

Travail de semestre

Thomas Fujise

Thomas Fujise

CFPT 2022

Table des matières

1. Cahier des charges	4
1.1 Sujet	4
1.2 But du projet	4
1.3 Utilisation	4
1.4 Spécifications	4
1.5 Restrictions	5
1.6 Environnement	5
1.7 Livrable	5
2. Documentation technique	6
2.1 Résumé	6
2.2 Abstract	6
2.3 Introduction	6
2.4 Analyse de l'existant	6
2.5 Cahier des charges	6
2.6 Analyse fonctionnelle	6
2.6.1 Cas d'utilisations	6
2.6.2 Sitemap	8
2.6.3 Maquettage	10
2.7 Analyse Organique	25
2.7.1 Mise en place / Envirronement	25
2.7.2 Technologies utilisées	26
2.7.3 Base de données	27
3. Logbook	32
3.1 Journal de bord - Thomas Fujise	32
3.1.1 Lundi, 04 Avril 2022	32
3.1.2 Mardi, 05 Avril 2022	32
3.1.3 Mercredi, 06 Avril 2022	32
3.1.4 Jeudi, 07 Avril 2022	32
3.1.5 Vendredi, 08 Avril 2022	33
3.1.6 Lundi, 11 Avril 2022	33
3.1.7 Mardi, 12 Avril 2022	33
3.1.8 Mercredi, 13 Avril 2022	34
3.1.9 Lundi, 25 Avril 2022	34
3.1.10 Mardi, 26 Avril 2022	35
3.1.11 Mercredi, 27 Avril 2022	38

3.1.12 Jeudi, 28 Avril 2022	39
3.1.13 Vendredi, 29 Avril 2022	40
3.1.14 Lundi, 2 Mai 2022	42
3.1.15 Mardi, 3 Mai 2022	42
3.1.16 Mercredi, 4 Mai 2022	44
3.1.17 Jeudi, 5 Mai 2022	44
3.1.18 Vendredi, 6 Mai 2022	45

1. Cahier des charges

1.1 Sujet

Créer une application WEB regroupant des clients et des coachs de sport pouvant récupérer des données d'entraînements directement depuis un appareil connecté Polar

1.2 But du projet

Le but du projet est de créer une plateforme WEB regroupant pratiquants et coachs afin de gérer une salle de sport et le suivi des pratiquants. Les données d'entraînements suivantes sont récupérées depuis une smartwatch Polar:

- Pulsation cardiaque (Repos/Actif/Après effort)
- Le type d'exercice effectué
- Date et durée de l'entraînement
- Nombre de total de calories brûlées
- Nombre de calories active (Optionnel)
- Les données relatives au sommeil
 - Pulsation cardiaque
 - Cycles de sommeil
 - Interruption
 - Hypnogramme

Il y a également un système de badging pour savoir quand le pratiquant commence et met fin à son entraînement.

1.3 Utilisation

La plateforme Web permet de s'enregistrer soit en tant que pratiquant, soit en tant que coach.

En tant que clients : L'ID utilisateur Polar doit être renseigné lors de l'inscription. Une fois le compte créé, on dispose d'une page profil avec plusieurs données d'entraînement, quelques graphiques pour les illustrer permettant d'observer la progression au fur et à mesure du temps. Lorsqu'un compte client est créé, une carte RFID lui est attribuée afin de badger le début et la fin d'un entraînement. Une fois la fin de l'entraînement badgé, l'API Accesslink Polar permettra de récupérer les données de la séance achevée. Le profil du client sera mis à jour automatiquement. 2 onglets sont également disponibles "Entraînements" et "Diète" qui comporteront les programmes d'entraînements et de nutritions du client.

En tant que coach: Une fois connecté, on dispose d'une page avec la liste de tous les pratiquants dont on a la charge. On peut ajouter un pratiquant en le recherchant par son nom, en cliquant sur le bouton "ajouter" un mail de confirmation sera envoyé au client concerné pour valider la prise en charge. Le coach peut uploader un fichier (.csv/.xls) pour les programmes et la diète sur le profil de ses pratiquants pour leur transmettre directement depuis l'application, lors de la mise à jour du programme d'un pratiquant, celui-ci reçoit un mail pour le prévenir. Un fichier "Template" pour conserver un format commun entre coachs sera disponible sur la page d'accueil, si le format mis à disposition n'est pas respecté le fichier ne sera pas pris en compte.

1.4 Spécifications

- Les données d'entraînements sont récupérées avec l'API Accesslink de Polar en Python sous forme d'objet, elles sont vérifiées et stockées directement dans la base de données
- Le système de badge fonctionne à l'aide de cartes RFID et un lecteur NFC (ACS ACR 122u)
- Les graphiques liés aux stats sont effectués avec la librairie javascript Chart.js

- Les programmes d'entrainements et de nutritions doivent respecter le format proposé (Lors de l'upload une vérification est effectuée)
- Un système de notification est mis en place pour avertir les utilisateurs des différents événements

1.5 Restrictions

- Pour l'instant, il est uniquement possible d'utiliser des appareils Polar pour les données d'entrainement.
- Le système de badge est utilisé au début et à la fin et non pendant l'entrainement.
- L'API accepte au maximum 500 requête pour 20 utilisateurs différents en 15 min et 5000 pour 100 utilisateurs en 24h
- Les montres utilisées lors des entraînements sont celles du coach (montres déjà enregistrer sur le client qui accède à l'API)

1.6 Environnement

Technologies utilisées :

- HTML5
- CSS3
- Javascript
- SQL
- Python
- Flask (Micro Framework Python)
- Lecteur NFC (ACS ACR122U)
- [API AccessLink Polar](#)
- [Chart.js](#)

Système d'exploitation :

- Développement sur Windows

Versionning / Documentation :

- GitHub
- MarkDown (Mkdocs)

1.7 Livrable

- Fichier zip contenant le projet + bdd
- Documentation technique et journal de bord au format PDF

2. Documentation technique

2.1 Résumé

FitJourney est une application permettant, d'une part, à un coach sportif de gérer le suivi de tous ses clients avec des données d'entraînements récupérées à l'aide de montres connectés Polar. D'autre part, elle permet aux clients d'accéder à leurs programmes d'entraînements et de nutrition. L'application permet également aux clients de visionner le détail de chacune de leurs séances.

FitJourney est une application WEB réalisée avec le framework Flask du langage Python, elle repose principalement sur l'API Polar Accesslink qui permet l'utilisation des données des montres connectées

Ce document reprend toutes les étapes du projet qui a été réalisé dans le cadre du travail de diplôme de la formation Technicien ES en informatique de Thomas Fujise.

2.2 Abstract

FitJourney is an application that allows personal trainers to follow up all their clients training data that is retrieved using Polar connected watches. It also allows clients to access their training and diet plan. The application allows clients to look at the details of each of their workouts.

FitJourney is a WEB application made with Flask Python framework. It relies mainly on the Polar Accesslink API which allows the use of data from connected watches.

This document contains all the steps of the project, which was carried out within the diploma project of the IT Technician formation of Thomas Fujise

2.3 Introduction

Il existe très peu d'outil qui permet à un coach sportif de gérer sa salle de sport avec le suivi de tous ses clients. C'est pourquoi, j'ai décidé de créer une application qui permettrait de gérer une salle de sports ainsi que le suivi des membres. Ayant passé un diplôme de coach sportif l'année passée, j'étais à l'aise avec les besoins qu'un professionnel aurait en cas d'utilisation de l'application.

Cette application permet de gérer la salle de sport avec les cartes de membres qui permettent l'accès à la salle ou encore les montres connectés pour enregistrer les données d'entraînements des clients. Elle permet également le suivi des clients.

2.4 Analyse de l'existant

2.5 Cahier des charges

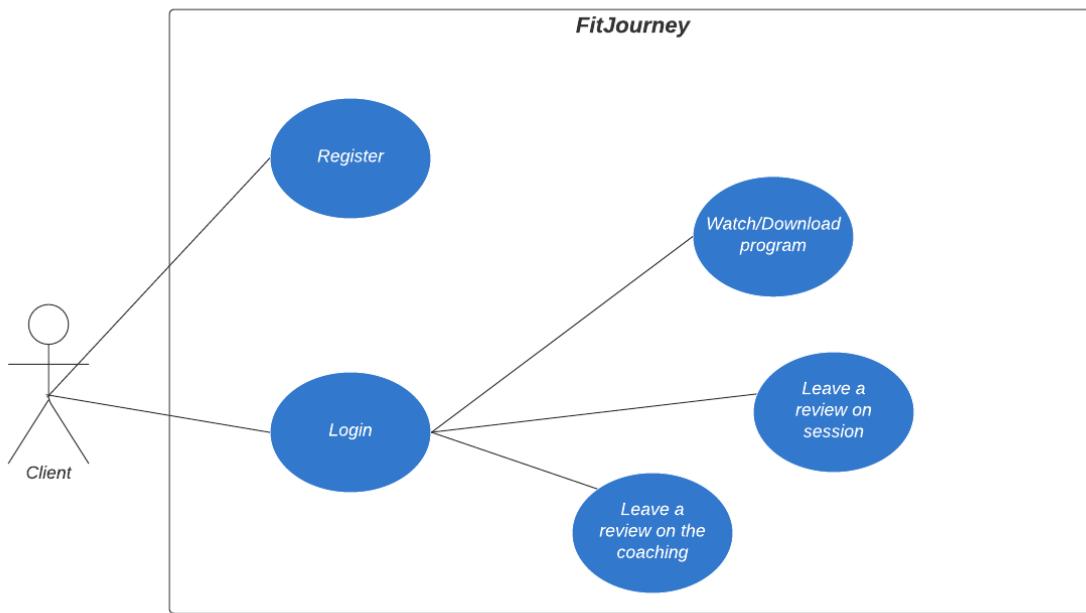
2.6 Analyse fonctionnelle

2.6.1 Cas d'utilisations

2 cas d'utilisations sont possibles avec l'application.

Client

Le cas d'utilisation pour client :



Le client n'a que 2 possibilités sur l'application :

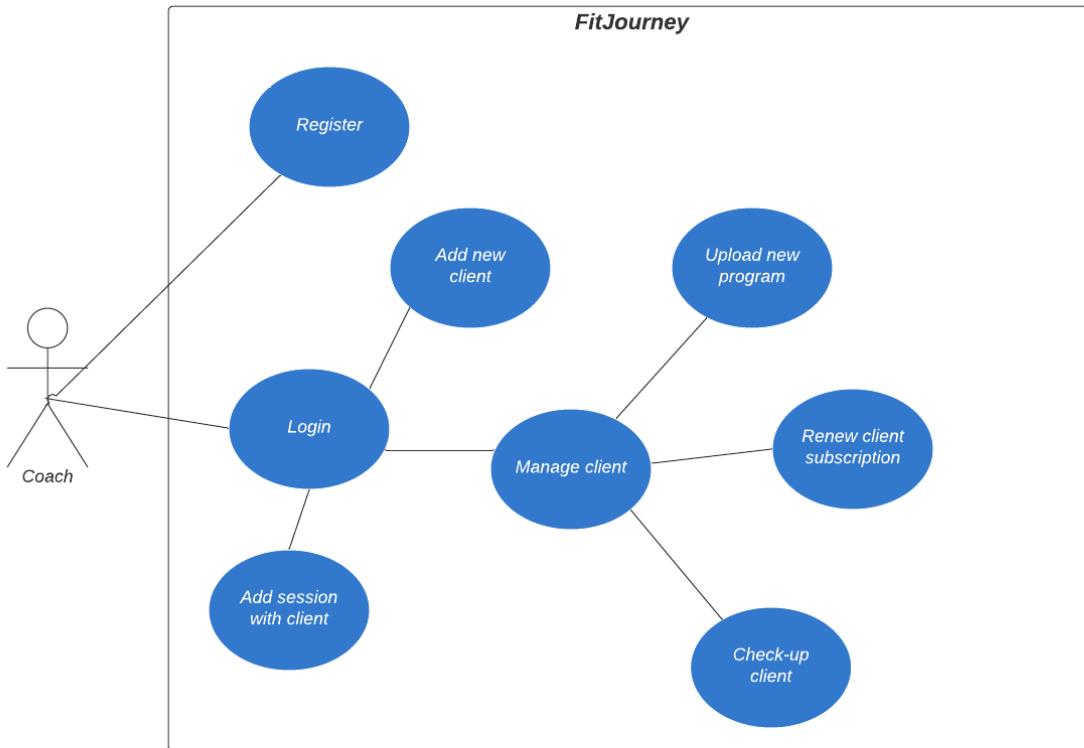
- S'enregistrer
- Se connecter

Si le client se connecte à l'application, il a alors accès à plusieurs fonctionnalités :

- Visualiser/Télécharger ses programmes (Entrainement et Nutrition)
- Laisser un retour sur une session effectué
- Laisser un retour sur le coaching de manière générale

Coach

Le cas d'utilisation pour coach :



Le coach à lui également 2 possibilitées en arrivant sur l'application (Enregistrement et connexion).

Une fois connecté, le coach a accès à une multitude de fonctionnalités :

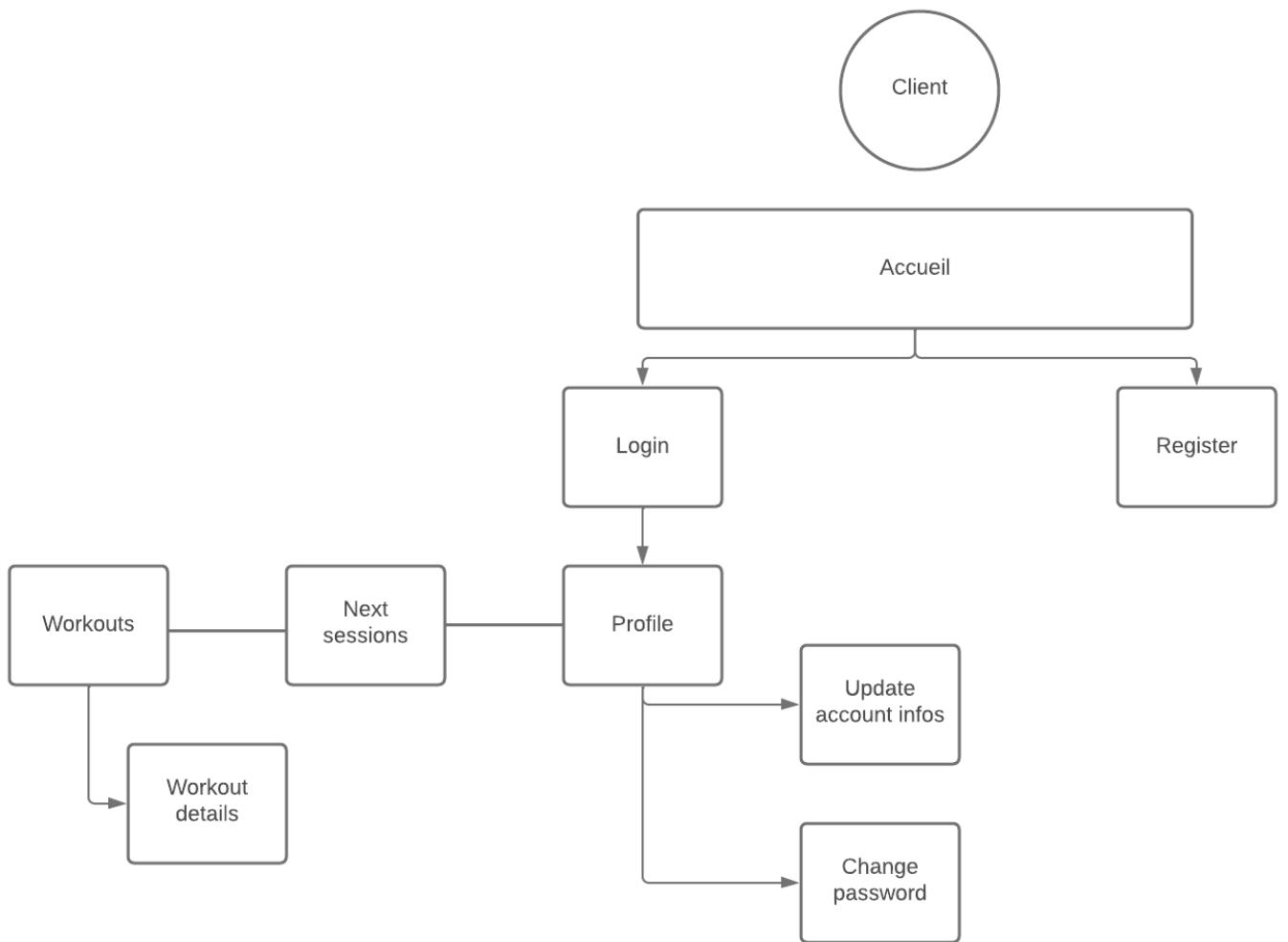
- Ajout d'un nouveau client
- Ajouter une session avec un client (Prise de rendez-vous)

