# Runscape Web App

*Philibert, Alexandre*
*Rue du temple 8*
*1148, Cuarnens*

SI-C4b

5 juin 2019

Table des matières

# 1 Analyse préliminaire

## 1.1 Introduction

Le projet consiste à enrichir fonctionnellement un prototype existant par la création d'une interface graphique permettant de dialoguer avec un back-end existant. Ce projet fait suite à la création d'une API permettant la création, lecture, modification et suppression d'utilisateurs, de types d'activités et d'activités sportives. L'API intègre également un mécanisme de création de token d'authentification. J'ai choisi d'entreprendre ce projet car je souhaiterai ajouter un composant graphique au travail que j'ai déjà pu accomplir.

Les fonctionnalités existantes sur l'application étant reprises pour la réalisation de ce projet sont :

- Standardisation du retour d'erreur de l'API.
- Authentification par session et json web token.
- Importation d'une activité.
- Profil administrateur.
- Un effort particulier a été produit pour obtenir une architecture cohérente et faciliter les futures évolutions.

## 1.2 Objectifs

*Les objectifs sont repris du cahier des charges.*

### 1.2.1 Création d'activité - sans gpx (Story 001)

En tant que sportif (membre), je veux créer des activités, afin de suivre ma progression.

**Test d'acceptations**

Contexte

En tant que membre (sportif), je navigue jusqu'à la page me permettant de créer une nouvelle activité. Je ne dispose pas de gpx[1].

Événement

Je crée l'activité en saisissant les attributs suivants :

- L'heure de départ
- Le type d'activités (vélo, natation, course à pied, marche)
- La durée de l'effort
- Distance parcourue

[1] Le GPX est un format de fichier qui permet d'enregistrer des positions GPS ainsi que des horodatages.

- Lieu et pays de réalisation de l'activité
- Les dénivelés positifs et négatifs
- La vitesse moyenne

Critères de réussite

Le système enregistre chacune des dimensions mentionnées ci-dessus et affiche l'activité.

### 1.2.2 Création d'activité - avec gpx (Story 002)

En tant que sportif (membre), je veux créer des activités, afin de pouvoir suivre ma progression.

**Test d'acceptations**

Contexte

Je navigue jusqu'à la page me permettant de créer une nouvelle activité. Je dispose d'un gpx.

Événement

Je crée l'activité en important le gpx et saisissant manuellement les données qui ne sont pas présente ou interprétable depuis le gpx.

- Tous les attributs présents dans la story 001 (sans gpx) doivent être traités.
- Le parcours.

Critères de réussite

Le système enregistre chacune des dimensions mentionnées ci-dessus et affiche l'activité.

Note : la représentation graphique du parcours fait partie d'une story séparée.

### 1.2.3 Interprétation du parcours (Story 003)

En tant que sportif, je veux pouvoir observer mon entraînement sur une carte, afin d'analyser plus précisément mes performances.

**Test d'acceptations**

Contexte

Je dispose d'au moins d'une activité qui a été créer à l'aide d'un gpx. Je navigue jusqu'à la page me permettant d'afficher l'activité de manière graphique.

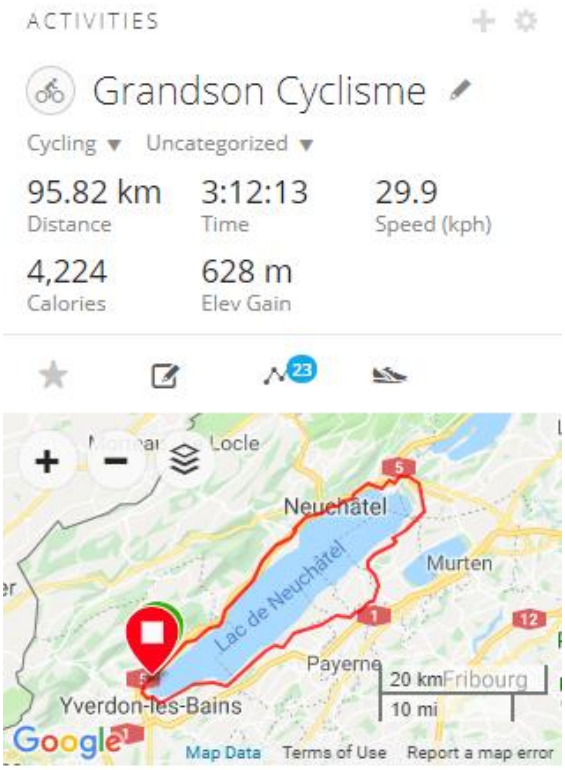
Evénement

Je demande à afficher mon parcours de manière graphique.

Critères de réussite

Le système affiche le parcours. Une carte (topologique, satellite) est présente avec une représentation du parcours.

Soit en déplaçant le curseur sur le parcours, soit un tableau doivent me permettre de connaître le "pace" (moyenne au kilomètre) pour chaque kilomètre.




Exemple d'IHM : Source garmin connect

### 1.3 Planification initiale

Est disponible en annexe (PHILIBERT_PlanificationInitiale).

# 2 Analyse

## 2.1 Analyse concurrentielle

L'analyse concurrentielle compare des solutions gratuites, cela peut entraîner des limitations aux solutions analysées. L'analyse est, par conséquence, orientée sur les fonctionnalités des applications plutôt que l'esthétique

- VisuGPX: https://www.visugpx.com/
- uTrack: http://utrack.crempa.net/
- trackreport: https://www.trackreport.net/

