Projet PDA

Thomas Fujise

Thomas Fujise

CFPT 2022

Table des matières

1. Documentation technique

1.1 Rappel du cahier des charges

1.1.1 Introduction

Dans le cadre du cours PDA, nous devions effectuer un projet pendant 9 semaines pour la préparation à notre travail de diplôme. Il nous était demandé d'effectuer une application compatible mobile et d'utiliser un framework à choix.

But du projet

Le but de mon projet est de réaliser une application WEB avec Python Flask. L'application doit permettre l'affichage de graphique avec la librairie Chart.JS.

Contraintes techniques

Pour ce projet, nous devions utiliser un framework de notre choix. L'application réaliser doit également être compatible mobile.

1.1.2 Analyse

Technologies

- Python
- Python Flask
- Chart.js
- Mkdocs Markdown(Documentation)
- Github (versionning, task manager)
- Trello
- Ordinateur de type PC, 2 écrans
- Visual Studio Code

REACT JS

React est une bibliothèque JavaScript développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état. J'avais fait le choix de partir sur React à la base mais je suis vite revenu sur ma décision pour m'orienter sur Python Flask.

PYTHON FLASK

Flask est un micro-framework python facile et simple d'utilisation qui permet de faire des applications web évolutives. Flask dépend de la boite à outils WSGI de Werkzeug et du moteur de templates Jinja. J'ai fait le choix d'utiliser Python Flask pour mon application car pour mon travail de diplôme j'utilise beaucoup de Python et je trouvais cohérent d'utiliser ce framework Python qui permet de réaliser une application WEB.

Avantages

Flask a été conçu pour être un micro-framework simple et léger. Son but premier est de permettre de démarrer le développement d'une application web sur une base solide sur laquelle appuyer le reste des phases de développement. A partir de cette base, il est possible de construire son projet bloc à bloc, selon les besoins.

Pour le développeur, Flask est synonyme de totale liberté. Il maîtrise en effet parfaitement son code puisqu'il l'écrit lui-même et n'a pas à plonger dans la compréhension de fonctionnalités préfabriquées comme ce peut être le cas avec des frameworks plus complets.

Inconvenient

Étant un micro framework, Flask ne propose pas beaucoup de fonctionnalités de haut niveau. Il faut également comprendre le système de routes utilisé avec Flask.

Installation

Pour installer Flask il suffit d'entrer la commande :

```
pip install Flask
```

MICRO FRAMEWORK

Un micro framework est un framework qui tente de fournir uniquement les composants absoluments 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.