Il a également accès à des fonctionnalités pour gérer ses clients :

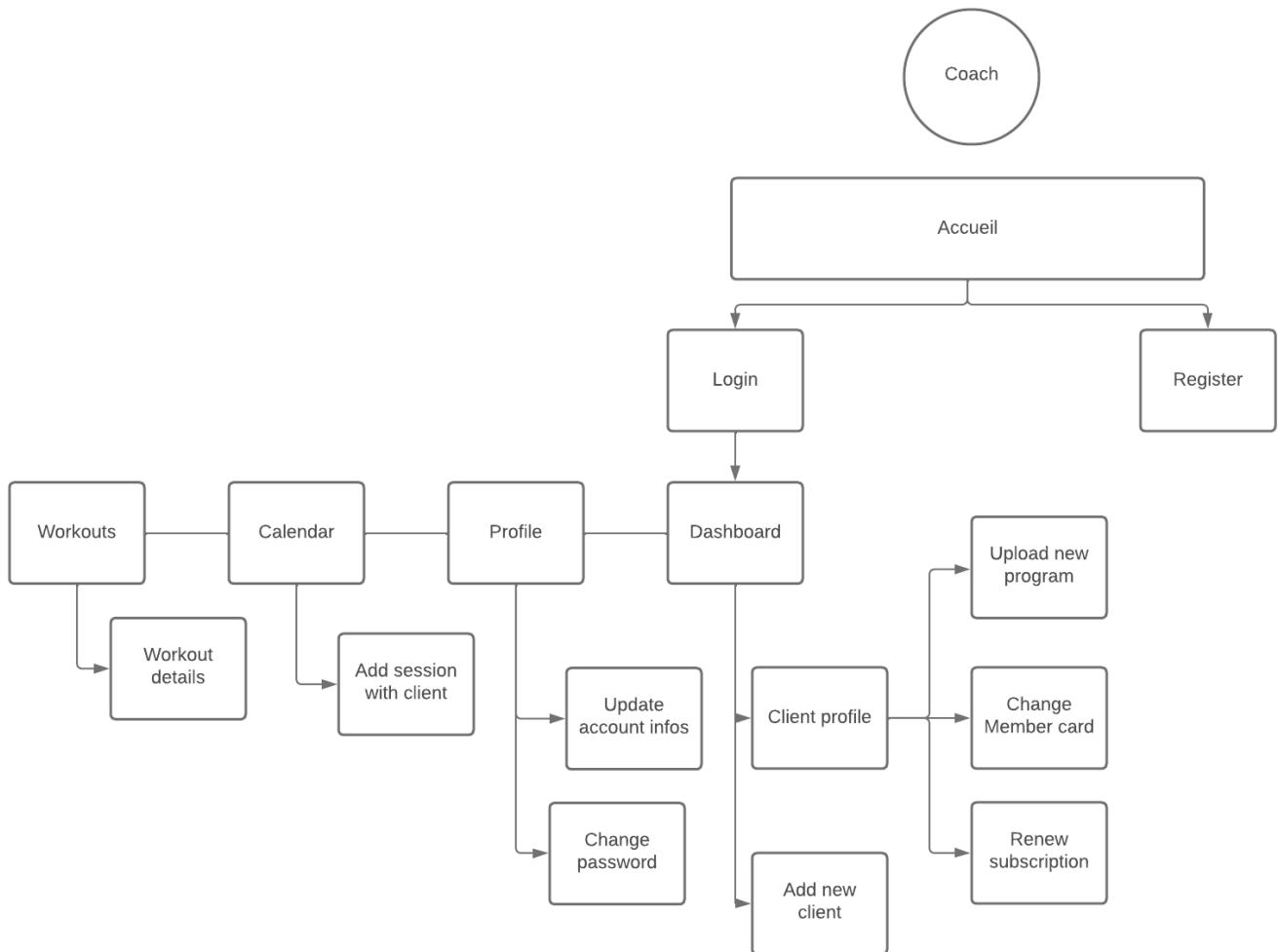
- Importation des programmes (entraînement et nutrition)
- Renouveler l'abonnement souscrit par le client
- Effectuer un bilan avec un client

2.6.2 Sitemap

La sitemap de l'application possède 2 alternatives, 1 pour les clients et 1 pour les coachs.

Sitemap Client

Sitemap Coach



2.6.3 Maquettage

Pour préparer les interfaces, j'ai réalisé des maquettes avec l'outil Figma. Les maquettes m'ont permis de mettre à plat les éléments nécessaires pour les interfaces et ont évité de perdre trop de temps lors de la création des interfaces.

L'application FitJourney propose 3 niveaux d'accès :

- Visiteur
- Client
- Coach

En tant que visiteur

Lorsqu'on arrive sur l'application sans être authentifié, seuls 3 pages sont accessibles.

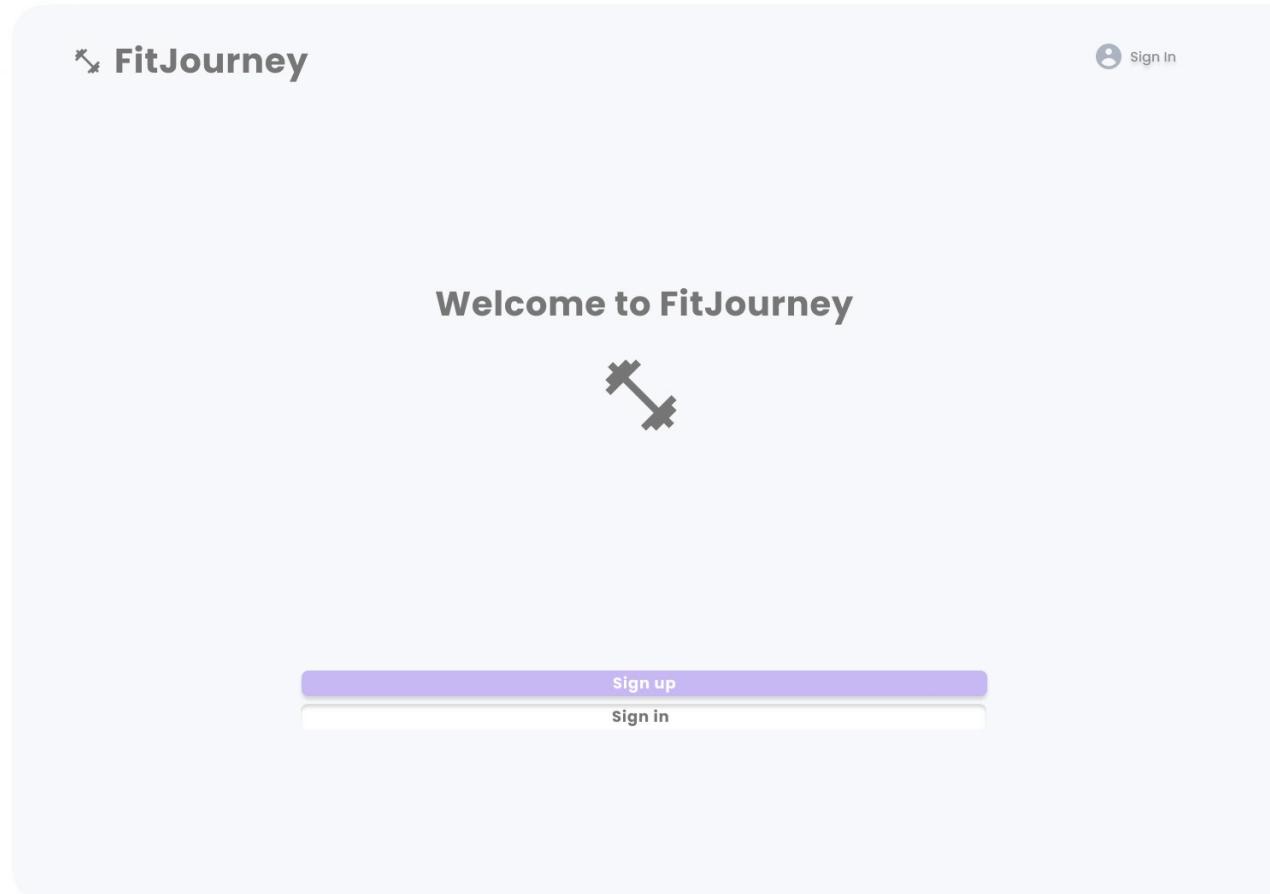
BARRE DE NAVIGATION



La barre de navigation disponible en tant que visiteur. Elle est visible sur le côté gauche de l'écran à la vertical. Sans être connecté à l'application seul 2 boutons sont disponibles :

- Accueil
- Login

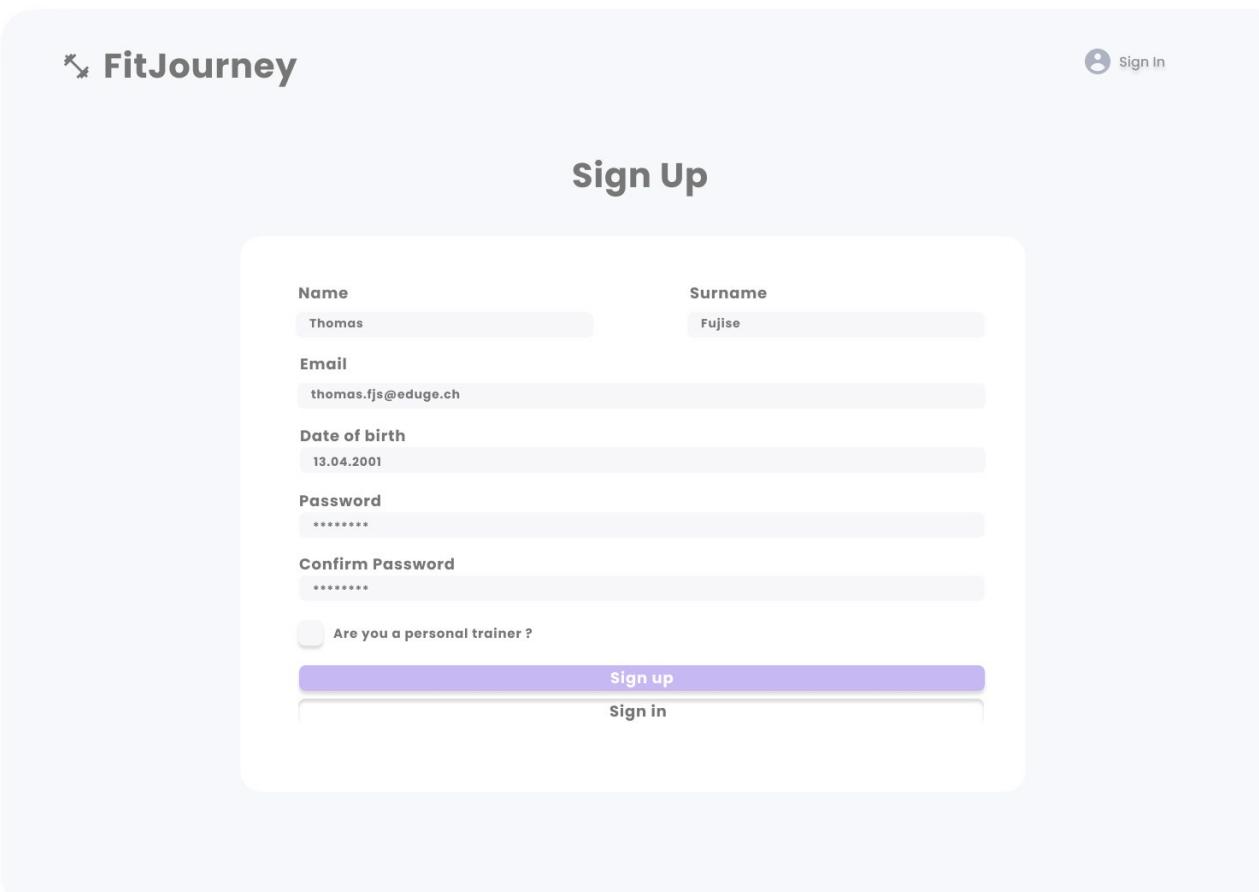
PAGE D'ACCUEIL



La page d'accueil est très basique et propose 2 boutons :

- 1 bouton de connexion
- 1 bouton pour s'enregistrer

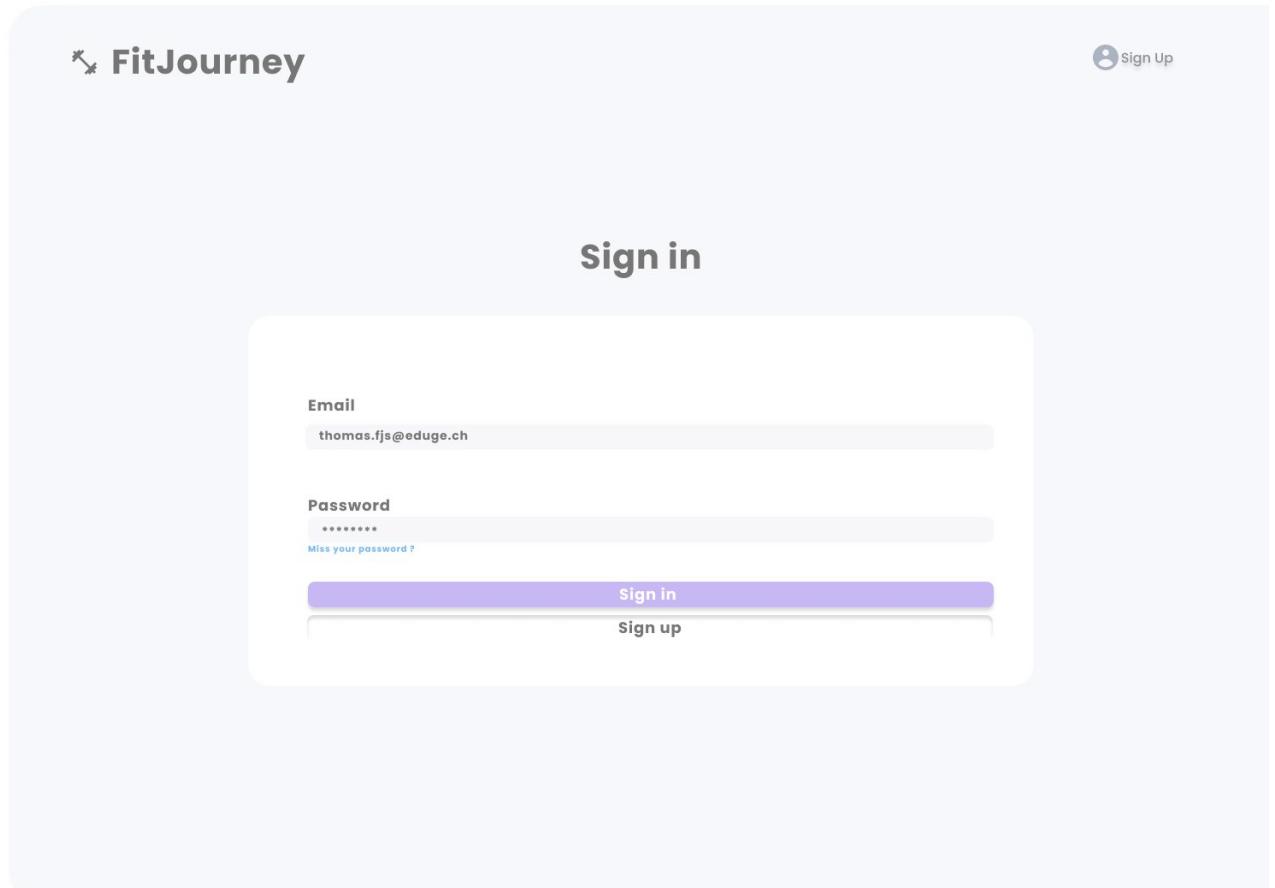
PAGE D'ENREGISTREMENT



The image shows a user interface for a sign-up process. At the top left is a house icon and the logo 'FitJourney'. At the top right is a user icon and the text 'Sign In'. The main title 'Sign Up' is centered above a form. The form fields include: 'Name' (Thomas), 'Surname' (Fujise), 'Email' (thomas.fjs@eduge.ch), 'Date of birth' (13.04.2001), 'Password' (*****), 'Confirm Password' (*****), and a checkbox for 'Are you a personal trainer?'. Below the form are two buttons: a purple 'Sign up' button and a white 'Sign in' button.