### 2.1.1 uTrack

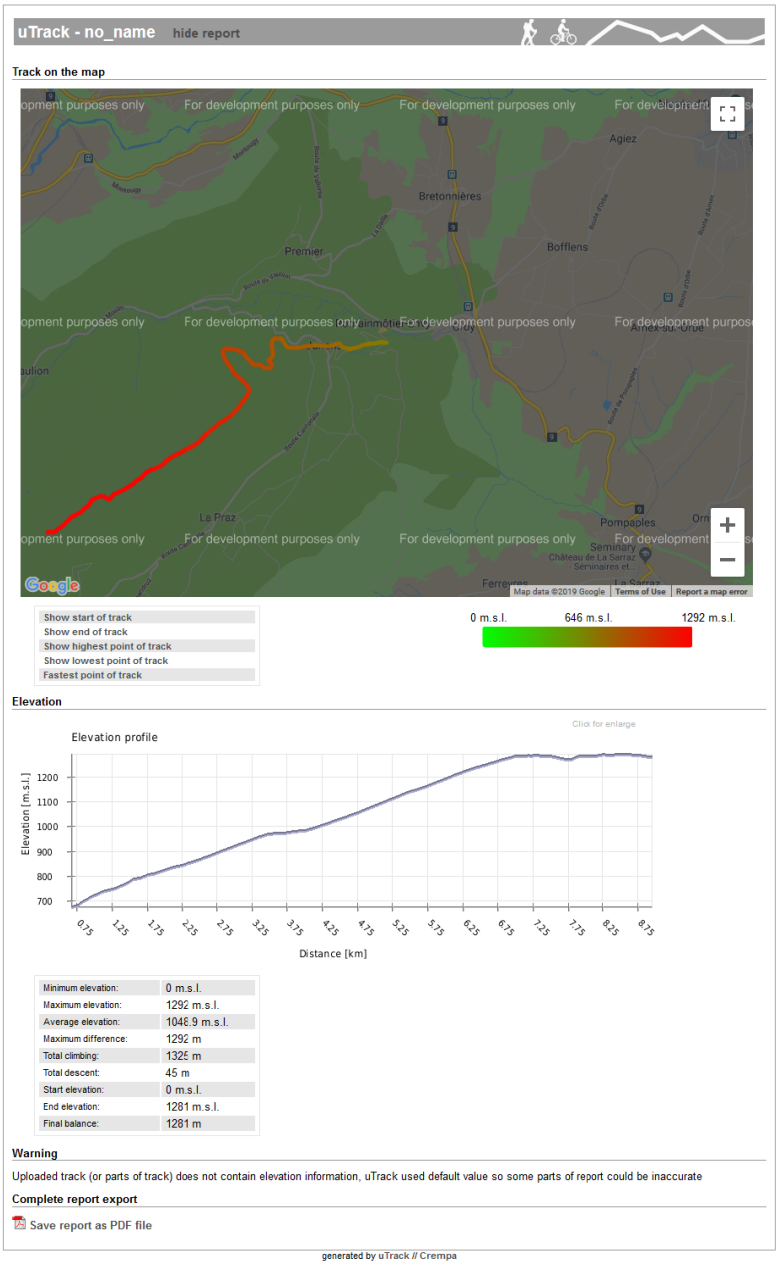
**Points positifs**

- La carte est interactive

**Contraintes et limites**

- La taille maximale des fichiers GPX pouvant être envoyés est de 1Mb
- Les graphiques sont représentés en tant qu'images, cela limite l'interaction que peut avoir un utilisateur avec ceux-ci.

**Interface graphique**




La page propose des graphiques simple et les données de l'activité sont également affichées dans un tableau. Le dénivelé du parcours est affiché avec un gradient de couleur.

### 2.1.2 trackreport

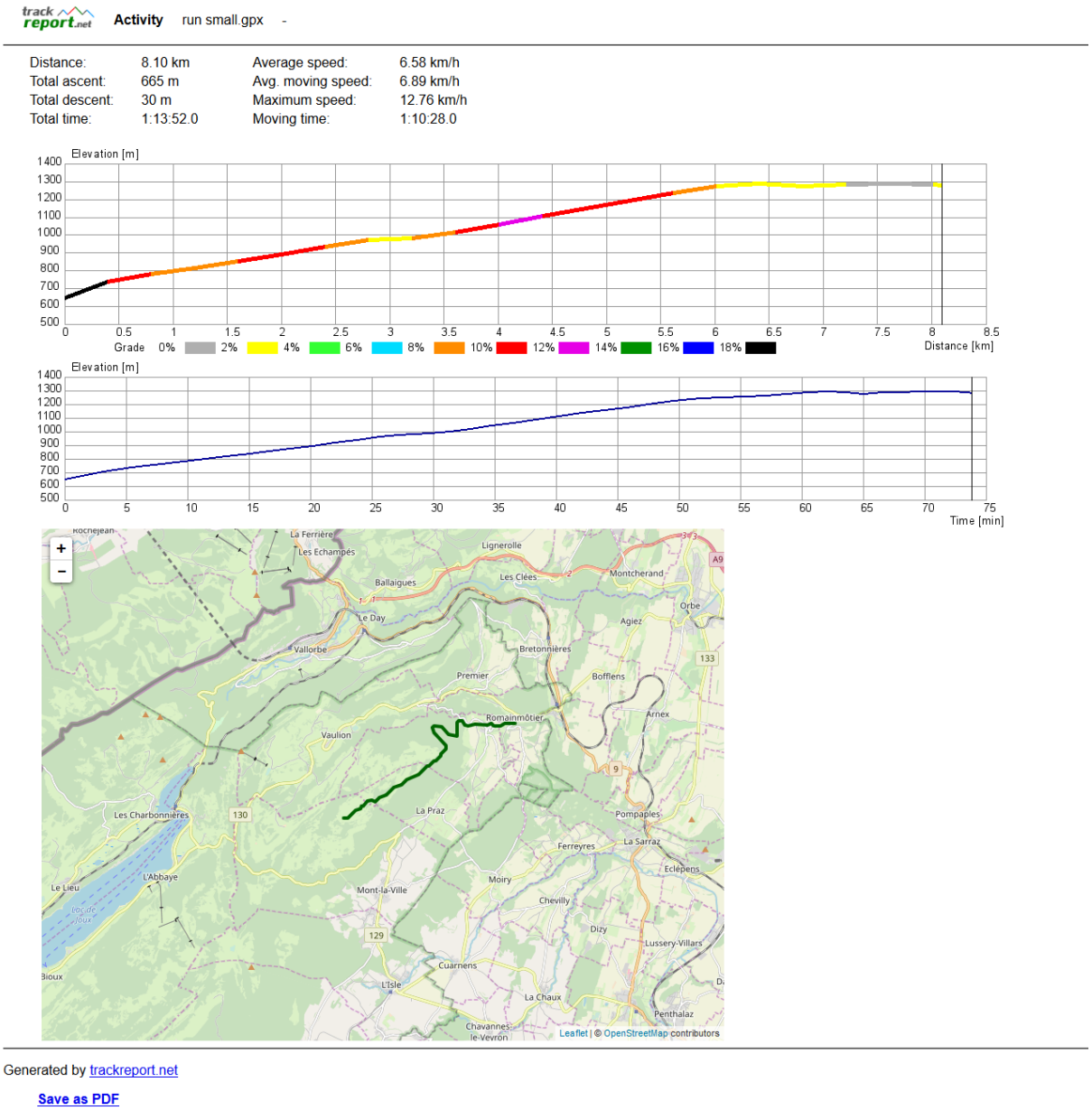
#### Points positifs

- La carte est interactive

#### Contraintes et limites

- La taille maximale des GPX pouvant être envoyés sont est de 2,5 Mb
- Les graphiques sont représentés en tant qu'images, cela limite l'interaction que peut avoir un utilisateur avec ceux-ci.