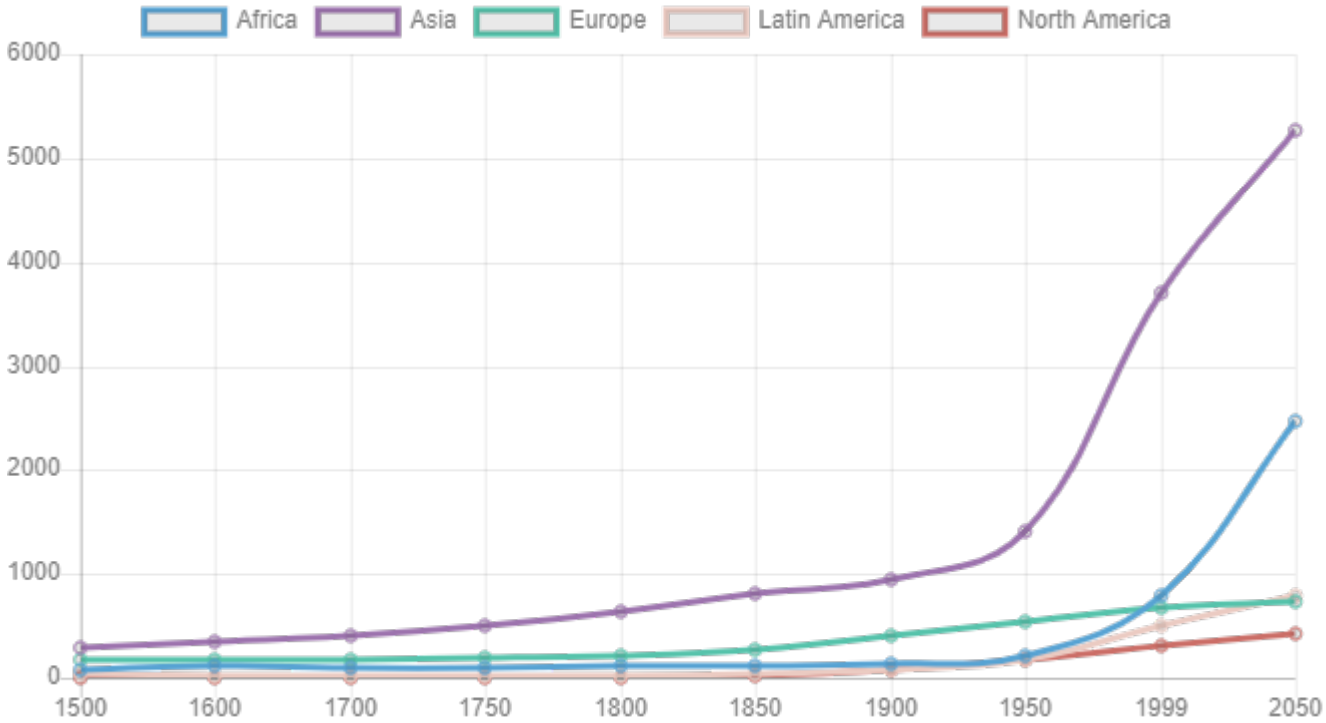
En définissant uniquement le moteur de templates et un système de routes, Flask laisse le choix de personnaliser (en ajoutant des packages) pour la gestion des formulaires par exemple.

CHART.JS

Chart.js est une librairie open source, elle permet la visualisation de données en utilisant JavaScript. Elle est similaire à Chartist ou Google Chart. Chart.js supporte 8 différents type de graphique et sont tous responsive. Pour pouvoir utiliser Chart.js il faut :

- Definir ou dessiner le graphique sur notre page
- Definir quel type de graphique afficher
- Definir les données, labels et toutes les autres options

Voici un exemple de graphique que peut générer chart.js :



SQL ALCHEMY

SQLAlchemy est un toolkit open source SQL et un mapping objet-relationnel écrit en Python. Il utilise le pattern Data Mapper au lieu de l'active record utilisés par de nombreux autres ORM.

Installation

Pour installer SQLAlchemy il faut exécuter la commande :

```
pip install sqlalchemy
```

Il est généralement recommandé d'utiliser une bibliothèque supplémentaire :

```
pip install flask-sqlalchemy
```

Exemple d'utilisation

Dans mon application, SQLAlchemy est utilisé pour créer et gérer toutes les tables de la base de données. Exemple de modèle (pour la table utilisateur):

```python
class Users(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False, unique=True)
    name = db.Column(db.String(200), nullable=False)
    email = db.Column(db.String(120), nullable=False, unique=True)
    favorite_color = db.Column(db.String(120))
    about_author = db.Column(db.Text(), nullable=True)
    date_added = db.Column(db.DateTime, default=datetime.utcnow)
    profile_pic = db.Column(db.String(), nullable=True)

    # Do some password stuff!
    password_hash = db.Column(db.String(128))
    # User Can Have Many TrainingData
    training_data = db.relationship('TrainingData', backref='client')
```

TRELLO

Trello est un outil de gestion de projet en ligne, lancé en septembre 2011 et inspiré par la méthode Kanban. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. J'ai choisi d'utiliser Trello car c'est le seul outil de gestion de projet que l'on a pu survoler durant le cours et je ne voulais pas m'attarder à apprendre l'utilisation d'un autre outil.