La page d'enregistrement permet aux utilisateurs de s'enregistrer, une option est disponible pour permettre la création d'un nouveau compte coach.

PAGE DE CONNEXION



La page de connexion permet aux utilisateurs de se connecter. Un lien est disponible si le mot de passe a été oublié.

En tant que client

Si on se connecte à l'application en tant que client, 4 pages supplémentaires sont disponibles.

BARRE DE NAVIGATION



La barre de navigation disponible en tant que client verticalement à gauche de l'écran. 3 boutons de navigation supplémentaires sont disponibles :

- [Profil](#)
- [Agenda](#)
- [Entrainement](#)

PAGE PROFIL

The mockup displays the FitJourney profile page. At the top left is a house icon, and at the top right are a bell icon and a user profile for Thomas Fujise. The main area is divided into three sections:

- Profile:** Contains fields for Name (Thomas), Surname (Fujise), Email (thomas.fjs@eduge.ch), Date of birth (13.04.2001), Address (Chemin des Curiades 35), NPA (1233 Bernex), Height (1.85 m), Weight (99 kg), and a "Subscription until" date (21.05.22). Buttons for "Update" and "Change Password" are present.
- Reports:** A green-bordered section showing "Last reports" from 10.03.2022: one general report (blue button) and one session report (orange button).
- Workout Activity:** A red-bordered section featuring a bar chart of weekly activity levels (01 to 30) and a summary of this week's gains: 5835 Calories burn, 120 bpm Heart rate avg, and 08 h Time Working out.

La page profil permet au client de modifier ses informations personnelles. Plusieurs boutons sont disponibles :

- 1 bouton "Update" pour appliquer les modifications effectuées sur les informations du compte
- 1 bouton pour modifier le mot de passe
- 1 bouton pour importer une photo de profil.

Il a également accès à des statistiques sur les données des entraînements qu'il a effectués cette semaine (en rouge) et la liste des bilans généraux et de session que l'utilisateur a posté (en vert). Il a la possibilité d'ajouter un nouveau bilan général en cliquant sur le bouton au-dessus.

PAGE AJOUT BILAN

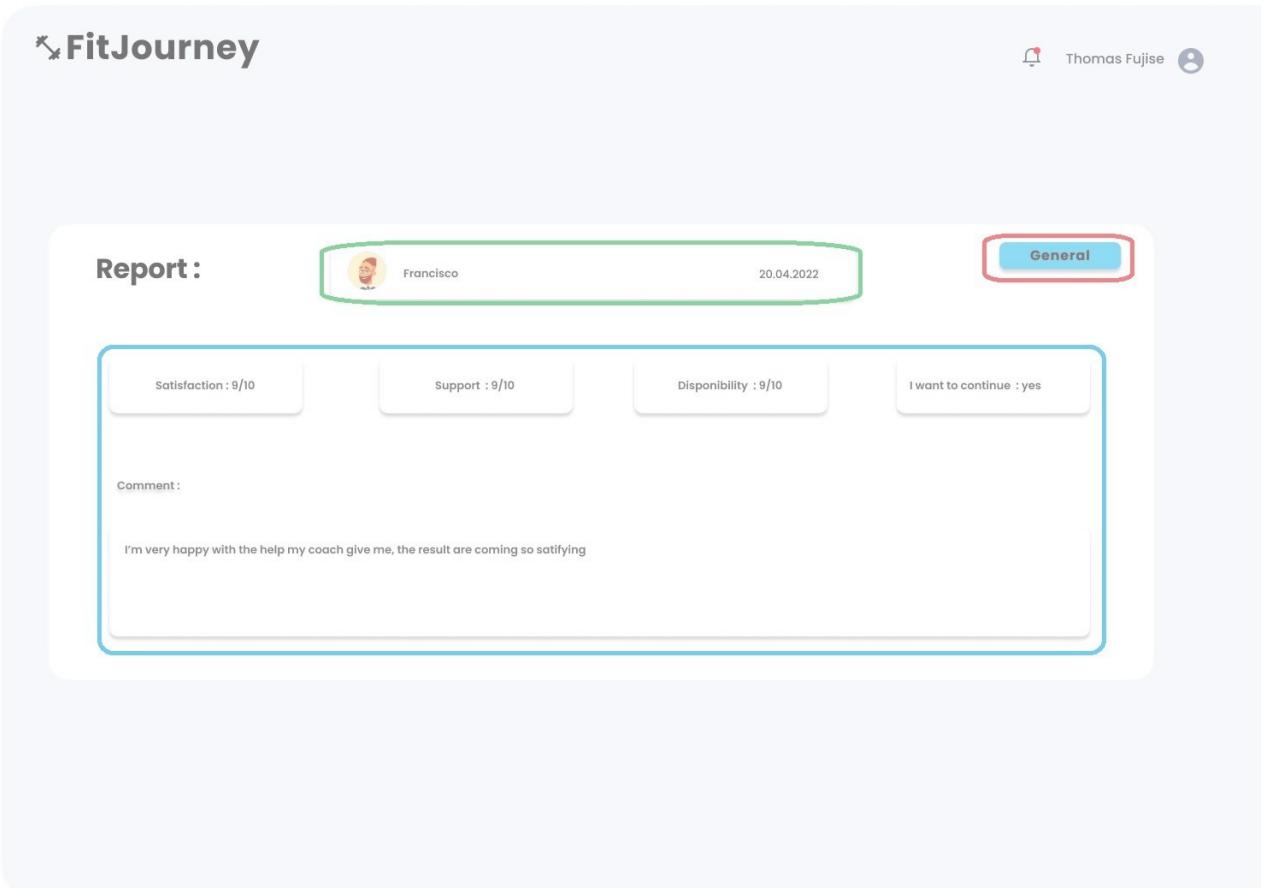
The mockup shows the 'New report' screen of the FitJourney app. At the top right, there is a user profile for 'Thomas Fujise' with a small photo and a gear icon. The main area is titled 'New report'. It features three horizontal rating scales: 'Satisfaction' (value 7), 'Support' (value 8), and 'Disponibility' (value 9). Each scale has a range from 1 to 10. Below these scales is a large input field labeled 'Comment' with a placeholder 'Type your comment here...'. At the bottom is a prominent purple button labeled 'ADD'.

La page "Ajout Bilan" permet au client de noter, soit la qualité du suivi effectué par le coach, soit une session effectuée.

Dans la zone rouge, l'élément qui est évalué (un coach ou une session).

Dans la zone bleue, les différents éléments à noter ainsi qu'une zone pour ajouter un commentaire.

PAGE BILAN



La page "Bilan" permet de voir le bilan ajouté, soit un bilan sur le suivi de manière générale, soit un bilan sur une session effectuée.

Dans la zone rouge, on peut voir de quel type de bilan il s'agit (Bilan général sur le coaching ou bilan d'une session).

Dans la zone verte, on retrouve le client et la date à laquelle le bilan a été posté.

Dans la zone bleue, on retrouve les éléments qui ont été noté par le client.

PAGE LISTE ENTRAINEMENTS

The mockup displays the 'Workouts' section of the FitJourney app. On the left, there is a vertical sidebar with icons for Home, Profile, Calendar, and Exercises. The main area shows a table with four rows of workout data:

Workouts	Date	Duration	Calories burned	Avg Heart Rate
Weightlifting	02.04.2022	2:13	569 cal	111 bpm
Cycling	30.03.2022	0:57	489 cal	89 bpm
Weightlifting	02.04.2022	1:43	369 cal	85 bpm

Below the table, there is a large button with a right-pointing arrow.

Cette page affiche la liste de tous les entraînements effectués par le client. L'utilisateur peut cliquer sur chaque élément de la liste pour avoir les détails de l'entraînement.

PAGE DÉTAILS ENTRAINEMENT

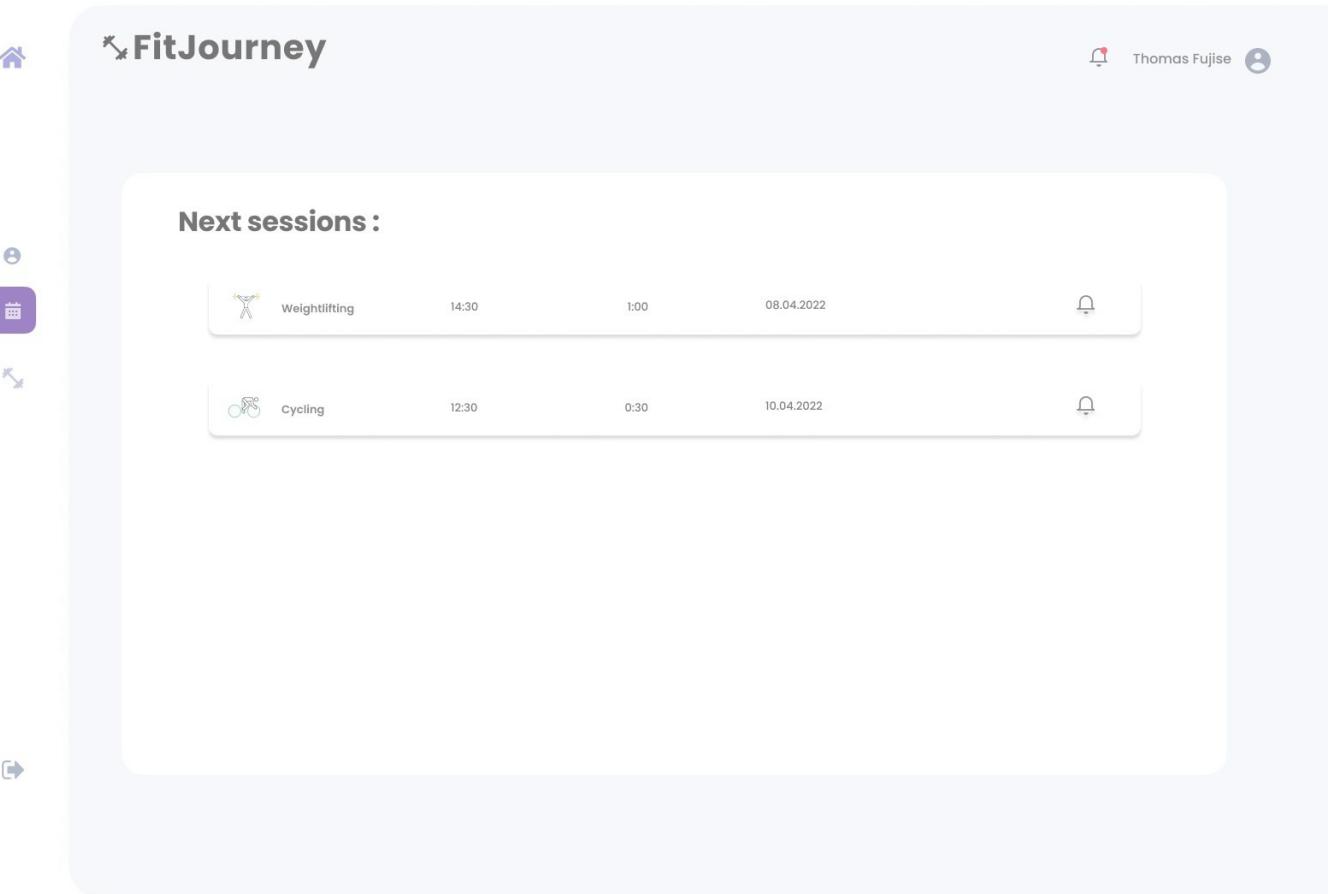
The mockup displays the 'Workout' details screen. At the top right is a red-bordered button labeled 'ADD REPORT'. Below it, the workout type is 'Weightlifting' and the date is '02.04.2022'. Key metrics shown are '120 bpm Heart rate avg', '569 cals Burned', and '2:13 Duration'. A graph titled 'Heart rate : (during workout)' plots heart rate over time, with data points at 01, 03, 06, 09, 12, 15, 18, 21, 24, 27, and 30 minutes.

Cette page affiche les détails d'un entraînement avec les données récupérées à l'aide de la smartwatch. On peut retrouver des informations comme :

- Le nombre de calories brûlées
- Les pulsations cardiaques par minute avec un graphique montrant l'évolution durant l'entraînement
- La durée de l'entraînement
- Le type d'entraînement

Le client peut ajouter un bilan en cliquant sur le bouton (encadré en rouge) pour donner son ressenti sur la séance.

PAGE PROCHAINE SESSION



Cette page affiche les prochaines sessions d'entraînements avec un coach du client

En tant que coach

Si on se connecte à l'application en tant que coach, on a alors accès à 5 autres pages.

BARRE DE NAVIGATION



La barre de navigation disponible en tant que client verticalement à gauche de l'écran. 1 bouton supplémentaire est disponible :

- [Dashboard](#)

PAGE TABLEAU DE BORD

The dashboard features a header with the FitJourney logo and a user profile for Thomas Fujise. On the left, there's a sidebar with icons for Home, Profile, Analytics, and Settings. The main area is divided into two sections: 'Next session' (green border) and 'Clients' (red border).
Next session: Displays session details for 'Josh':

- Subscription: Valid
- Last Workout: 04.04.2022 12:00
- Card ID: 22721902134
- Active (last 24h): Green dot

Session time: 14:30 (yellow dot)

Duration: 01 h (green dot)

Clients: Shows two clients:

- Josh:** Valid subscription, last workout 04.04.2022 12:00, Card ID 22721902134, Active (green dot).
- Francisco:** Expired subscription, last workout 15.03.2021 18:30, Card ID 5432084337, Inactive (red dot).

A blue-outlined button labeled 'ADD NEW CLIENT' is located at the bottom of the clients section.

La page tableau de bord permet au coach de voir la liste des clients (en rouge) dont il effectue le suivi. Le coach peut cliquer sur le nom d'un de ses clients pour afficher le profil du client concerné.

Il a également accès à un bouton pour ajouter un nouveau client (en bleu). Les informations de la prochaine session avec un client sont disponibles au-dessus de la liste des clients (en vert).

PAGE AJOUT DE CLIENT

The screenshot shows the FitJourney application's 'Add new client' screen. The interface is clean with a white background and light gray input fields. At the top right, there is a user profile icon for 'Thomas Fujise'. On the left side, there are several purple circular icons representing different features: a house (Home), a gear (Settings), a calendar (Schedule), a person (Client), and a right-pointing arrow (Next). The main form has a title 'Add new client' at the top. It contains the following fields:

- Name:** Thomas (highlighted with a green box)
- Surname:** Fujise (highlighted with a green box)
- Email:** thomas.fjs@eduge.ch
- Date of birth:** 13.04.2001
- Height:** 1.85 m
- Weight:** 99 kg
- Subscription type:** 1 Month (highlighted with a red box)
- Starting date:** 04.04.2022
- Subscription until:** 04.05.22

At the bottom of the form is a blue 'Add' button.