#### Interface graphique



### 2.1.3 VisuGPX

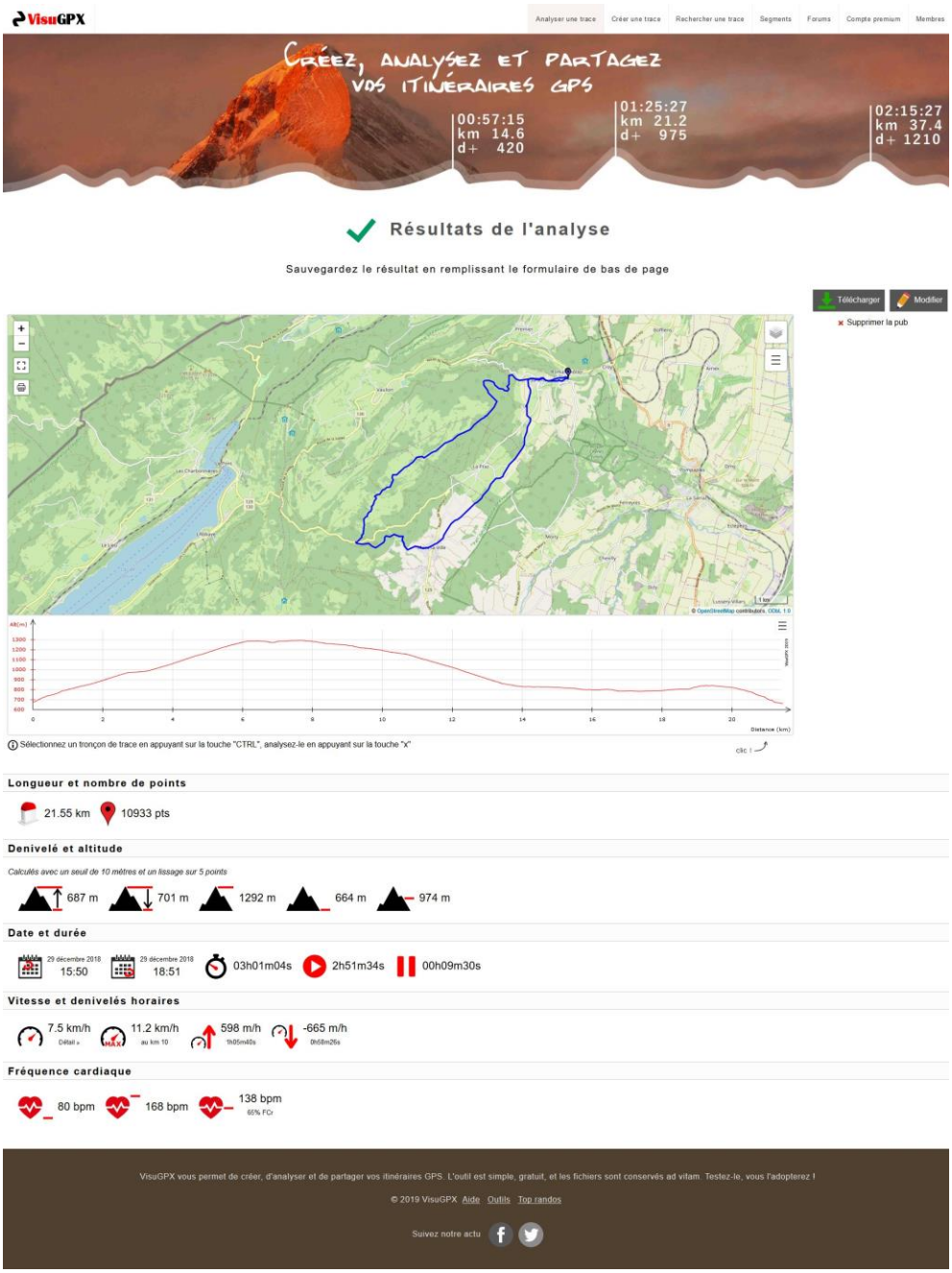
**Points positifs**

- La carte et les graphiques sont interactifs

**Contraintes et limites**

- La taille maximale des GPX pouvant être envoyés sont est de 2,5 Mb
- Les graphiques sont représentés en tant qu'images, cela limite l'interaction que peut avoir un utilisateur avec ceux-ci.

**Interface graphique**



## 2.2 Création d'activités

Le cahier des charges demande l'intégration de la création d'activités sportives. Le cahier des charges mentionne également la manipulation d'activités sportives sans et avec un fichier GPX. L'application doit gérer le lieu et le pays de réalisation des activités, ces champs nécessitent d'être ajouté à l'application.

Le sportif disposera d'une interface se présentant sous forme d'un client web lui permettant de créer des activités sportives. Pour faciliter la création d'une activité par un sportif, celui-ci aura la possibilité d'envoyer le fichier GPX entier à l'aide de l'interface graphique. Ce fichier pouvant s'avérer volumineux, des modifications doivent être apportées à l'application pour qu'elle puisse gérer le téléchargement des fichiers.

## 2.3 Visualisation des activités

Le sportif doit disposer d'un moyen d'afficher son entrainement sur une carte. Pour ce faire, une page regroupant toutes les activités d'un sportif peut s'avérer pratique. L'utilisateur disposera d'une interface centralisée où il pourra très simplement naviguer entre les activités.

## 2.4 Visualisation d'une activité sous forme de carte

Le cahier des charges demande qu'un sportif puisse observer son entraînement sur une carte. Des statistiques globales de l'activité seront ajoutées sur la page permettant de visualiser une activité pour permettre au sportif une meilleure analyse de son activité.