Environnement de développement

Pour l'environnement de développement, j'ai utilisé Visual Studio Code

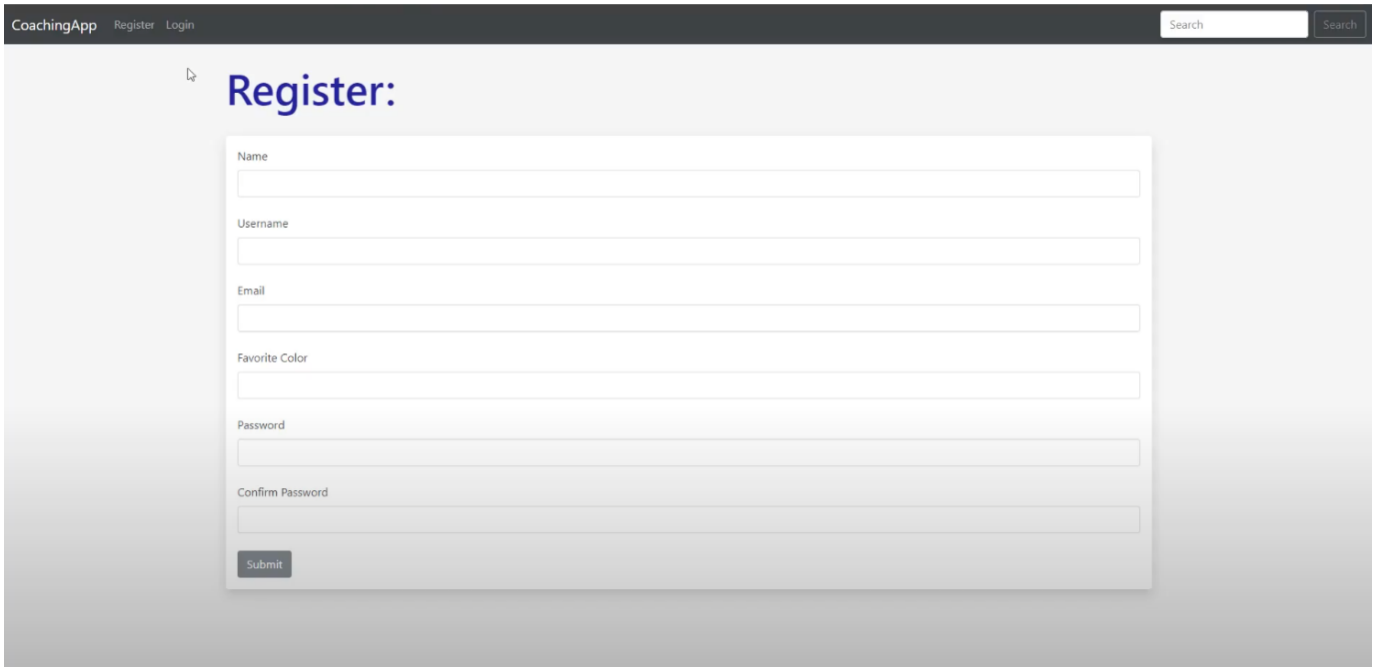
1.2 Analyse fonctionnelle

1.2.1 Interface

L'interface réalisée pour cette application est très basique, j'ai utilisé Bootstrap 5.