La page ajout de client permet au coach d'effectuer la prise en charge d'un nouveau client. Si le client possède déjà un compte, le coach peut sélectionner un compte déjà existant (en vert) ou créer un nouveau compte client.

Si le coach sélectionne un compte déjà existant les champs se remplisse automatiquement. Il faudra uniquement sélectionner le type d'abonnement souhaité (en rouge)

PAGE PROFIL CLIENT

The page profil client permet au coach de visionner le profil de ses clients (en bleu). Plusieurs boutons sont disponibles :

- 1 bouton pour le changement de la carte de membre (Bouton bleu)
- 1 bouton pour le renouvellement de l'abonnement (Bouton vert)
- 1 bouton pour l'annulation de l'abonnement (Bouton rouge)

Il peut également voir les graphiques/statistiques disponibles sur le profil.

Le coach peut ajouter les nouveaux programmes du client (en rouge) à l'aide des boutons d'importation (1 bouton pour le programme d'entraînement et 1 bouton pour le programme de nutrition). Les anciens programmes sont disponibles en cliquant sur les boutons du programme souhaité (bouton vert et orange), leurs dates d'importation sont affichées à côté.

Dans la zone verte, on peut retrouver les différents bilan de satisfaction que le client a ajouté. En haut de cette zone, un bouton d'ajout de bilan est disponible. Il va permettre d'effectuer le bilan du client et d'ajouter les nouvelles informations (Poids, Masse grasseuse, etc)

PAGE BILAN CLIENT

The screenshot shows the FitJourney mobile application interface. At the top, there's a navigation bar with icons for home, profile, and search. The main header is "FitJourney". On the right, it shows the user "Thomas Fujise" with a profile picture.

The main content area is titled "New report". It displays a summary of the client's information:

Name	Surname	Height	Age
Thomas	Fujise	1.85 m	21

Below this, there are three sections of data:

- Weight & BMI:** Weight 98.7 kg, BMI 29.1, Water % 59.9 %, Protein % 12.7 %.
- Muscle Mass & Body Fat:** Muscle Mass % 67.3 %, Body Fat % 27.1 %, Bone Mass % 3.6 %, BMR 2045 kcal.
- Visceral & Bone Mass:** Muscle Mass 66.4 kg, Body Fat 26.1 kg, Visceral Fat 10.0 kg, Bone Mass 3.5 kg.

At the bottom of the report section, there's a "Photos" button with four options: "Front", "Right side", "Left side", and "Back". To the right of this is a green "ADD" button.

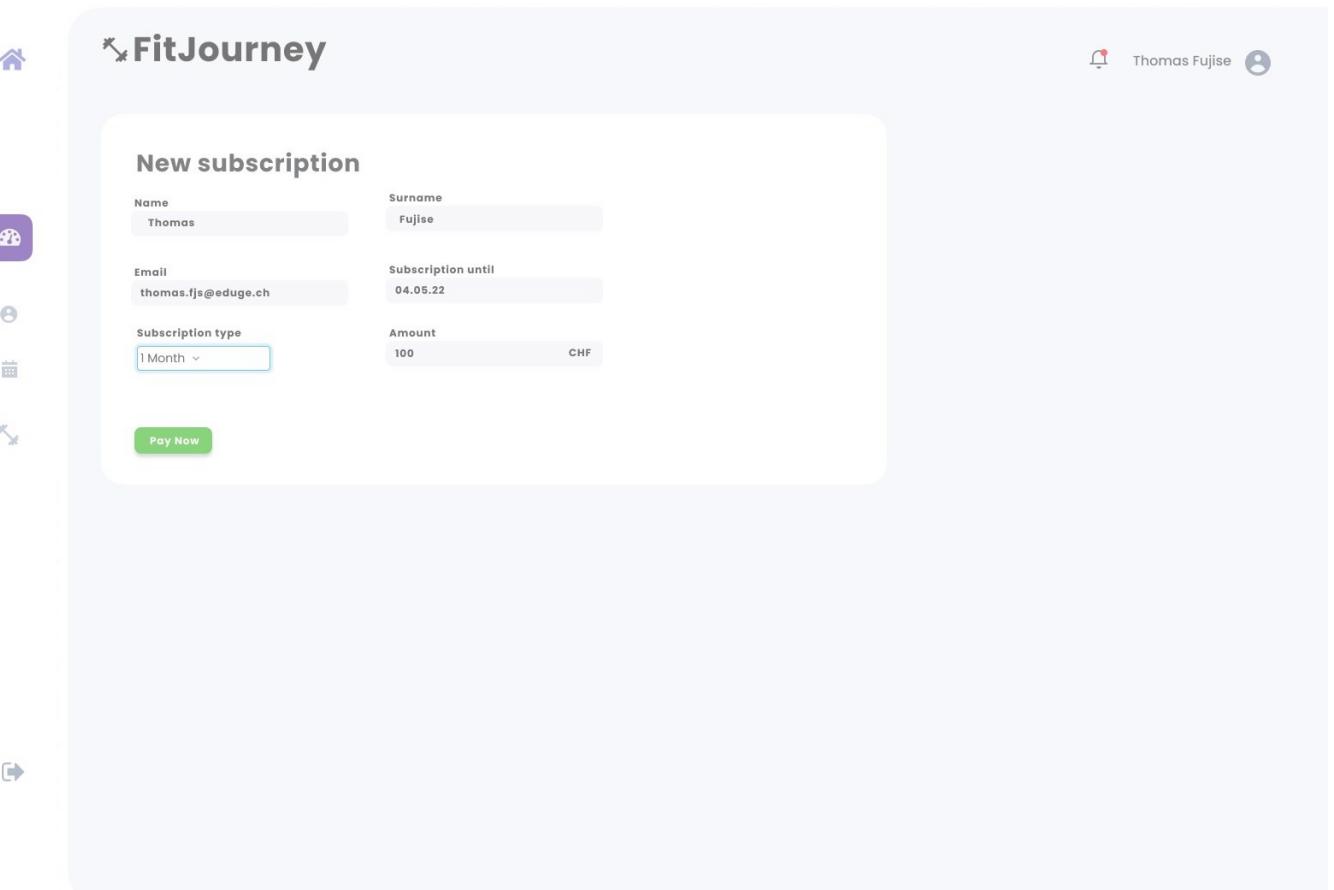
La page Bilan client permet au coach d'ajouter le bilan d'un client.

Dans la zone rouge, on retrouve les infos qui changent le moins voire jamais (Nom, taille ou encore l'âge).

Dans la zone bleue, les informations importantes qui ont pu être récupérées suite à la pesée.

Dans la zone verte, il est possible d'ajouter des photos du physique du client pour pouvoir éventuellement avoir des avant/après en guise de comparaison.

PAGE ABONNEMENT



La page abonnement permet de valider le paiement d'un client (Aucune transaction n'est effectué par l'application). Le coach peut sélectionner le type d'abonnement que le client souhaite prendre, la date d'échéance du nouvel abonnement sélectionné ainsi que son coût seront affichés.

PAGE CALENDRIER

The screenshot shows the FitJourney application interface. At the top, there is a navigation bar with icons for home, calendar, and user profile (Thomas Fujise). Below the navigation bar is a sidebar containing five icons: a house, a person, a calendar, a gear, and a wrench. The main content area features a calendar for March 2022 with the date 04 highlighted. To the right of the calendar is a red-bordered section titled "Sessions:" which lists two sessions: Francisco (Time: 14:30, Duration: 1:00, Type: Weightlifting) and Daniel (Time: 16:00, Duration: 1:30, Type: Weightlifting). Below this is a green-bordered "ADD NEW SESSION" form with fields for Client (David), Time (10:00 AM), Duration (00:00), and Type (Weightlifting). At the bottom left of the main content area is a blue arrow pointing right.

La page calendrier permet au coach d'avoir accès à un calendrier et de visionner les rendez-vous enregistré à la date sélectionnée. Les sessions enregistrés sont affichés dans la zone rouge, avec quelques détails sur la séance.

Dans la zone verte, un bouton pour ajouter une nouvelle session avec un client est disponible. Le coach doit renseigner :

- Le nom du client (Liste déroulante parmi ses clients)
- L'heure de la session
- La durée
- Le type de session

2.7 Analyse Organique

2.7.1 Mise en place / Environnement

Visual Studio Code

J'ai choisi d'utiliser Visual Studio code pour éditer mon code, il est directement relié à mon repo sur Github. Je peux donc directement depuis Visual Studio Code commit tous les changements que j'effectue.

Mkdocs

Mkdocs permet de générer un site statique pour la documentation. Il prend en compte tous les fichiers Markdown (.md) dans le dossier docs/, et un fichier de configuration YAML qui se trouve à la racine du projet. Mkdocs me permet de visionner mes fichiers Markdown en direct et à l'aide de l'extension with-pdf, de générer un fichier PDF de tous mes fichiers Markdown.

GitHub

Pour pouvoir garder un suivi constant de mon projet, j'ai choisi d'utiliser GitHub comme outil de contrôle de version.

Voici comment est structuré le github :

```
FitJourney
├── docs
├── src
├── mkdocs.yml
├── README.md
└── .gitignore
```



- Le dossier docs/ contient les fichiers de documentation et de journal de bord
- Le dossier src/ contient tout le code source de l'application

Trello

Trello est un outil de gestion de projet en ligne, inspiré par la méthode Kanban. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches.



Afin de créer une roadmap pour mon projet, j'ai utilisé Trello pour lister les différentes étapes de mon projet. J'ai ensuite pu définir les échéances pour chaque tâches/étapes avec mon planning prévisionnel.

J'ai créé 5 colonnes :

- Backlog (Liste de toutes les tâches)
- To-Do (Les tâches qui ont été validées et qui sont à faire)
- Doing (Les tâches en cours)
- Testing (Les tâches en cours de test)
- Done (Les tâches terminées)

2.7.2 Technologies utilisées

Python Flask

Flask est un micro-framework Python qui permet la création d'applications web évolutives. Flask dépend de la boîte à outils WSGI de [Werkzeug](#) et du moteur de templates [Jinja](#). Flask était le framework qui répondait le plus à mes besoins, avec l'utilisation de l'API Polar qui est également en Python.



INSTALLATION FLASK

En premier lieu, il faut disposer d'une version à jour de PIP afin d'installer Python Flask avec la commande :

```
pip install Flask
```

UTILISATION FLASK

Pour lancer une application Flask, il faut utiliser la méthode de l'objet Flask :

```
.run()
```

ou lancer directement l'application à l'aide de la commande :

```
Flask run
```

MICRO FRAMEWORK

Un micro framework est un framework qui tente de fournir uniquement les composants absolument nécessaires à un développeur pour la création d'une application. Par exemple dans le cas d'une application Web, un micro framework peut être spécifiquement conçu pour la construction d'API pour un autre service/application.

Le terme micro dans le micro framework signifie que Flask vise à garder le code de base simple mais extensible. Flask ne prendra pas beaucoup de décisions, par exemple quelle base de données utiliser. Les décisions qu'il prend, telles que le moteur de templates à utiliser, sont faciles à modifier. Tout le reste est libre, de sorte que Flask puisse répondre à tous nos besoins et à tous ce que vous ne voulez pas en même temps.

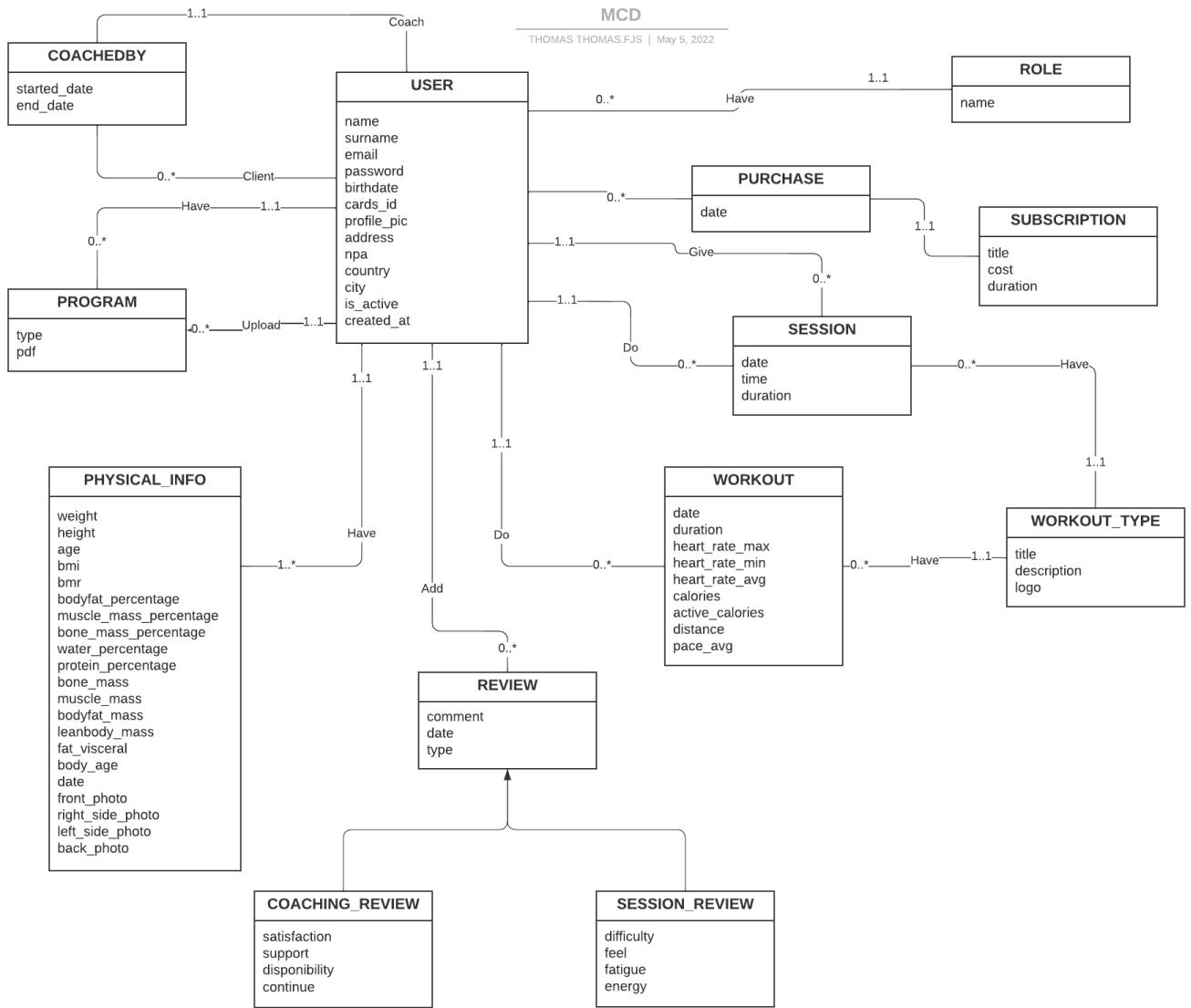
En définissant uniquement le moteur de templates et un système de routes, Flask laisse le choix de personnaliser (en ajoutant des packages) pour la gestion des formulaires par exemple.

