-lab-bangladesh-engineering/why-i-choose-nestjs-over-other-node-js-frameworks-6cdbd083ae67

Sadissa Babeni, (2020, 24 janvier) - Most Popular Databases in 2020: Here's How They Stack Up
Récupéré sur:
https://ormuco.com/blog/most-popular-databases

Krasimir Hristozov, (2019, 19 juillet) - MySQL vs PostgreSQL -- Choose the Right Database for Your Project
Récupéré sur:
https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres

Michael "Monty" Widenius, (2019, 5 février) - Time to move on
Récupéré sur:
http://monty-says.blogspot.com/2009/02/time-to-move-on.html

optimizdba.com, (s.d.) - MariaDB vs PostgreSQL: Know the difference between the databases
Récupéré sur:
https://optimizdba.com/mariadb-vs-postgresql-know-the-difference-between-the-databases%EF%BB%BF/

Kirk Roybal, (2020, 23 mars) - PostgreSQL is the worlds' best database
Récupéré sur:
https://www.2ndquadrant.com/en/blog/postgresql-is-the-worlds-best-database

nestjs.com, (s.d.) - Database documentation
Récupéré sur:
https://docs.nestjs.com/techniques/database

extremeprogramming.org, (2013, 8 octobre) - Extreme Programming: A gentle introduction
Récupéré sur:
http://www.extremeprogramming.org

CircleCi Docs, (2021) - Sample config.yml Files
Récupéré sur:
https://circleci.com/docs/2.0/sample-config

Marius Nestor, (2016, 2 mai) - Git 2.8.2 Popular Source Code Management System Released with over 18 Bug Fixes
Récupéré sur:
https://news.softpedia.com/news/git-2-8-2-popular-source-code-management-system-released-with-over-18-bug-fixes-503591.shtml

William G. Wong, (2016, 15 juillet) - What's the Difference Between Containers and Virtual Machines?
Récupéré sur:
https://www.electronicdesign.com/technologies/dev-tools/article/21801722/whats-the-difference-between-containers-and-virtual-machines

semaphoreci.com, (s.d.) - Docker Layer Caching
Récupéré sur:
https://docs.semaphoreci.com/ci-cd-environment/docker-layer-caching

Nicolas Trésegnie, (2019, 24 juillet) - How SuperAwesome scales containers for kidtech
Récupéré sur:
https://www.superawesome.com/blog/how-superawesome-scales-containers-for-kidtech

David Currie, (2017, 13 janvier) - How did we develop software without Docker?
Récupéré sur:
https://www.ibm.com/blogs/cloud-computing/2017/01/13/develop-software-without-docker

Felix Richter, (2021, 5 juillet) - Amazon Leads $150-Billion Cloud Market
Récupéré sur:
https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers

Scott Carey, (2020, 23 janvier) - AWS vs Azure vs Google Cloud: What's the best cloud platform for enterprise?
Récupéré sur:
https://www.computerworld.com/article/3429365/aws-vs-azure-vs-google-whats-the-best-cloud-platform-for-enterprise.html?page=2

Nick Underwood, (2021, 26 mai) - 3 Reasons Why AWS Is Dominating the Cloud Marketplace
Récupéré sur:
https://www.privoit.com/resources/3-reasons-why-aws-is-dominating-the-cloud-marketplace

aws.amazon.com, (s.d.) - Offre gratuite d'AWS
Récupéré sur:
https://aws.amazon.com/fr/free

Werner Vogels, (2020, 2 novembre) - Grüezi Schwiiz! Bonjour la Suisse! Buongiorno Svizzera! An AWS Region comes to Switzerland
Récupéré sur:
https://www.allthingsdistributed.com/2020/11/an-aws-region-is-coming-to-switzerland.html

infomaniak.com, (s.d.) - Jelastic Cloud
Récupéré sur:
https://www.infomaniak.com/fr/hebergement/serveurs-dedies-et-cloud/jelastic-cloud

Yulia Shevchenko, (2017, 12 janvier) - Jelastic and Hidora: Pure Combination of Scalability and Swiss Quality
Récupéré sur:
https://jelastic.com/blog/jelastic-and-hidora-pure-combination-of-scalability-and-swiss-quality

Petr Jahoda, (2021, 8 janvier) - Benchmark PostgreSQL on all three systems: Docker versus native
Récupéré sur:
https://itnext.io/benchmark-postgresql-docker-versus-native-2dde6b5a8552

wordpress.org, (s.d.) - Editing wp-config.php
Récupéré sur:
https://wordpress.org/support/article/editing-wp-config-php

docs.jelastic.com, (s.d.) - Custom Environment Variables
Récupéré sur:
https://docs.jelastic.com/custom-environment-variables

docs.npmjs.com, (s.d.) - Creating a package.json file
Récupéré sur:
https://docs.npmjs.com/creating-a-package-json-file

eslint.org, (s.d.) - Find and fix problems in your JavaScript code
Récupéré sur:
https://eslint.org

cypress.io, (s.d.) - The web has evolved. Finally, testing has too.
Récupéré sur:
https://www.cypress.io

M. Jones, J. Bradley, N. Sakimura, (2015, mai) - RFC 7519 - JSON Web Token (JWT)
Récupéré sur:
https://datatracker.ietf.org/doc/html/rfc7519

Selva Karthik, (2020, 23 décembre) - Introduction to Timing Attacks!
Récupéré sur:
https://medium.com/spidernitt/introduction-to-timing-attacks-4e1e8c84b32b

redis.io, (s.d.) - How fast is Redis?
Récupéré sur:
https://redis.io/topics/benchmarks

nestjs.com, (s.d.) - Validation documentation
Récupéré sur:
https://docs.nestjs.com/techniques/validation

gnu.org, (s.d.) - GNU gettext utilities
Récupéré sur:
https://www.gnu.org/software/gettext/manual/gettext.html

poeditor.com, (s.d.) - How to get human translation services
Récupéré sur:
https://poeditor.com/kb/how-to-get-human-translation-services

poeditor.com, (s.d.) - Pricing plans that scale with your business
Récupéré sur:
https://poeditor.com/pricing

poeditor.com, (s.d.) - API v2
Récupéré sur:
https://poeditor.com/docs/api

docs.jelastic.com, (s.d.) - Jelastic Cloud API
Récupéré sur:
https://docs.jelastic.com/api

ANNEXE

Pour ne pas alourdir le document, la documentation technique se trouve en annexe sur la clef USB.

DÉCLARATION DE L'AUTEUR

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide 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, 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 :

- Jean-Pierre Rey
- Jérôme Nanchen