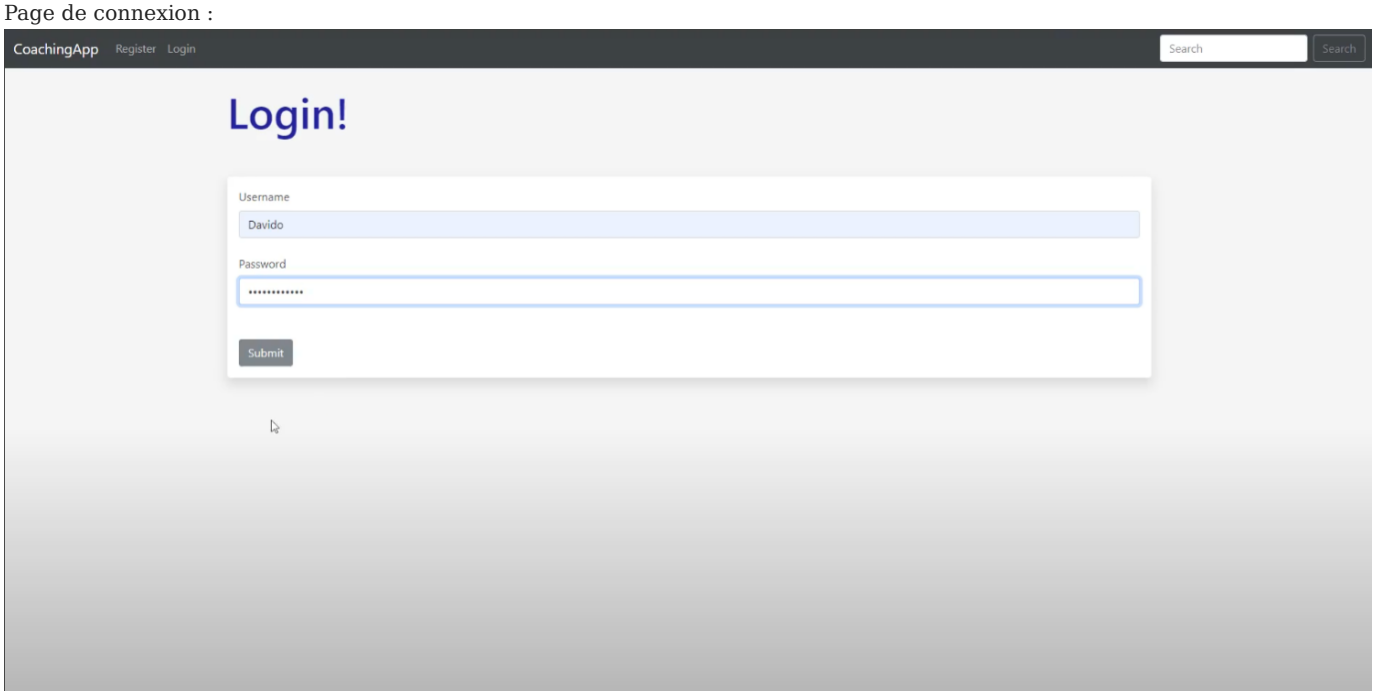
L'application possède, pour le moment, uniquement 4 pages :

Page d'inscription :



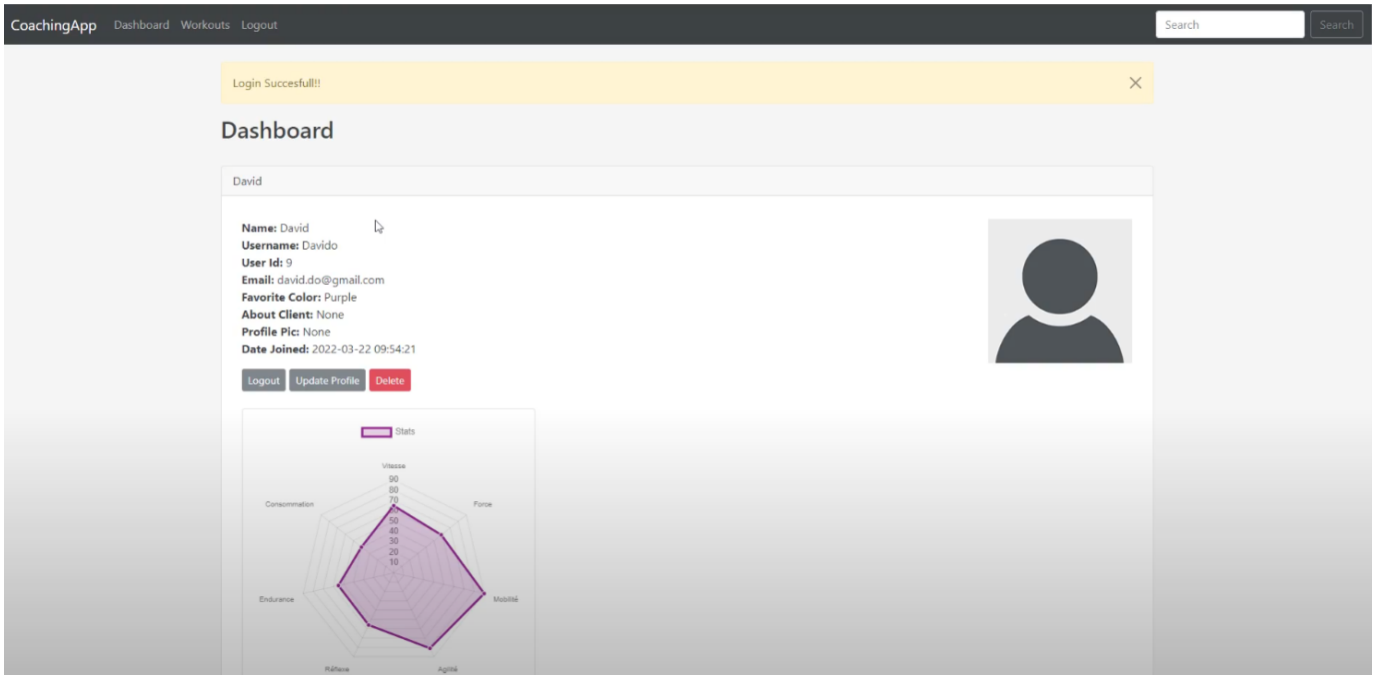
La page comporte un simple formulaire d'inscription.