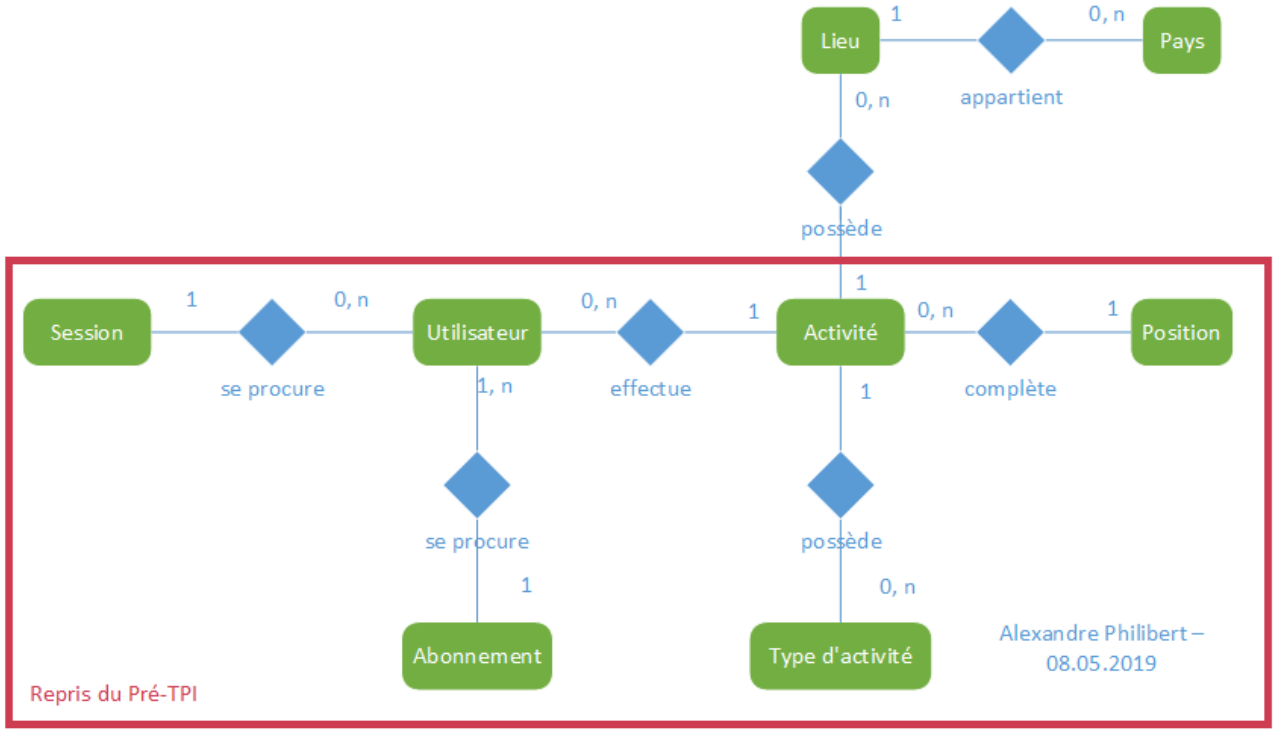
## 2.5 Gestions de plusieurs requêtes simultanées

L'ajout d'une fonctionnalité permettant de gérer plusieurs requêtes simultanément sur l'application permettrait de fluidifier l'expérience d'un sportif lorsqu'il navigue sur le client web ou effectue des actions tel que l'envoi d'un fichier GPX.

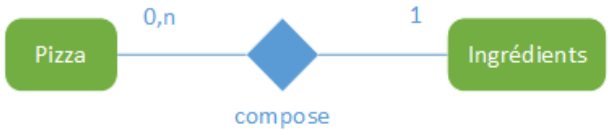
## 2.6 Amélioration de la navigation

L'application dans son état actuel ne rafraîchit pas la page des activités réalisés par un sportif lorsque celui-ci créer une nouvelle activité. Le sportif ne peut donc pas visualiser directement son parcours. Il serait préférable d'ajouter une fonctionnalité permettant de rafraîchir la page pour offrir une meilleure expérience utilisateur.

## 2.7 Modèle conceptuel des données




### 2.7.1 Exemple nomenclature




Une pizza se compose de zéro à plusieurs ingrédients

## 2.8 Stratégie de test

### 2.8.1 Étendue des tests

Des tests seront effectués sur les endpoints de l'API à l'aide du logiciel Postman.

Des tests manuels seront effectués sur l'interface graphique.

### 2.8.2 Testeurs

- Alexandre Philibert : alexandre.philibert@cpnv.ch

### 2.8.3 Type de tests

- Unitaire
- Intégration
- Fonctionnels
- Tests de charge

### 2.8.4 Liste des tests

- Use cases et Scenarii (décrit ci-dessous) à l'aide de Postman
- Tests manuels de l'IHM

### 2.8.5 Données de test à prévoir

Des données aléatoires seront créées pour effectuer des tests sur l'API lors de la création d'utilisateurs et d'activités sans GPX. Des données provenant d'activités sportives fournies par Monsieur Glassey seront également utilisées.

### 2.8.6 Logiciels de test

- Postman
- JMeter

### 2.8.7 Version des navigateurs

Les navigateurs supportés sont listés ci-dessous :

| Nom | Version testée |
|---|---|
| **Chrome** | Version 74.0.3729.169 (64 bits) |
| **Firefox** | 67.0 (64 bits) |

Deux navigateurs sont supportés car ils représentent ensemble plus de 66% du trafic internet[1]. Safari n'est pas supporté car il n'est pas possible d'installer Safari sur Windows. Internet Explorer et Edge ne sont pas supportés car certaines fonctionnalités utilisées dans l'interface client ne sont pas compatibles.

[1] https://www.w3counter.com/globalstats.php

### 2.9 Budget initial

Aucun budget n'est alloué à la réalisation de ce projet.

### 2.10 Planification finale

Est disponible en annexe (PHILIBERT_PlanificationFinale).