2.7.3 Base de données

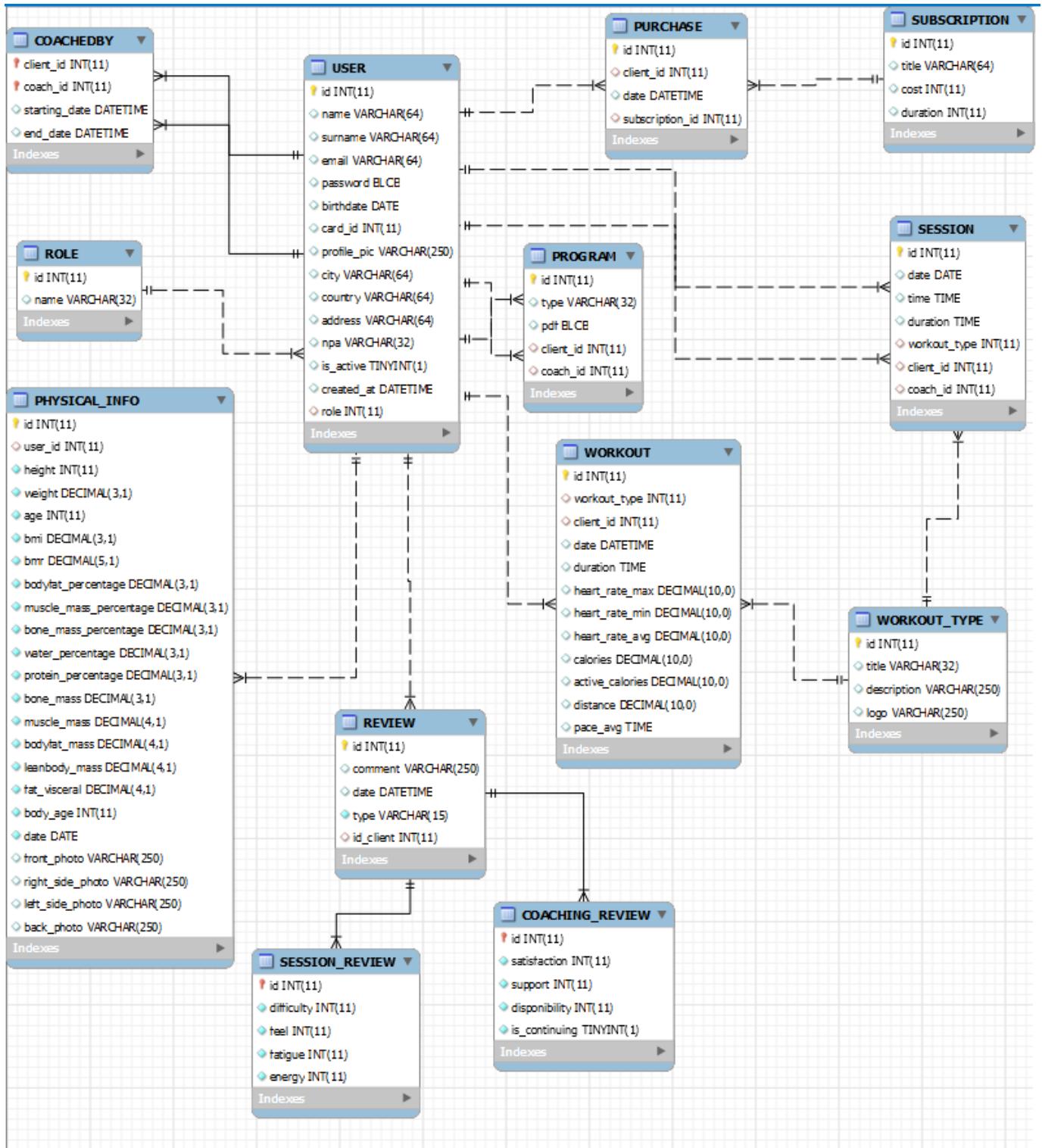
Pour permettre le stockage des données, j'ai créé une base de données nomées "fitjourney". Cette base de données me permet d'enregistrer et stocker toutes les données requis pour le bon fonctionnement de l'application.

MCD

Au lieu de créer la base de données directement, j'ai commencé par créer un MCD pour définir tous les besoins de l'application au niveau de la base de données. Pour faire mon MCD, je suis allé sur LucidChart qui est une plateforme de collaboration en ligne permettant la création de diagrammes et la visualisation de données et autres schémas conceptuels.

**MLD**

Une fois les besoins identifiés à l'aide du MCD, j'ai pu utiliser SQL Alchemy pour créer ma base de données directement.



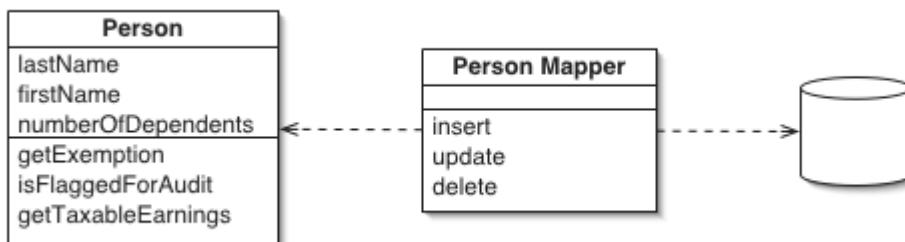
SQLAlchemy

SQLAlchemy est un ORM (mapping objet-relationnel) écrit en Python, il utilise le pattern [Data Mapper](#) et me permet de créer directement mes tables.

Exemple d'initialisation d'une table avec SQLAlchemy :

Data Mapper

Data Mapper est un pattern qui sépare les objets en mémoire de la base de données. Il consiste à transférer les données entre les deux et à les isoler l'une de l'autre. Avec le pattern Data Mapper, les objets en mémoire ne doivent même pas savoir qu'une base de données est présente, ils n'ont pas besoin de code d'interface SQL, et certainement pas de connaissance du schéma de la base de données. (Le schéma de la base de données ignore toujours les objets qui l'utilisent).



Accès

Données de tests

Tables

TABLE USER

La table USER contient tous les utilisateurs de l'application, les coachs ainsi que les clients. les champs email et card_id sont uniques. Le champ card_id représente l'ID de la carte de membre (RFID) qui est attribuée à l'utilisateur.

TABLE PHYSICAL_INFO

La table PHYSICAL_INFO contient toutes les informations physiques récupérées par le coach lors d'un bilan ou d'une prise en charge. La date permet de garder un historique pour visualiser la progression du client. La plupart des données insérées dans cette table peuvent être récupérées à l'aide d'une balance connectée.

TABLE COACHEDBY

Cette table permet de différencier un coach d'un client et permet de retrouver tous les clients d'un coach. Les dates de début et de fin permettent de retrouver des anciens coachs/clients si plusieurs coachs travaillent dans la salle de sport.

TABLE PROGRAM

La table PROGRAM contient les programmes d'entraînement et de nutrition ajouté par un coach.

TABLE ROLE

La table ROLE contient tous les rôles de l'application. Elle permet de définir les accès que possèdent les utilisateurs.

TABLE PURCHASE

La table PURCHASE contient l'historique de toutes les transactions effectuées. Elle permet de retrouver le type d'abonnement que chaque client a souscrit.

TABLE SUBSCRIPTION

La table SUBSCRIPTION contient tous les différents types d'abonnement disponible. Elle permet de connaître la durée et le coût des abonnements souscrits par les clients.

TABLE SESSION

La table SESSION contient l'historique de toutes les sessions effectuées par les coachs avec la date et l'heure de la session ainsi que sa durée. Pour rappel, une session représente un rendez-vous avec un coach. Cela peut être pour un entraînement ou encore un bilan.

TABLE WORKOUT_TYPE

Cette table contient les différents types d'entraînement. Elle permet d'identifier les entraînements effectués par les clients.

TABLE WORKOUT

La table WORKOUT contient toutes les données d'entraînements des séances effectuées. La majorité des données sont obtenues à l'aide des capteurs sur les montres connectées. Les données contenues dans cette table permettent de vérifier l'efficacité et l'intensité de la séance et peuvent être utilisés pour des comparaisons.

TABLE REVIEW

La table REVIEW est la table "mère" des 2 tables : COACHING_REVIEW et SESSION_REVIEW. Elle permet de relier les reviews aux clients.

TABLE COACHING_REVIEW

Cette table contient tous les retours client sur le coaching effectué par le coach. Les champs disponibles ont la satisfaction, le support le coach lui apporte, la disponibilité du coach en cas de besoin et si le client souhaite continuer son suivi.

TABLE SESSION_REVIEW

Cette table contient tous les retours client sur les sessions qu'il effectue avec un coach. Les champs disponibles sont la difficulté, le ressenti de la séance, le niveau de fatigue à la fin de la séance et l'énergie que le client avait en arrivant.

3. Logbook

3.1 Journal de bord - Thomas Fujise

3.1.1 Lundi, 04 Avril 2022

Premier jour du travail de diplôme, il faudrait réussir à mettre en place l'environnement de travail aujourd'hui. Suite à la présentation du déroulement du TD, je change de salle pour m'installer dans la salle à côté. Je vais prendre contact avec M. Jossi, qui est responsable du suivi de mon projet, pour connaître ses exigences en ce début de projet. Je commence à lister les backlogs identifiables à ce stade sur un Trello. Je réalise en parallèle le planning prévisionnel sur Excel. La journée se termine un peu plus tôt (vers 15h) car nous avons une visite à l'HEPIA.

3.1.2 Mardi, 05 Avril 2022

Deuxième jour du travail de diplôme, aujourd'hui il faut que j'avance un maximum sur le planning prévisionnel et l'identification des backlogs. J'ai rajouté les difficultés, la priorité et les dépendances sur les tâches dans le Trello. Cela me permet de mieux m'organiser dans l'exécution de tous les backlogs. M. Garcia est venu vers moi pour définir le nom de mon projet (je l'avais appelé "CoachingTools" par défaut), j'ai finalement décidé de nommer mon projet "FitJourney". Je vais terminer de définir les backlogs avant la fin de la journée.

3.1.3 Mercredi, 06 Avril 2022

Aujourd'hui, je vais modifier encore quelques détails sur mon planning prévisionnel avant qu'il soit terminé. Le planning prévisionnel est donc terminé ainsi que l'identification des backlogs (du moins ceux identifiables à ce stade). Je vais commencer le maquettage des interfaces avec Figma. Pour les maquettes je vais essayer de faire quelque chose d'assez simple, pour éviter de prendre trop de temps sur cette tâche. Je n'avais pas beaucoup utilisé Figma auparavant mais j'ai réussi à bien prendre l'outil en main durant la matinée. J'ai pu terminer déjà quelques maquettes (Profil, Login, Register) et bientôt celle du tableau de bord pour coach. J'ai pu avancer sur les maquettes, il ne me manque plus que la maquette de la page pour créer une séance avec un coach. J'ai pris étonnamment beaucoup moins de temps que prévu pour réaliser les maquettes. Je vais donc pouvoir très bientôt commencer à coder. **RAPPEL:** Il faut que je commence la documentation demain pour déjà documenter les maquettes que j'ai réalisées.

3.1.4 Jeudi, 07 Avril 2022

Aujourd'hui, je vais terminer les maquettes et commencer la documentation technique. Je vais documenter déjà toutes les maquettes que j'ai réalisé pour ne pas avoir à le faire plus tard. J'ai pu terminer les maquettes avant la fin de la matinée. Il faut maintenant que je créer la structure de la documentation technique, je pourrai ensuite commencer à documenter les maquettes que j'ai réalisé.

Suite à une discussion avec un ami coach, j'ai découvert l'existence d'une solution similaire à mon projet **Inithy**. Inithy propose globalement les mêmes fonctionnalités que FitJourney à l'exception que leurs application est orienté coaching à distance là où moi je m'oriente plus sur le coaching en salle (d'où les cartes de membres RFID). Inithy est toujours au stade de démo, il faut les contacter pour avoir un accès à leurs application qui n'est pas encore ouvert à tous.

Je vais leurs envoyer une demande pour essayer d'avoir accès à leurs application et pouvoir analyser leurs application un peu plus en détail.

J'ai pu commencer la documentation des maquettes que j'ai réalisé. J'ai eu une réflexion sur l'enregistrement d'un nouveau coach, j'envisage deux cas possible :

- Soit l'application est délivré avec un compte coach(admin) et le formulaire d'enregistrement n'est disponible que lorsqu'on est authentifié en tant que coach(admin) (C'est le coach qui rempli le formulaire pour les nouveaux clients)
- Soit rajouter un champ lors de l'inscription avec un code pour créer un nouveau compte coach.

3.1.5 Vendredi, 08 Avril 2022

Fin de la première semaine sur ce travail de diplôme, j'ai pu commencer la documentation et documenter les maquettes que j'ai réalisé. Suite à ma réflexion sur l'enregistrement d'un nouveau coach, j'ai décidé de simplement rajouter une option sur le formulaire d'enregistrement pour créer un compte coach. Si une personne venait à créer un compte coach sans l'être ce ne serait pas très dérangeant car un coach n'a accès qu'aux comptes des clients qu'il suit et pour suivre un client il faut que le client confirme de son côté.

Pour un client 2 manières d'avoir un compte :

- Par le formulaire d'enregistrement au préalable, le coach n'aura qu'à sélectionner le profile client lors de la prise en charge.
- Par le coach (un coach peut créer un compte lors de l'ajout d'une prise en charge d'un client, le compte sera créé avec les infos basiques qui seront demandées (Nom, prénom, date de naissance, adresse) et un mot de passe est généré pour permettre au client de se connecter plus tard)

J'ai ajouté la structure de l'application et initialisé les blueprints. (Les blueprints permettent de séparer l'application en plusieurs modules qui sont ensuite importés au même endroit)

J'ai créé un fichier de config qui permet d'avoir 2 mode :

- Debug
- Production

Qui permet d'avoir des paramètres différents. J'ai également ajouté un fichier run qui permet de lancer l'application. Il charge la configuration correspondant au mode actuel (Debug/Production) et lance l'application Flask.

3.1.6 Lundi, 11 Avril 2022

Début de la deuxième semaine du travail de diplôme, aujourd'hui je vais commencer mes pages HTML. Je pense que cela risque de prendre un peu de temps mais en tout cas les maquettes que j'ai réalisé vont me faire gagner pas mal de temps. J'ai réussi à créer une structure de base pour les fichiers html. J'ai décidé d'utiliser [Sneat](#) qui est un thème CSS open-source publié par [ThemeSelection](#) sous licence MIT. J'ai ajouté un fichier HTML pour les liens CSS qui sera inclus dans toutes mes pages.

M. Maréchal est venu voir l'avancer du projet car il va superviser mon travail de diplôme de "loin". M. Jossi reste mon suiveur principal. M. Maréchal m'a conseillé de faire un diagramme explicatif du fonctionnement de l'application.

J'ai ajouté la route par défaut pour pouvoir tester une page index avec le CSS (pour vérifier que tout marche bien). Pour l'instant, je n'ai pas de problème tout à l'air de fonctionner comme il faut. J'ai fait une classe "prototype" Users avec SQLAlchemy pour empêcher des erreurs bloquantes avec Flask-Login.

Flask-Login est une librairie qui permet de gérer les sessions utilisateurs pour Flask. (Flask-Login gère les connexions, déconnexions et garde la session utilisateur pour savoir l'utilisateur connecté ou qui vient de se déconnecter)

J'ai maintenant une page index qui peut afficher des composants à l'aide du thème CSS Sneat. Demain je poursuivrai la création de mes pages HTML et il faut également que je commence à rajouter des éléments dans la documentation

Ne pas oublier d'avancer la documentation

3.1.7 Mardi, 12 Avril 2022

Aujourd'hui je vais essayer de terminer la partie authentification de l'application (Login/Register). J'utilise FlaskForm pour créer mes formulaires et je pense utiliser Marshmallow pour valider les champs en relation avec la BD (Je regarde quelques exemples d'utilisation avec SQLAlchemy : [Exemple](#)).

Il faut également que je fasse le diagramme qui m'a été conseillé par M. Maréchal pour la description du fonctionnement de l'application.

Pour la sécurité des mots de passe j'ai déjà rajouté 2 méthodes pour permettre de hasher et de vérifier les mots de passe. J'utilise la librairie Python hashlib qui me permet d'hasher les mots de passe avec un salt. Pour éviter d'enregistrer le salt en base, le salt est hashé en sha256 et est placé avant le mot de passe dans la chaîne. Comme un sha256 a toujours 64 caractères, pour vérifier le mot de passe je saute les 64 premiers caractères de la chaîne pour pouvoir comparer avec la saisie de l'utilisateur.