Page de connexion :



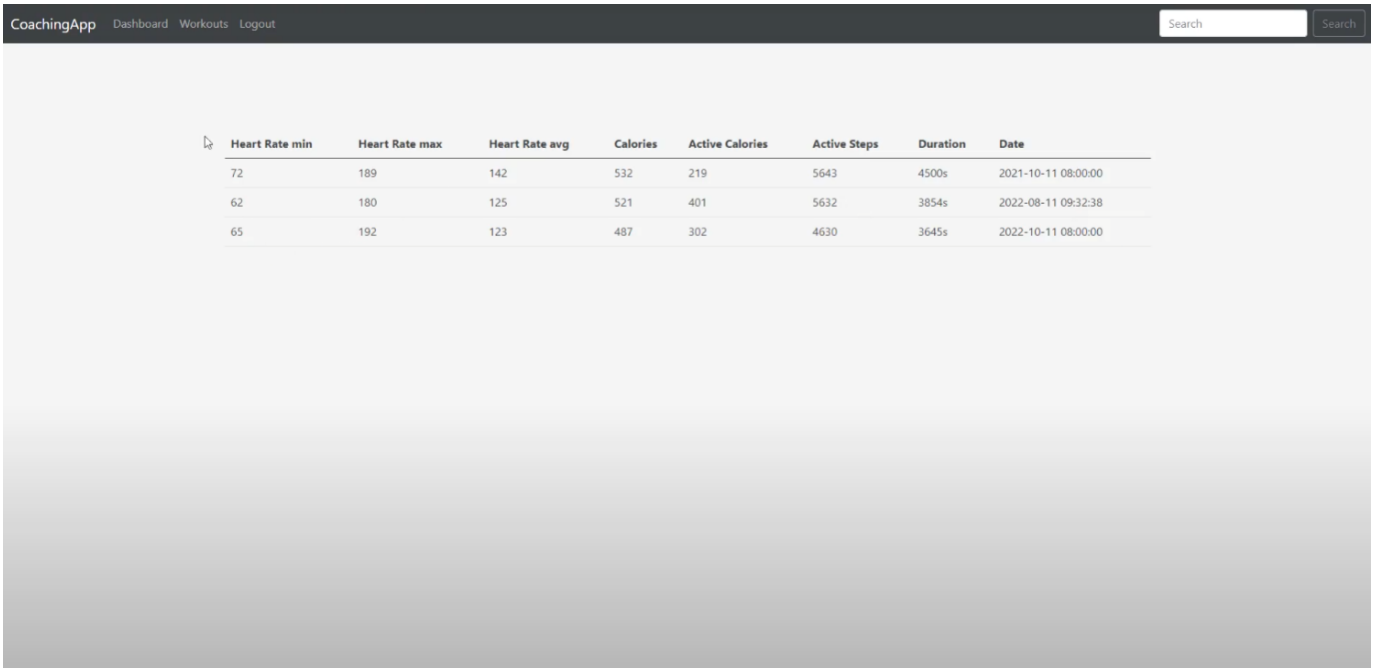
La page comporte un simple formulaire de connexion