## 3 Conception

### 3.1 Ressources de l'API

| Ressource | Utilisation |
|---|---|
| **User*** | Stocke les données d'un utilisateur |
| **Token*** | Créer un token d'authentification |
| **Activity*** | Stocke une activité sportive réalisée par un sportif |
| **Subscription*** | Stocke l'abonnement d'un utilisateur (date de début, date de fin, …) |
| **Position*** | Stocke une position GPS à un temps donné |
| **Place** | Contient les localités des pays ou le sportif peut réaliser une activité |
| **Country** | Contient la liste des pays ou le sportif peut réaliser une activité |

* sont repris du Pré-TPI

### 3.2 Endpoints

| Fonction | Endpoints | Verbe HTTP |
|---|---|---|
| Création compte* | /user | POST |
| Modification compte* | /user/{userid} | PUT |
| Authentification* | /token | GET |
| Création d'activité* | /user/{userid}/activity/ | POST |
| Lecture d'activité* | activity/{activityid} | GET |
| | /user/{userid}/activity/{activityid} | |
| Modification d'activité* | activity/{activityid} | PUT |
| | /user/{userid}/activity/{activityid} | |

| | | |
|---|---|---|
| Suppression d'activité* | activity/{activityid} | DELETE |
| | /user/{userid}/activity/{activityid} | |
| Modification d'une position* | /position/{positionid} | PUT |
| Suppression d'une position* | /position/{positionid} | DELETE |
| Création type d'activité* | /activity-type/ | POST |
| Lecture types d'activité* | /activity-type/ | GET |
| Suppression type d'activité* | /activity-type/{typeid} | DELETE |
| Modification type d'activité* | /activity-type/{typeid} | PUT |
| Lecture des localités d'un pays | /country/{countryid}/place/ | GET |
| Lecture des pays | /country | GET |
| Lecture des localités | /place | GET |

* sont repris du Pré-TPI

### 3.3 Gestion lieu et pays

Le cahier des charges demande l'insertion de lieu et de pays de réalisation des activités sportives. Pour faciliter la création des activités pour le sportif, une liste des pays et des lieux seront accessible sur le endpoint country. Le cahier des charges ne demande pas la modification de la liste des lieux et pays par les utilisateurs, cette fonctionnalité ne sera donc pas intégrée.

Une liste contenant toutes les communes de Suisse a été récupérée : https://www.bfs.admin.ch/bfs/fr/home/bases-statistiques/repertoire-officiel-communes-suisse.assetdetail.6986904.html

## 3.4 Gestion des fichiers GPX

L'application dans son état actuel ne permet pas la gestion de fichier GPX trop volumineux (plusieurs Mb). Cela est dû au fait que le fichier GPX est envoyé dans le corps (body) de la requête HTTP d'une seule traite. Plusieurs possibilités existent pour résoudre ce problème :

### 3.4.1 Envoie de multiples requêtes

L'une des possibilités serait de découper le fichier GPX sur le client web et d'envoyer plusieurs requêtes consécutives contenant à chaque fois une partie du fichier. Cette méthode à comme désavantage de devoir renégocier le contexte d'envoi. Il serait aussi nécessaire de numéroter les parties de requêtes pour reconstruire le fichier dans l'ordre du côté l'API.

### 3.4.2 Requête "multipart/form-data"

Une autre possibilité est l'utilisation du *Content-Type multipart/form-data* du protocole HTTP. L'avantage de cette solution est la conservation du contexte de la requête, car une seule requête HTTP est envoyée au serveur contenant plusieurs parties. L'envoi de formulaires HTML est également simplifié avec cette solution, aucun traitement n'est nécessaire sur le client avant l'envoi du GPX.

**Formidable**

Le module node Formidable[1] permet de gérer les requêtes *multipart/form-data.* Ce module stocke néanmoins le fichier sur le serveur (dans le répertoire temporaire par défaut). Il serait nécessaire d'ajouter une gestion plus approfondie de ces fichiers.

**Multer**

Le module node Multer[2] permet également de gérer les requêtes *multipart/form-data.* L'avantage de ce module est de pouvoir stocker le fichier uniquement en mémoire vive, cette fonctionnalité n'est cependant pas nécessaire si nous souhaitons stocker le fichier sur le serveur.

**Solution retenue**

Le module Formidable semble, dans ce cas, plus adapté. Ce module n'intègre pas le stockage fichier en mémoire vive, mais il ne semble pas nécessaire d'intégrer cette fonctionnalité pour le moment. La syntaxe du module semble aussi plus simple que Multer.

## 3.5 Intégration du middleware Formidable

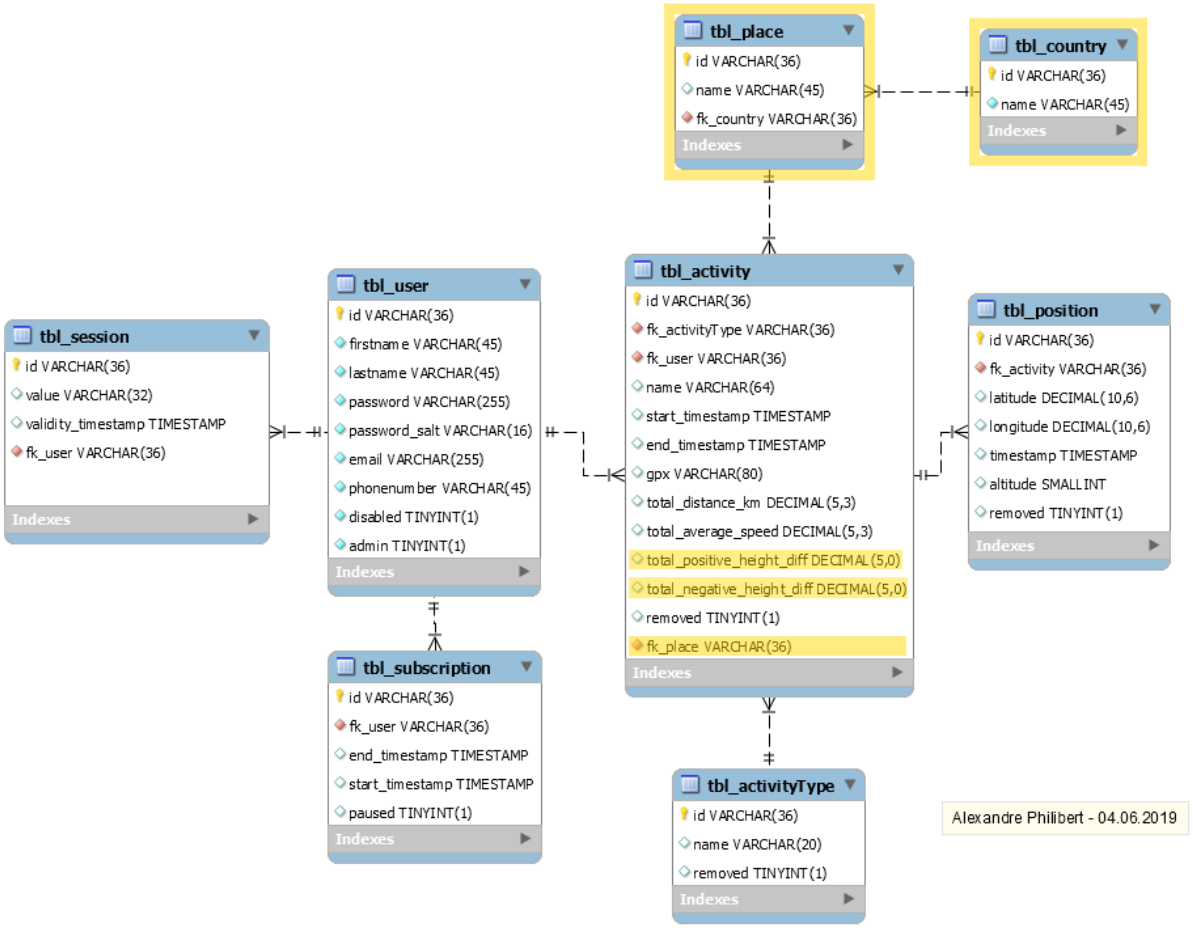
Un middleware doit être enregistré dans le routeur principal (src/router.js) pour gérer les requêtes *multipart/form-data* reçue par l'API. Le middleware sera enregistré dans le répertoire src/middlewares.