J'ai pu faire un diagramme explicatif des différents processus lors de l'utilisation de mon application en fonction de notre rôle (Client, coach)

Image

J'ai également terminé le login (le register ne devrait pas prendre beaucoup de temps). J'ai eu quelques soucis pour inclure les formulaires avec FlaskForm (avec les imports python). J'ai ajouté une navbar pour mes pages qui servira pour la navigation dans l'application. J'ai juste encore quelques soucis avec le css mais ce n'est pas du tout prioritaire pour le moment.

J'ai regardé pour utiliser un outil de génération de documentations, je pense utiliser Pdoc qui à l'air assez complet et qui permet de générer la documentation sur les librairies incluses dans le projet.

3.1.8 Mercredi, 13 Avril 2022

Dernier jour avant les vacances de pâques, j'essaye d'intégrer pdoc à mon projet mais lorsque je lance la commande pour générer le doc j'obtiens une erreur. Il n'arrive pas à trouver apps qui est le dossier principal.

J'ai pris un peu de retard par rapport à ma planification initiale mais j'ai avancé sur des points que j'étais censé effectuer plus tard dans le projet donc au final je n'ai pas beaucoup de retard. Je vais juste terminer encore ma page register qui va de pair avec la page Login. Il faudra ensuite que je commence à documenter tous le back-end que j'ai déjà effectué. Je pourrai ensuite commencer à travailler sur la base de données car pour le moment j'utilisais une base de données temporaire pour tester mon login.

En faisant des tests sur le login je me rends compte que j'ai une erreur qui s'affiche lors de la connexion. Lorsque j'effectue la requête pour voir si le mail que l'utilisateur a saisi existe je ne rencontre aucune erreur si le mail inséré n'est pas dans la base mais lorsque le mail rentré se trouve dans la base de données alors une erreur s'affiche :

```
string argument without an encoding
```

Après quelques heures d'incompréhension sur cette erreur, je viens d'en trouver la raison. J'avais seulement mis le mauvais type dans l'initialisation du champ mot de passe avec SQLAlchemy.

J'ai enfin finis le register et le login, tout fonctionne comme il faut. Je n'ai plus qu'à rajouter les champs que je veux dans le register (tout le back-end est fonctionnelle). Je vais commencer à documenter les points que j'ai avancé jusque là je pourrai ensuite commencer à travailler sur la base de données.

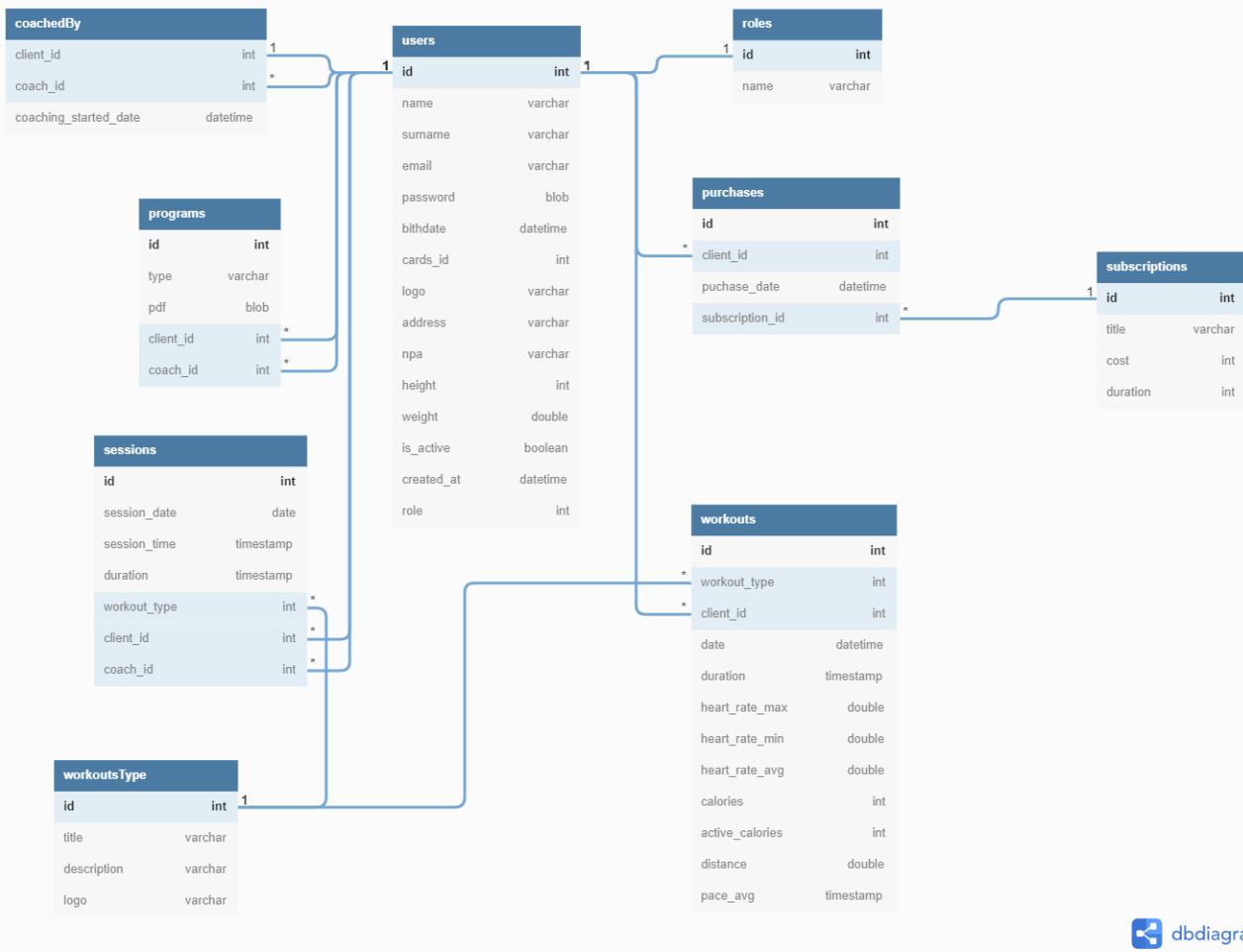
J'ai pu commencé à documenter l'environnement de mon projet mais il reste encore beaucoup de point que je peux documenter déjà maintenant. Pour avancer la documentation en même temps que le projet je vais faire ça en priorité et je continuerai sur la base de données une fois la documentation à jour.

3.1.9 Lundi, 25 Avril 2022

Retour de vacance, je n'ai malheureusement pas pu énormément avancer pendant la semaine et demie de vacance. Le premier rendu intermédiaire a lieu demain. Je vais terminer le diagramme UML que j'ai commencé pour la base de données. J'utilise l'application WEB dbdiagram.io qui permet de créer un diagram et de l'exporter directement en SQL par la suite.

Ma machine a eu un blue screen qui a été très long (30min) à redémarrer. Je n'ai malheureusement pas eu le temps de sauver le diagram que j'avais quasiment terminé, je vais devoir recommencer.

J'ai pu terminer le diagramme UML à l'aide de dbdiagram.io :



dbdiagram.io

Je peux maintenant commencer la création des tables avec SQLAlchemy. J'ai découvert un outil qui est actuellement en BETA [dbdocs](#) qui permet de générer une documentation WEB d'une base de données. Je regarderai l'outil un peu plus en détail plus tard mais je pense que cela pourrait être une bonne idée d'avoir cette documentation en plus de celle qui sera présente dans le document de documentation technique.

Je viens de me rendre compte que je n'avais pas créé le planning effectif pour l'instant. J'ai dupliqué le planning prévisionnel et effectué les quelques modifications pour le planning effectif.

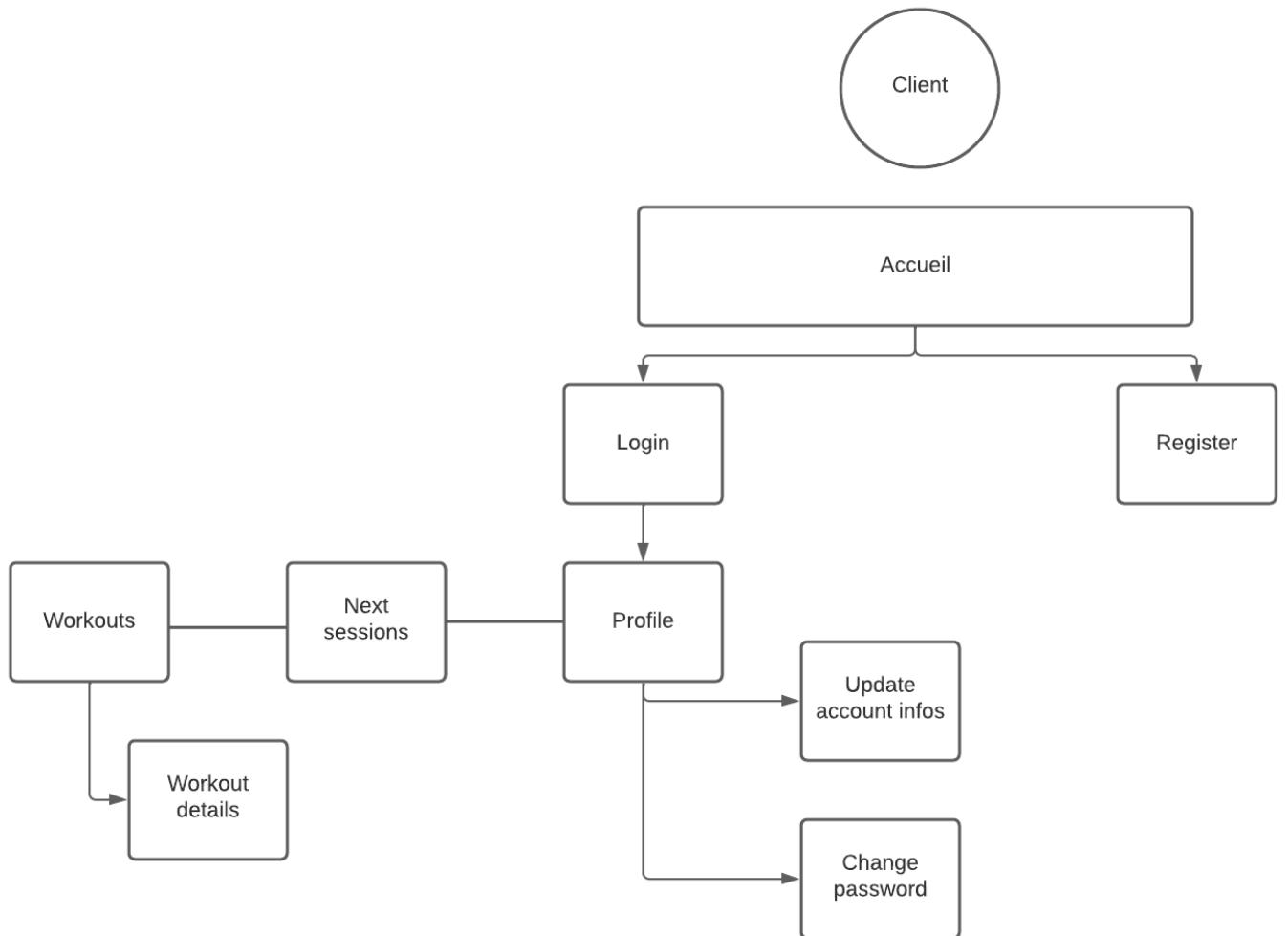
M.Jossi est venu voir l'avancement du projet. Nous avons discuté de plusieurs points notamment le diagramme que j'avais fait pour expliquer l'utilisation de l'application. Le diagramme que j'ai fait était un mix d'un sitemap et d'un diagramme de cas d'utilisation. Je vais donc les refaire séparément. Je vais également faire un MCD pour être sûr de ne rien oublier pour ma base de données. Je pourrai ensuite poursuivre la création de ma base avec SQLAlchemy.

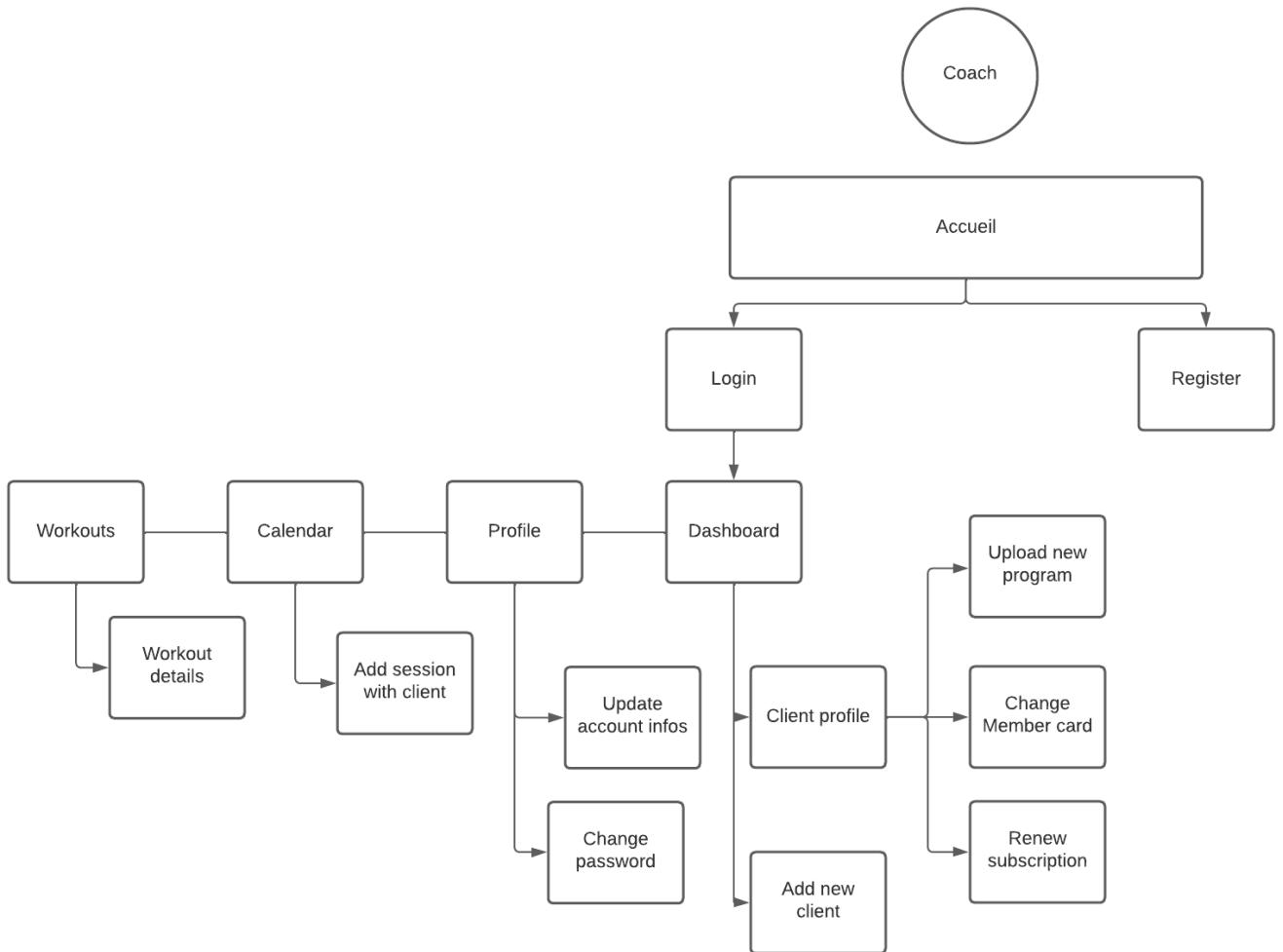
Pour l'évaluation intermédiaire, M.Jossi m'a demandé de remplir la grille d'évaluation de mon côté et nous referons un point mercredi pour voir si tout est ok.