Page tableau de bord :



La page comporte un tableau de bord avec toutes les informations du compte connecté. Un graphique est affiché avec Chart.js affichant des données sauvegardé en base. La photo de profil est affichée sur la droite (Si aucune photo n'a été importée par l'utilisateur, une photo de base est utilisée). Un formulaire de mise à jour est disponible en bas de la page pour mettre à jour son profil :

Page workouts :



La page comporte un tableau qui affiche les données des entrainements effectués par l'utilisateur.

1.2.2 Fonctionnalités

Login/Register

L'application permet de se connecter et si l'utilisateur ne possède pas encore de compte, d'en créer un. Le formulaire d'inscription comporte plusieurs champs :

- Nom
- Pseudo
- Email
- Couleur préférée (Utilisée pour la couleur des graphiques)
- Mot de passe

Pour la connexion uniquement l'email et le mot de passe sont demandés.

Dashboard

L'application propose un dashboard pour visionner son profil. Toutes les informations du compte sont affichés sur cette page. Un graphique généré par Chart.js affiche les données de l'utilisateur. La photo de profil est affichée (si l'utilisateur n'en a pas importé une photo de base est utilisée).

Update

L'application propose un formulaire de mise à jour qui permet de mettre à jour les informations de notre profil. Les informations suivantes sont modifiables :

- Nom
- Pseudo
- Email
- Couleur préférée
- Description
- Photo de profil

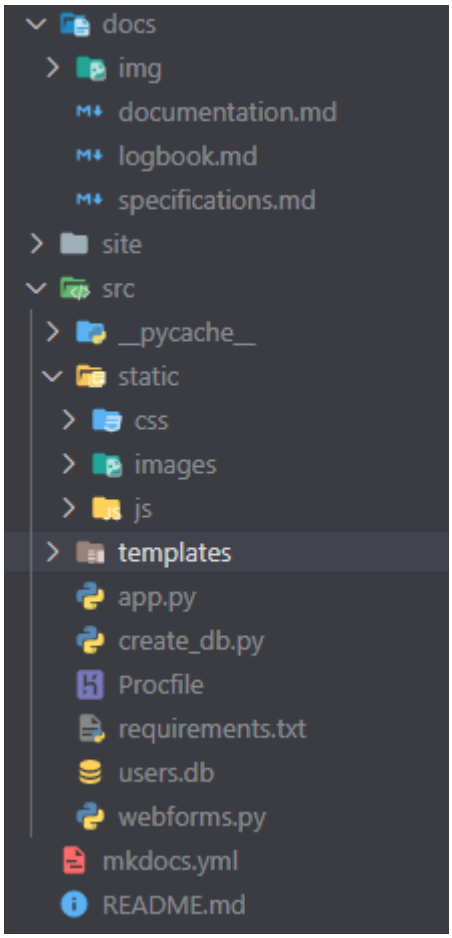
Logout

L'application possède un bouton de déconnexion.

1.3 Analyse organique

1.3.1 Architecture

Arborescence de fichier



static/css/

Dossier contenant les fichiers CSS pour l'application.

static/images/

Dossier contenant les images de l'application.

static/js/

Dossier contenant les fichiers JavaScript de l'application.

templates/

Dossier contenant toutes les templates HTML utilisées dans l'application. Toutes les templates sont des extensions de fichier base.html qui constitut la base de chaque page :

```
<!Doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1"
crossorigin="anonymous">
    <!-- Our CSS -->
    <link href="{{ url_for('static', filename='css/style.css') }}" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.7.1/chart.min.js"
integrity="sha512-
QSkVNOCYLtj73J4hbmVoOV6KVZuMluZlioC+trLpewV8qMjsWqlIQvkn1KGX2StWvPMdWGBqim1xlC8krl1EKQ=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>

    <title>Coaching App</title>
  </head>
  <body>
    {% include 'navbar.html' %}
    <br/>
    <div class="container">
        {% block content %}
        {% endblock %}
    </div>
    <!-- Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW"
crossorigin="anonymous"></script>


  </body>
</html>
```

app.py

Fichier Python contenant toutes les routes de l'applications ainsi que les modèles SQLAlchemy utilisés pour la base de données.
Exemple de route :