Il est nécessaire de modifier la table tbl_activity, le champ GPX stockant actuellement le fichier dans son intégralité devra désormais stocker le chemin vers le fichier dans le système de fichier de l'OS.

[1] https://www.npmjs.com/package/formidable

[2] https://www.npmjs.com/package/multer

## 3.6 Modèle logique des données




** Les champs en jaunes sont les champs ajoutés lors du TPI*

| **tbl_activity** | |
|---|---|
| **Champs** | **Utilisation** |
| **gpx** | chemin jusqu'au fichier GPX stocké sur le serveur |
| **total_positive_height_diff** | Dénivelé positif total exprimé en valeur positive |
| **total_negative_height_diff** | Dénivelé négatif total exprimé en valeur positive |
| **tbl_country** | |
| **Champs** | **Utilisation** |
| **name** | Nom du pays |
| **tbl_place** | |
| **Champs** | **Utilisation** |
| **name** | Nom de la localité |

### 3.6.1 Unicité

L'unicité des tables est gérée par la base de données pour la table des utilisateurs (champ email). La table d'activité ne possède pas d'unicité car valider que le fichier téléchargé est unique demanderai de passer dans tous les fichiers téléchargés.

## 3.7 Serveur web

Plusieurs solutions de serveurs HTTP existent, la solution mise en œuvre actuellement sur l'application est assez basique en NodeJS. Lorsqu'un client demande un page web, l'application fait une correspondance entre le nom demandé est un nom de fichier sur le serveur, l'application va ensuite lire le fichier et l'envoyer au client. Le fichier est lu à chaque nouvelle requête, aucun cache n'est présentement en place. Les contraintes de temps de permettent pas de mettre en place un système plus avancé que celui-ci.

### 3.7.1 Mapping des clés étrangères

L'API ne permet actuellement pas d'envoyer une réponse contenant les ressources étant référencées dans la ressource demandée. Plusieurs solutions peuvent être implémentées pour améliorer les performances et simplifier les requêtes API. La solution implémentée actuellement dans l'API ne permet pas de découvrir les ressources liées. Les ressources liées ne sont non plus pas intégrées dans la réponse. Il est nécessaire d'effectuer d'autres requêtes pour récupérer les ressources liées.

**HATEOAS**

Hypermedia As The Engine Of Application State (HATEOAS) permet d'inclure des liens dans la réponse de l'API vers des ressources liées. Cette solution apporte l'avantage d'avoir une petite réponse, car les ressources liées ne sont pas contenues dans la réponse. Le désavantage de cette solution est que si le client effectue à chaque requête des requêtes vers les ressources liées, cela créera énormément de requêtes.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/accounts/12345/deposit" />
    <link rel="withdraw" href="/accounts/12345/withdraw" />
    <link rel="transfer" href="/accounts/12345/transfer" />
    <link rel="close" href="/accounts/12345/close" />
</account>
```

Exemple prit de : https://phauer.com/2015/restful-api-design-best-practices/

### Sideloading

Le *sideloading* permet d'intégrer les ressources liées à la ressource demandée. Cette solution évite de créer de nouvelles requêtes pour résoudre les ressources liées. Les ressources liées sont chargée une seule fois, le client ne doit pas gérer un cache des objets liés

```
{
    "data": [{
        "id": 3549,
        "name": "John",
        "rel": {
            "city": 45
        },
        "extras": {
            "city": {
                "id": 45,
                "name": "Lausanne"
            }
        }
    }]
}
```

### Embedding

Cette solution est similaire au *sideloading*, les différences étant que les références vers les ressources liées peuvent être dupliquées dans la réponse et les objets contiennent directement les ressources liées.

```
{
    "data": [{
        "id": 3549,
        "name": "John",
        "city": {
            "id": 45,
            "name": "Lausanne"
        }
    }]
}
```

### 3.7.2 Gestion du multithreading

Le multithreading[1] est géré dans NodeJS grâce au module Cluster. La documentation officielle de NodeJS dit:

"*A single instance of Node.js runs in a single thread. To take advantage of multi-core systems, the user will sometimes want to launch a cluster of Node.js processes to handle the load*."

Le module Cluster permet de créer des processus enfants partageant le même port HTTP. Cela permet de séparer la charge entrante sur le serveur sur les différents ports.

**Pseudo Code**

```
const cpus = os.cpus();
if (cluster.isMaster) {
    // The main cluster creates new workers
    for (let i = 0; i < cpus.length; i++) {
        cluster.fork();
    }

    cluster.on('exit', (worker, code, signal) => {
        console.log('an error happened');
    });
} else {
    // Task new workers are doing
    http.createServer((req, res) => {