3.1.10 Mardi, 26 Avril 2022

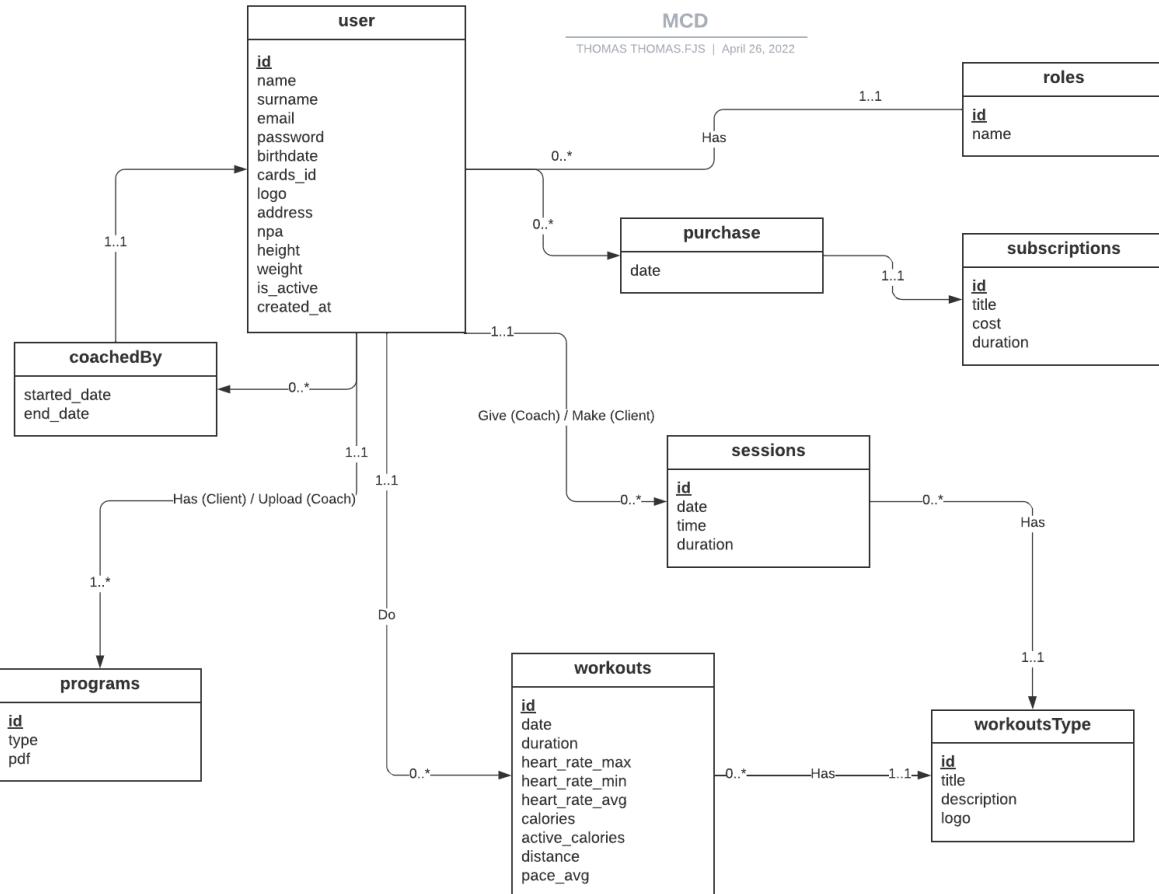
Aujourd'hui je dois compléter la grille pour la première évaluation intermédiaire comme convenu avec M.Jossi.

J'ai pu terminer les 2 sitemaps pour le client et le coach.





Je vais maintenant faire le MCD que j'aurai dû faire avant le MLD pour être sûr de n'avoir rien oublier. Il faudrait que j'avance la documentation sur les diagrammes que je suis en train d'effectuer. J'ai pu terminer le MCD, je le ferai vérifier par M.Jossi car j'ai quelques doutes sur certains points. Je ne sais pas si, dans mon cas, il n'est pas mieux de faire une classe mère (Users) avec 2 enfants (Clients et coaches) pour le MCD.



J'ai ajouté les objets SQLAlchemy en fonction du MCD que j'ai fait au préalable. Je me suis rendu compte que certaines tables devront être remplies au préalable ou je devrais rajouter des écrans pour pouvoir les remplir. Un coach n'a pas moyen d'ajouter un type d'entraînement (la table 'workoutsType') ni d'ajouter un type d'abonnement (table subscriptions). Les relations ont l'air de fonctionner avec SQLAlchemy je vais pouvoir adapter les formulaires de register et login que j'ai fait au préalable pour tester.

J'ai mis également à jour le Trello que je n'avais plus touché depuis quelques jours.

3.1.11 Mercredi, 27 Avril 2022

M.Jossi devrait passer aujourd'hui à la pause pour faire un point et faire l'évaluation intermédiaire comme convenu.

Je vais adapter les formulaires de login/register à la nouvelle base de données. Je viens de me rendre compte que le champ 'DateField' dans mon formulaire, que j'utilise pour renseigner la date de naissance de l'utilisateur s'affiche comme un input texte basique. Pour avoir un input de type date j'ai utilisé le champ 'DateField' mais depuis wtforms.fields.html5.

J'ai rencontré un nouveau problème, en premier les validations des champs du formulaire ne s'effectuaient pas correctement. J'avais en fait simplement oublier d'utiliser un SubmitField (j'utilisais un input HTML en dur à la place). Les validateurs fonctionnent sauf ceux du mot de passe qui sont sensés vérifier que les 2 mots de passes saisies soient identiques. Le formulaire se valide et prend en compte uniquement le premier mot de passe saisi.

J'ai trouvé pourquoi les mots de passe ne se vérifiaient pas, je passais directement à la création de l'utilisateur lors du clique sur le bouton 'register'. Les champs se vérifiaient "eux-mêmes" par rapport au type de données saisies mais le check si les 2 inputs étaient identiques ne s'effectuaient donc pas. J'ai donc changé ma condition pour que je poursuive la création de l'utilisateur uniquement lors de la validation du formulaire.

J'ai réussi à adapter le login et le register avec les bons champs, il ne manque plus qu'à ajouter l'affichage des erreurs.

M.Jossi est passé pour l'évaluation intermédiaire. Globalement tout est ok, j'ai juste pris un léger retard avec toute la mise en place du système que j'avais légèrement sous-estimé. Pour la documentation, j'ai quelques points à ajouter/modifier :

- Ajouter les sitemaps
- Modifier le MCD (Pas de flèches, revoir les cardinalités, double relations pour coach/client)
- Ajouter Installation pour le projet en général (pas seulement Python Flask)
- Ajouter l'aborescence du dossier source (explication détaillée)
- Revoir l'ordre des maquettes (ordre logique)
- Explication des boutons sur les maquettes
- Use case dans l'analyse fonctionnelle
- Schéma pour montrer comment l'application va intéragir avec l'API

J'ai pu terminé la modification du MCD, je vais l'envoyer à M.Jossi pour qu'il puisse vérifier. Il faut maintenant que j'avance sur la documentation.

3.1.12 Jeudi, 28 Avril 2022

J'ai commencé à revoir la description de mes maquettes. Je me suis rendu compte de quelques incohérence sur certaines maquettes comme sur le tableau de bord du coach, la prochaine session était affiché tout en bas de la page. Il était donc plus cohérent d'afficher cette information tout en haut et de mettre la liste de client en bas.

J'ai également eu une réflexion sur les bilans. J'avais totalement oublié de penser au bilan qui est un élément essentiel du coaching. Je vais donc revoir les maquettes et je pourrai donc modifier le MCD quand je retournerai dessus plus tard. Je pense rajouter 2 "types" de bilan, un bilan général qui sert à évaluer le ressenti du client par rapport à sa prise en charge, le suivi de manière général. Le deuxième type de bilan serait un bilan par session (qui ne serait pas obligatoire, au début cela peut ne pas déranger mais si le client effectue beaucoup de séance cela peut vite devenir répétitif) où le client note son ressenti sur la séance. Pour le coach, un bilan est effectué de manière soit hebdomadaire, soit mensuelle pour enregistrer les nouvelles mesures (poids, mensu, chrono, cela dépend du type de coaching).

Pour le bilan, les données que le coach devra renseigner à l'aide d'une balance connecté qui permets leurs acquisition sont :

- Poids
- BMI (Body Mass Index / IMC)
- Body Fat %
- Water %
- Muscle Mass %
- Bone Mass
- BMR
- Fat Visceral
- Lean Body Mass
- Body Fat Mass
- Muscle Mass
- Protein
- Body Age

J'ai pu terminé les nouvelles maquettes, j'ai rajouter une page bilan de session pour l'utilisateur ainsi qu'une page bilan "général" pour noter la qualité du coaching. Du côté coach, j'ai rajouté la page pour effectuer le bilan général avec les informations mentionnées au-dessus.

3.1.13 Vendredi, 29 Avril 2022

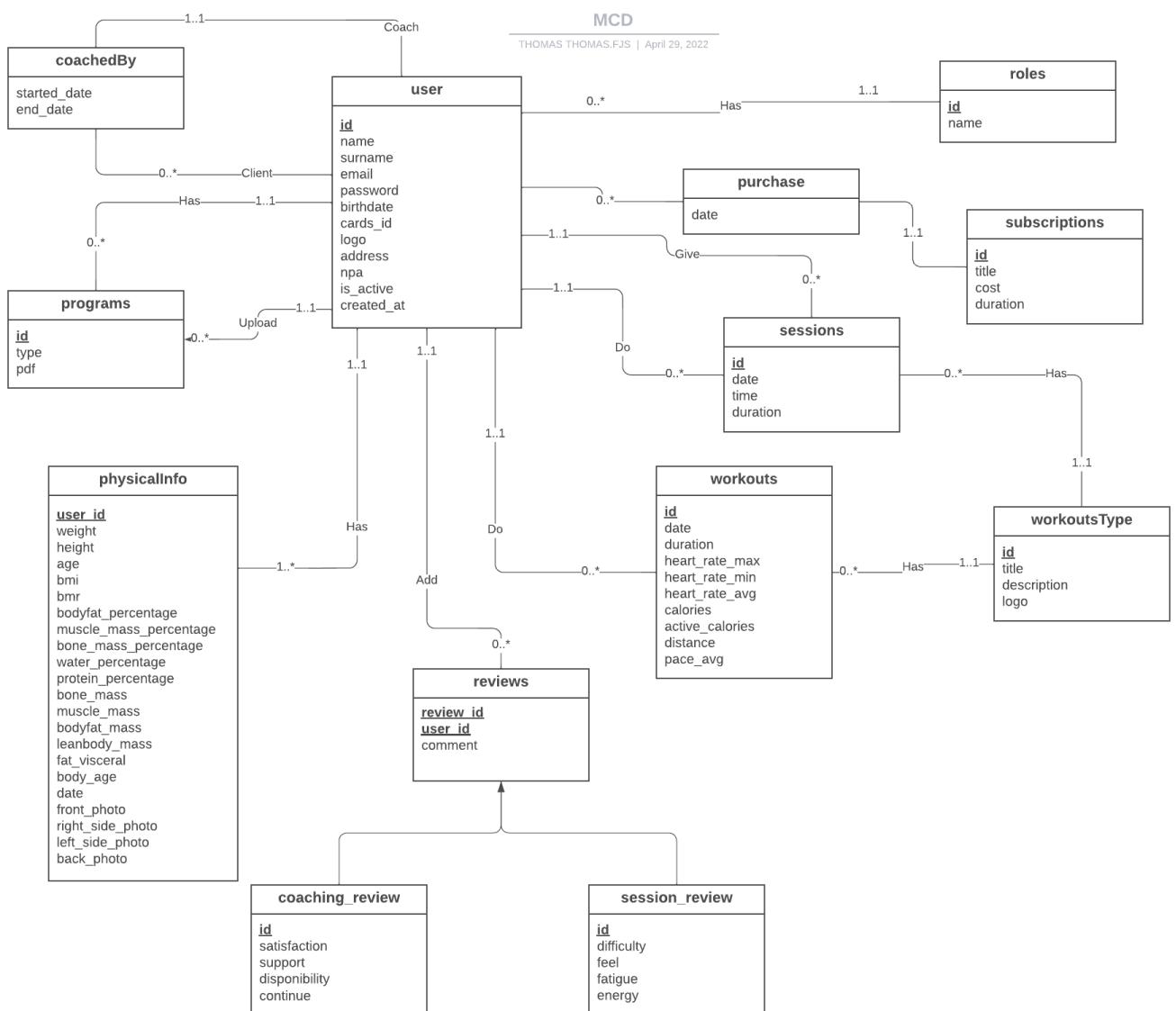
Maintenant que j'ai rajouté les bilans et les avis je vais revoir le MCD. Je décide de séparer les informations physiques de la table utilisateur et d'ajouter une date dans la table information physique pour avoir un historique de l'évolution. Il faut également que j'ajoute 2 tables pour les avis (Coaching et sessions).

Au niveau MCD, je pense que les 2 tables avis doivent être reliées à une table mère "Reviews" qui serait associée à la table Users. Je demanderai à M.Jossi lorsque je lui enverrai mon MCD pour qu'il puisse vérifier et me donner son avis.

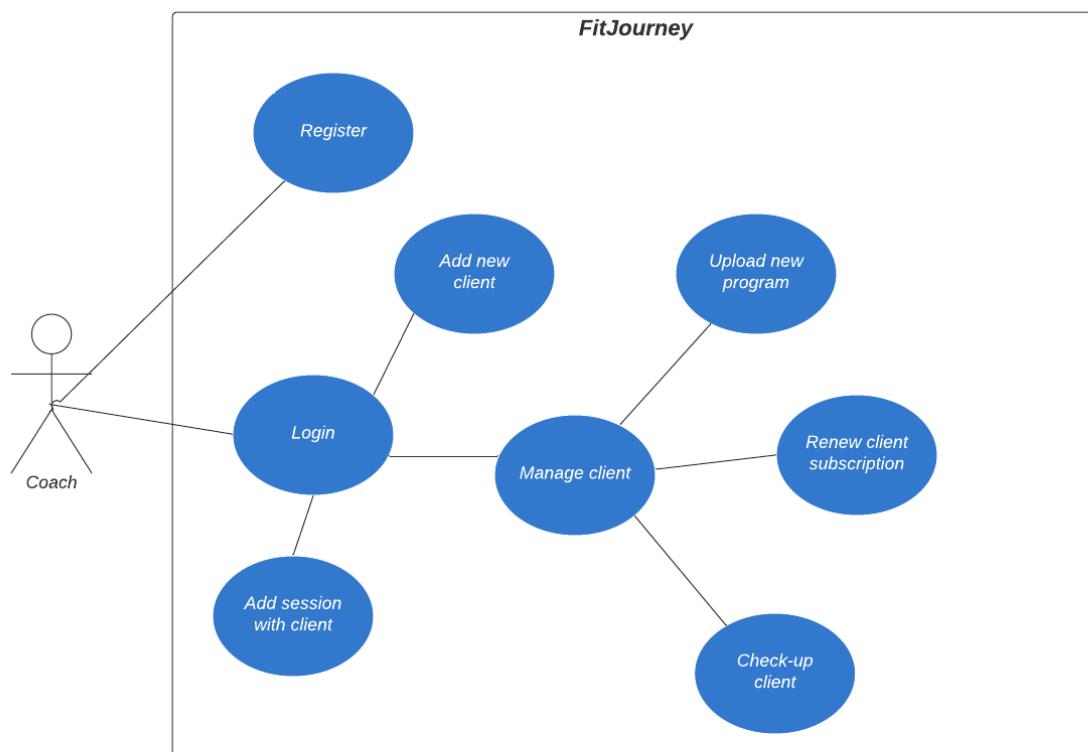
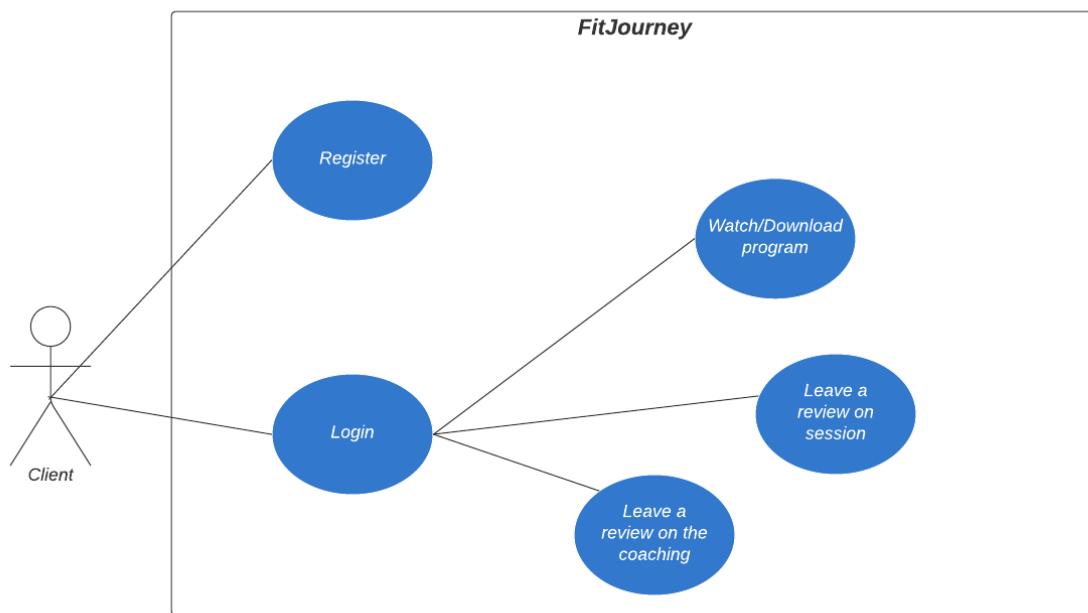
En utilisant une table mère pour les avis, je dois remonter le champ "commentaire" car c'est un champ commun entre les 2 tables avis. (Les champs d'évaluation pour les avis ne pourront donc pas être modifié sachant qu'ils représentent des champs définis dans la base de données)

Je vais encore revoir les cardinalités du MCD et je pourrai ensuite l'envoyer à M.Jossi.