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = Users.query.filter_by(username=form.username.data).first()
        if user:
            # Check the hash
            if check_password_hash(user.password_hash, form.password.data):
                login_user(user)
                flash("Login Succesfull!!")
                return redirect(url_for('dashboard'))
            else:
                flash("Wrong Password - Try Again!")
        else:
            flash("That User Doesn't Exist! Try Again...")


    return render_template('login.html', form=form)
```

Lors du login on vérifie si l'utilisateur existe, si oui on vérifie que le hash du mot de passe saisit soit le même que celui enregistré en base. Si ce n'est pas le cas un message d'erreur est affiché.

create_db.py

Fichier contenant un script Python qui permet de créer la base de données à l'aide de MySQLConnector.

```python
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="dbflask_dba",
    passwd = "Super2012",
    )

my_cursor = mydb.cursor()

my_cursor.execute("CREATE DATABASE our_users")
```

webforms.py

Fichier Python contenant tous les formulaires de l'application. Exemple de formulaire (formulaire de connexion) :

```python
class LoginForm(FlaskForm):
    username = StringField("Username", validators=[DataRequired()])
    password = PasswordField("Password", validators=[DataRequired()])
    submit = SubmitField("Submit")
```

requirements.txt

Fichier Texte contenant toutes les librairies/modules nécessaires pour le fonctionnement de l'application. Pour pouvoir tout installer il suffit d'exécuter la commande :

```
pip install -r requirements.txt
```

mkdocs.yml

Fichier YAML contenant la configuration pour Mkdocs.

documentation.md

Fichier Markdown contenant la documentation technique. Un pdf est généré à partir de ce fichier avec Mkdocs.

logbook.md

Fichier Markdown contenant un détail de l'activité journalière pendant le projet. Un pdf est généré à partir de ce fichier avec Mkdocs.

specification.md

Fichier Markdown contenant le cahier des charges qui explique toutes les attentes par rapport au projet.

1.4 Conclusion

1.4.1 Problèmes rencontrés

Le principal problème rencontré a sûrement été le faux départ avec React au départ. J'ai perdu pas mal de temps à essayer d'apprendre comment bien utiliser React. J'ai également perdu 1 semaine de travail après avoir attrapé le Covid-19. Une fois passé sur Python Flask, je n'ai pas vraiment rencontré de problème.

1.4.2 Améliorations possibles

Il faudrait améliorer l'interface (peut-être utiliser un autre framework que Bootstrap). J'aimerais également pouvoir rajouter un formulaire d'importation pour fichier Excel. Cela représenterait l'upload des programmes par le coach. Il faudrait ensuite afficher les données dans le fichier excel.

1.4.3 Bilan personnel

J'ai vraiment apprécié de travailler sur ce projet car j'ai pu apprendre comme il le fallait comment utiliser Python Flask et je compte bien l'utiliser pour mon travail de diplôme. J'ai également pu implémenter les graphiques avec la librairie Chart.js qui est également un gros point de mon travail de diplôme. Je regrette de ne pas avoir pu aller un peu plus loin et rajouter encore quelques fonctionnalités, comme la partie *coach qui n'a pas été implémenté mais dans l'ensemble, je suis quand même satisfait d'avoir pu finir avec une application fonctionnelle avec un framework que je ne connaissais pas à la base.

2. Logbook

2.1 21 Décembre 2021

Création du cahier des charges, du dépôt GIT ainsi que le Trello pour le projet.

2.2 11 Janvier 2022

Révision du cahier des charges, je choisis d'utiliser react pour le projet même si je n'ai pas énormement d'expérience avec. Je commence a créer quelques composants pour mes premières vues.

2.3 18 Janvier 2022

Je vois qu'avec npm je peux directement intégrer Chart.Js dans mon projet avec la commande :