    }).listen(PORT);
}
```

## 3.8 Client web

L'architecture cliente actuellement en place sur l'application distribue des pages statiques. Chaque page est un nouveau fichier HTML. L'architecture du client va cependant être modifiée pour permettre aux sportifs d'avoir une meilleure expérience utilisateur.

Les désavantages avec la solution en place actuellement est qu'il n'est pas possible de réutiliser des parties du DOM entre plusieurs pages (tel que le menu par ex.).

Les navigateurs supportés seront Firefox et Chrome car Chrome représente la plus grande part du marché des navigateurs (89.9% avril 2019[2]) et Chrome ainsi que Firefox intègrent toutes les nouvelles fonctionnalités de ES6. Safari n'est pas supporté car la version Windows de Safari n'est plus supportée par Apple et par manque de temps.

[1] Aptitude d'exécuter des processus de manière concurrente sur plusieurs threads

[2] https://www.w3schools.com/browsers/

### 3.8.1 Architecture client

#### Pages statiques et appels XHR

Cette solution est actuellement mise en place sur l'application. Elle est la plus simple à mettre en œuvre car elle ne nécessite pas de traitements particuliers. Des appels XHR[1] sont cependant nécessaire pour personnaliser le contenu des pages tel que l'affichage des activités appartenant à un sportif.

Cette solution ne permet pas de simplement réutiliser des parties du DOM tel que le menu par exemple.

#### Génération des pages en PHP

La génération des pages en PHP peut nécessiter la mise en place d'un serveur HTTP tel qu'Apache ou NGNIX. Cette solution génère les pages servies au client sur le serveur, il est donc possible de réutiliser des parties du DOM[2] en utilisant des designs patterns tel que le MVC[3].

#### Génération de pages en Javascript (côté client)

La génération du DOM sur le client permet de diminuer la charge du serveur. Il est aussi possible grâce à cette solution de garder le contexte d'exécution car aucun rechargement de pages n'est nécessaire.

Cette solution a été choisie car elle permet de garder une application développée uniquement en Javascript autant du côté back-end que front-end.

### 3.8.2 Choix

La solution retenue est la génération des pages en Javascript car elle permet d'avoir une interface plus réactive. Il est également possible avec cette solution de réutiliser des éléments du DOM.

### 3.8.3 Framework de génération de pages côté client

#### Routage

Pour une gestion très facile du routage des pages, il est possible d'implémenter le *listener onhashchange* sur l'objet *window*. Cela permet de ne pas recharger la page à chaque changement. Il est possible d'enregistrer une route à l'aide d'une méthode sur l'objet router qui associe un path et une fonction de génération de page.

Il serait par la suite possible d'implémenter une solution de routage plus avancée pour ne pas utiliser de hash dans l'URL. Cette solution n'est cependant pas nécessaire dans ce projet car elle ne présente pas d'améliorations.