Voici le nouveau MCD :



J'ai revu également les cas d'utilisations que j'avais réalisé, j'ai décidé de faire 2 schéma différent pour le client et le coach.



J'ai pu avancer sur la documentation, il faut encore que :

- J'ajoute les navbars dans la section maquette
- J'explique plus en détail l'arborescence de l'application
- J'ajoute l'explication de l'installation générale du projet
- J'explique les 2 sitemaps
- J'ajoute le MCD
- J'ajoute un schéma pour expliquer l'utilisation de l'API par l'application.

3.1.14 Lundi, 2 Mai 2022

Aujourd'hui je vais continuer d'avancer sur la documentation et je vais également reprendre le code maintenant que j'ai un MCD plutôt correcte. M.Jossi a pu regarder la nouvelle version du MCD que je lui avais envoyé. Je dois refaire quelques modifications pour respecter les standards Ecole (nom des entités en majuscule singulier) et mettre les verbe d'association à l'indicatif si possible.

M. Jossi est passé me voir, il m'a aidé à éclaircir le passage du MCD au MLD avec la table mère REVIEW. Nous avons décidé de ne pas prendre la table mère et de garder uniquement les tables filles car il n'y avait qu'un seul champ en commun.

J'ai du retoucher encore une fois mes maquettes pour revoir les navbars, j'ai également refait le sitemap du côté coach car j'avais oublié quelques chemins. J'ai rajouté la documentation sur les navbars de l'application.

J'ai implémenté le MCD avec SQLAlchemy pour créer les tables.

J'ai rajouté des routes pour afficher une page d'erreur en fonction de l'erreur qui survient :

- 403 pour un accès non autorisé
- 404 si la page n'est pas trouvée
- 500 si c'est un problème avec le serveur

J'ai également créé une route dynamique pour tous les templates dans le dossier templates/home, toutes les pages sont accessibles uniquement si l'utilisateur est connecté (@login_required)

Je rencontre un problème, je n'arrive pas à accéder aux pages car même une fois le login passé le login manager ne détecte pas la connexion (Il agit comme si l'utilisateur n'était pas connecté)

3.1.15 Mardi, 3 Mai 2022

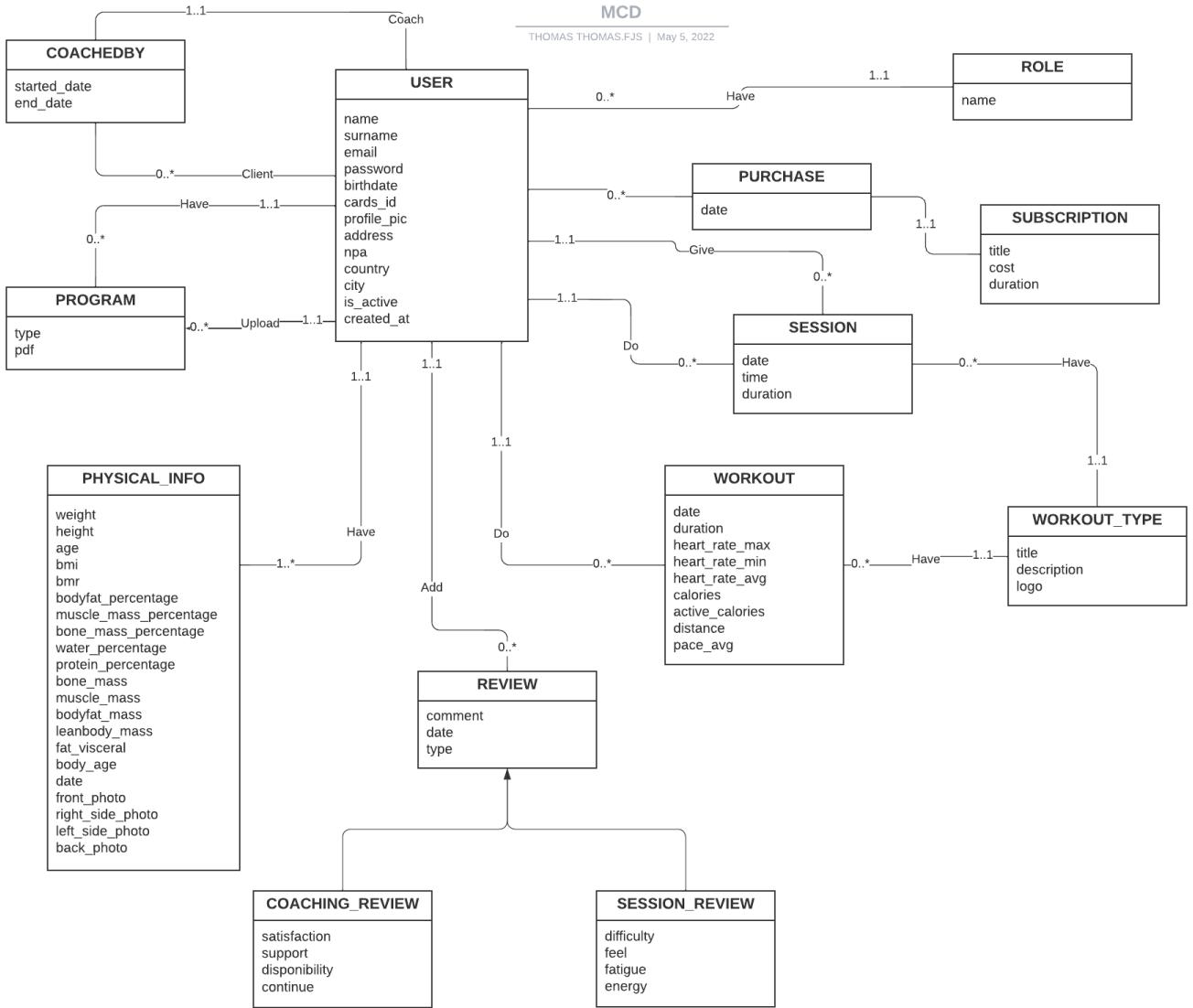
J'ai continué de regarder mon problème avec le login_manager hier soir, je n'ai toujours pas trouvé de solution.

Ce matin en relisant la documentation de Flask Login, j'ai essayé d'ajouter les méthodes que la classe User doit implémenter pour que le Login manager fonctionne. Cette fois le Login manager fonctionne avec les méthodes implémentées à la main, le problème est que pour éviter de dévoir les implémenter à la main j'avais fait hériter ma classe User d'UserMixin qui permet l'implémentation par défaut des propriétés et méthodes nécessaires.

Donc je ne sais pas pourquoi en faisant hériter ma classe de UserMixin cela ne fonctionne pas mais quand j'implémente les propriétés et méthode à la main le Login manager fonctionne.

[Documentation Flask Login](#)

J'ai très rapidement retouché le MCD pour enlever encore les id qui ne sont pas obligatoire et j'ai juste rajouter un champ City et Country pour la table USER.



J'ai commencé à avancer sur la page profil du client.

A la base j'avais séparé l'application en 2 gros dossiers :

- authentication/
- home/

home/ était sensé contenir toute l'application hormis l'authentification, je pense séparer ce dossier en 2 parties :

- client/
- coach/

Je vais alors avoir quelques fichiers en double mais ils seront bien précis par rapport à leurs utilités.

Chaque dossier doit contenir les fichiers : routes.py, forms.py, init.py qui seront bien sûr différents pour chaque dossier.

J'ai pu terminer la base de la page profil, les informations du client sont affichés et peuvent être modifiés. Il faut encore que je rajoute le "widget" pour les retours ainsi que l'option pour changer le mot de passe.

3.1.16 Mercredi, 4 Mai 2022

J'ai pu ajouter les requêtes SQL pour afficher les données sur la page profil comme la fin de l'abonnement qui a été souscrit en dernier.

Il faut que je commence le Poster car le rendu est fixé à Lundi et je dois également poursuivre l'avancement de ma documentation pour le rendu intermédiaire qui a lieu Mardi.

Les points que je dois encore ajouter :

- Expliquer plus en détail l'arborescence de l'application
- Ajouter l'explication de l'installation générale du projet
- Détailier les 2 sitemaps
- Remplir la section Base de données
- (Ajouter un schéma pour expliquer l'utilisation de l'API par l'application.)

Je peux éventuellement aussi commencer à documenter quelques fichiers comme les routes ou encore les blueprints que j'utilise.

J'ai pu avancer sur mon poster, M.Bonvin nous a proposé de passer demain pour voir les posters et donner un retour.

Pour revenir à l'application, je dois faire une requête SQL pour récupérer les retours du client. Le problème est que les retours sont dans 2 tables différentes. J'ai cherché pendant un petit moment pour trouver une solution, et je me suis rappelé des UNION en SQL qui me permettent exactement de faire ce dont j'ai besoin. Le seul problème est que les champs gardent le nom du premier SELECT.

Donc lorsque je récupère les retours client d'un client en particulier, je les récupère dans un objet CoachingReview même si les données proviennent de la table SessionReview (Car le premier SELECT de l'UNION est celui de CoachingReview). Il faut que je trouve comment rajouter des AS dans ma requête SQL.

3.1.17 Jeudi, 5 Mai 2022

J'ai trouvé le moyen de renommer les champs avec la méthode label(). J'ai donc renommé les champs du premier SELECT pour avoir des propriétés plus génériques. Je peux maintenant terminer l'affichage des retours client.

Finallement, je vais revoir la base de données pour optimiser la récupération des retours. Actuellement, il n'est déjà pas possible de savoir de quel type est le retour car pas de champs et pour savoir quel coach était assigné au client il faut vérifier dans 2 autres tables pour la date du coaching etc. Je pense finalement garder la table intermédiaire dans le MLD, une table "REVIEW" qui comporte les points communs entre les 2 (client_id, coach_id, date, comment, type) et garder les 2 tables fille qui comporteront les champs spécifiques à chacune. J'ai mis l'ID d'une review dans la table REVIEW et cet ID sera la clé primaire et étrangère des tables filles.

Une fois la table intermédiaire ajoutée, je pourrai terminer l'affichage des retours ainsi que la l'affichage de leurs détails.

Je suis bloqué sur un problème, je n'arrive pas à joindre une table récupérer à l'aide d'un UNION avec SQL Alchemy.

Voici la requête en SQL :

```
SELECT R.id, R.comment, R.date , R.type, I.Field1
FROM REVIEW AS R
JOIN (SELECT C.id AS ID, C.satisfaction AS Field1 FROM COACHING_REVIEW AS C UNION SELECT S.id, S.difficulty FROM SESSION_REVIEW AS S) AS I ON I.ID = R.id
WHERE id_client = 1
```

J'arrive à faire l'UNION avec tous les champs, seulement lorsque je veux join la table à une autre table je n'arrive pas à déterminer la jointure (le ON) car je ne peux pas accéder aux colonnes de la table "UNION".

La première requête qui va chercher toutes les reviews de coaching :

```
q1 = db.session.query(CoachingReview.id, CoachingReview.satisfaction.label("Field1"), CoachingReview.support.label("Field2"),
CoachingReview.disponibility.label("Field3"), CoachingReview.is_continuing.label("Field4"))
```

La deuxième requête qui va chercher toutes les reviews de sessions :

```
q2 = db.session.query(SessionReview.id, SessionReview.difficulty, SessionReview.feel, SessionReview.fatigue, SessionReview.energy)
```

L'union entre les deux :

```
q3 = q1.union(q2)
```

Et la requête qui relie les infos de la table mère REVIEW aux données récupérés dans l'UNION :

```
query = db.session.query(Review.id, Review.comment, Review.date, Review.type).join(q3, Review.id==q3.c.id).filter(Review.client_id==id)
```

La variable q3 qui contient le résultat de l'UNION est sensé posséder les colonnes de cette table accessible depuis q3.c mais j'ai l'erreur :

```
AttributeError : id
```

En debugant les objets SQL Alchemy, j'ai trouvé pourquoi je n'arrivais pas à accéder aux champs. Les champs avaient un alias généré automatiquement. Pour régler ce problème, j'ai rajouter un nouvel alias à chaque champs et désormais tout fonctionne.

Les reviews du clients sont maintenant affichés sur sa page profil, je peux maintenant ajouter la page de détails lorsqu'on clique sur la review.

Demain je vais beaucoup travailler sur la documentation car le rendu intermédiaire à lieu lundi et il faudrait que j'ai documenté toutes les fonctionnalités déjà effectué.

3.1.18 Vendredi, 6 Mai 2022

Je viens d'ajouter le changement d'image de profil. Sachant que l'application n'est pas destinée à contenir des milliers d'utilisateurs, j'ai créé un dossier "profile/" dans le dossier "img/" de l'application. Toutes les photos de profil seront enregistrées ici avec un UUID comme nom de fichier. Le nom de fichier est enregistré dans la base de données.

J'ai avancé un peu sur la page Workouts, il faut que je modifie la création des champs de la table 'WORKOUT' car les valeurs sont arrondies et n'ont pas de décimales.

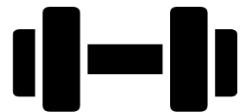
J'ai pu terminer l'affichage des workouts effectués par le client. Cet après-midi, je vais avancer sur la documentation technique et le poster.

Pour le moment, il me manque encore l'ajout des reviews par le client ainsi que l'affichage détaillé des reviews et des workouts et l'affichage des graphiques avec Chart.JS. Une fois cela terminé, j'aurai terminé le côté client.

La partie des graphiques avec Chart.JS sera je pense que la partie la plus "compliquée". Pour le reste de l'application, tous le côté coach est uniquement composé de formulaire et des mêmes pages que le côté client mais avec des options supplémentaires.

J'ai pris la décision de laisser un peu de côté l'aspect "esthétique" de l'application car n'étant vraiment en avance je veux me concentrer principalement sur les fonctionnalités de l'application.

J'ai pu avancé un peu sur la documentation de la base de données. Je vais devoir revoir mon poster ce week-end, M.Bonvin m'a fait un retour : Je n'explique pas ce que mon application fait et je fais plus de pub pour Polar qu'autre chose.



FITJOURNEY

POLAR®
WATCHES

START TODAY!

**TAKE YOUR COACHING TO
THE NEXT LEVEL**