```
npm i chart.js
```

Je continue mes composants pour créer une vue représentant le tableau de bord principal

2.4 25 Janvier 2022

J'ai regardé quelques sample d'utilisation de Chart.JS et essaye de l'implémenter dans mon projet pour afficher des graphiques.

2.5 01 Février 2022

J'ai pu ajouter un composant sur mon tableau de bord avec un graphique (Chart.JS), problème avec l'échelles sur le graphiques (Impossible de réduire l'échelles à .5). J'avance sur les autres vues (Login/Register). J'ai également eu le temps de créer ma base de données pour pouvoir stocker les données utilisateurs.

2.6 08 Février 2022

En travaillant sur mon travail de semestre, je me suis rendu compte que Python Flask serait vraiment très utile pour mon travail de diplôme et je pense de plus en plus à l'utiliser. Je pense donc changer de Framework et recommencer avec Flask, sachant que j'ai passé une grande partie du temps à me documenter sur Vue je peux rattraper assez vite en utilisant Flask. Je créer une page d'accueil avec Flask et je commence directement le Login/Register.

2.7 22 Février 2022

J'avance sur mon application de base avec Python Flask. J'ai regardé quelques tutos sur youtube pour en apprendre un peu plus sur le fonctionnement de Python Flask. J'ajoute une base de données pour enregistrer quelques infos sur les utilisateurs. Je découvre la librairies python SQL Alchemy.

2.8 1 Mars 2022

Absent (Covid-19)

2.9 8 Mars 2022

J'ai pu avancé sur mon application Flask, j'ai maintenant un Login/Register fonctionnel et une page "dashboard" qui me permet de modifier les infos de l'utilisateurs. J'utilise SQL Alchemy pour relier la base de données, SQLite pour le moment car la connexion était plus simple. Il faut que je regarde comment faire la connexion pour une base MySQL (Pas la même que pour SQLite).

2.10 15 Mars 2022

J'ai réussi à faire la transition sur MySQL. J'ai du changé l'URI dans la config SQLAlchemy. J'ai également du installer un module en plus **pymysql** qui me permet d'établir la connexion (je le rajoute dans le requirements.txt)

Commande :

```
pip install pymysql
```

Je découvre pipreqs qui permets de générer le fichier requirements.txt avec tous les packages et leurs versions.

3. Cahier des charges

3.1 Sujet

Créer une application WEB mobile permettant d'afficher des données d'entrainement avec des graphiques

3.2 But du projet

Le but du projet est de créer une plateforme WEB qui permet de gérer plusieurs données d'entrainements et avoir un aperçu sur la progression. Les données suivantes sont prises en comptes :

- Pulsation cardiaque (Repos/Actif/Après effort)
- Le type d'exercice effectué
- Date et durée de l'entrainement
- Nombre de total de calories brulées
- Nombre de calories active

Les programmes d'entrainements et de nutrition peuvent être importer par le coach et visionné par les praticants directement depuis l'application.

3.3 Utilisation

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

En tant que praticant : On a accès à nos données d'entrainements et au différents programme que le coach à publié pour nous. Un système de notification est mis en place pour avertir lorsqu'un nouveau programme/bilan est disponible

En tant que coach: On peut accéder aux données de tous nos praticants avec les courbes de progression. On peut importer les différents programmes des praticants directement depuis l'application. Un système de facturation est mis en place pour les praticants.

3.4 Spécifications

- Les données d'entrainements sont récupérées depuis une base de données
- Les graphiques liés aux stats sont effectués avec la librairie javascript Chart.js
- Le tableau de bord permettant d'afficher et de gérer toutes les données est réalisé avec ReactJS
- Les programmes d'entrainements et de nutritions doivent respecter le format proposé (Lors de l'upload une vérification est effectuée)

3.5 Environnement

Technologies utilisées :

- Python Flask
- HTML
- CSS
- Javascript
- SQL
- Chart.js

Système d'exploitation :

- Développement sur Windows

Versionning / Documentation :

- GitHub
- MarkDown (Mkdocs)

3.6 Livrable

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