#### Pages

Les pages sont stockées dans un dossier nommé "pages". Chaque page est un export ES6 d'une fonction qui créé la page. Toute la logique est gérée dans ces fichiers. Cela permet de ne pas remplir le contexte global de variables.

---

[1] XMLHttpRequest, interface Javascript permettant d'effectuer des requêtes HTTP

[2] Document Object Model, interface de programmation pour documents HTML

[3] Modèle Vue Contrôleur, Paterne architectural souvent utilisé pour le développement d'interface utilisateurs

### Éléments réutilisables

Les éléments réutilisables sont également stockés dans le dossier "pages". Ils ne sont cependant pas enregistrés dans le router sous une route. Le seul élément réutilisable pour le moment est le header qui est append à la page dans le fichier boot.js.

### Rafraichissement de pages

Dans certains cas, il est nécessaire de rafraichir une page pour mettre à jour son contenu. La page des activités doit être rafraichie lorsqu'une nouvelle activité est créée par exemple. L'ajout d'une méthode au routeur permettant d'enlever une page de la liste des pages chargées permettrai de résoudre ce problème.

Une méthode permettant de décharger une page se trouvant dans le cache possède l'avantage de ne pas recharger les ressources se trouvant dans la page quand le sportif ne s'y trouve pas.

Cette solution permet de recharger la page dans son intégralité. Il serait, par la suite, peut être nécessaire d'intégrer un système permettant une plus grande précision dans le rechargement.

## 3.8.4 Création d'activité – Story001 et Story002

L'interface de création des activités sportives sans GPX va évoluée et permettra désormais de créer des activités sportives avec GPX. Cette évolution permet de ne pas créer une nouvelle page qui disposera des mêmes champs. L'interface doit devenir plus intelligente pour permettre la création d'activités sans et avec GPX.

Lorsqu'un sportif ajoute un fichier GPX au formulaire, les champs de la colonne de droite (Début, Fin, Distance, Dénivelés) ne sont plus nécessaires à l'envoi du formulaire. Le sportif est informé de ces nouvelles contraintes par un message sur l'interface graphique.

Lors de la création d'une activité, la validation s'effectue en deux étapes. La première étape est la validation côté client des informations entrées (type, champs requis, valeurs minimales et maximales). Une seconde validation est faite côté serveur pour vérifier que la requête HTTP et les champs sont bien constitués.

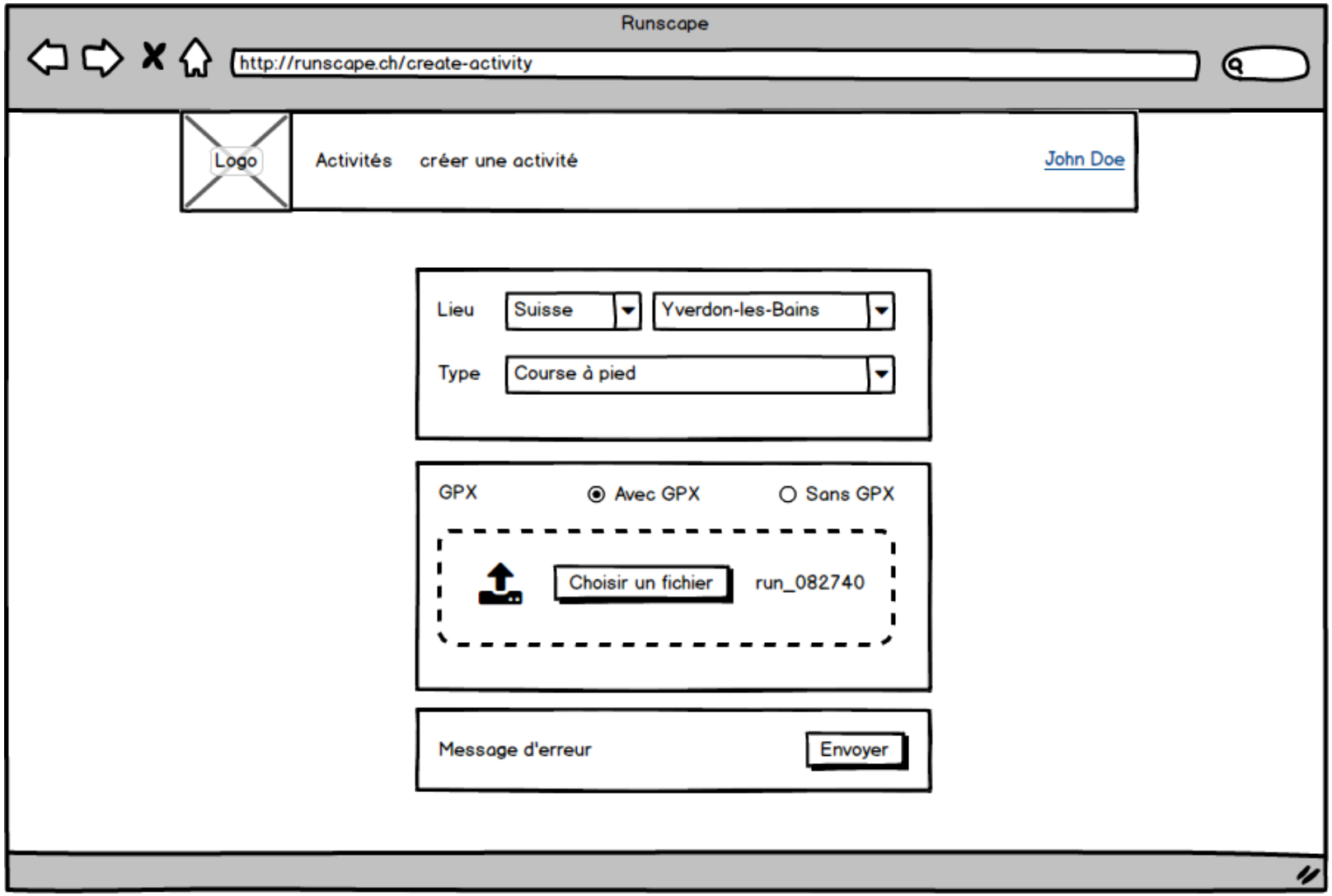
Dans un premier temps, aucune validation du GPX ne sera faite côté client. Cette fonctionnalité pourra être intégrée si du temps est encore disponible en fin de projet.

### Zoning

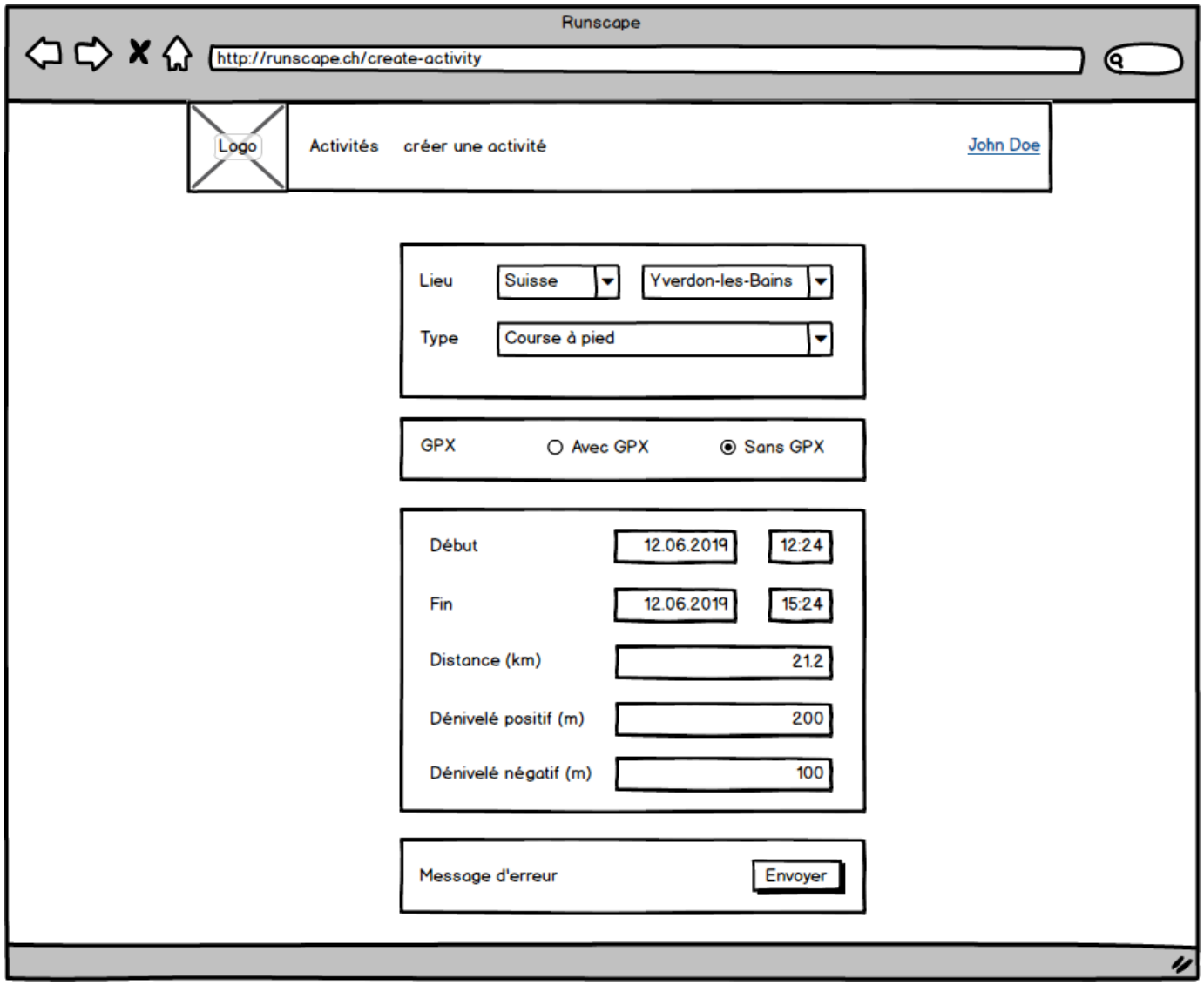
Il a été décidé de ne pas créer de zoning pour l'interface client car celle-ci reste très simple. Une attention particulière sera portée aux wireframes et à la création de l'interface pour compenser cette décision.

**Wireframe**

Création d'une activité avec GPX



Création d'une activité sans GPX



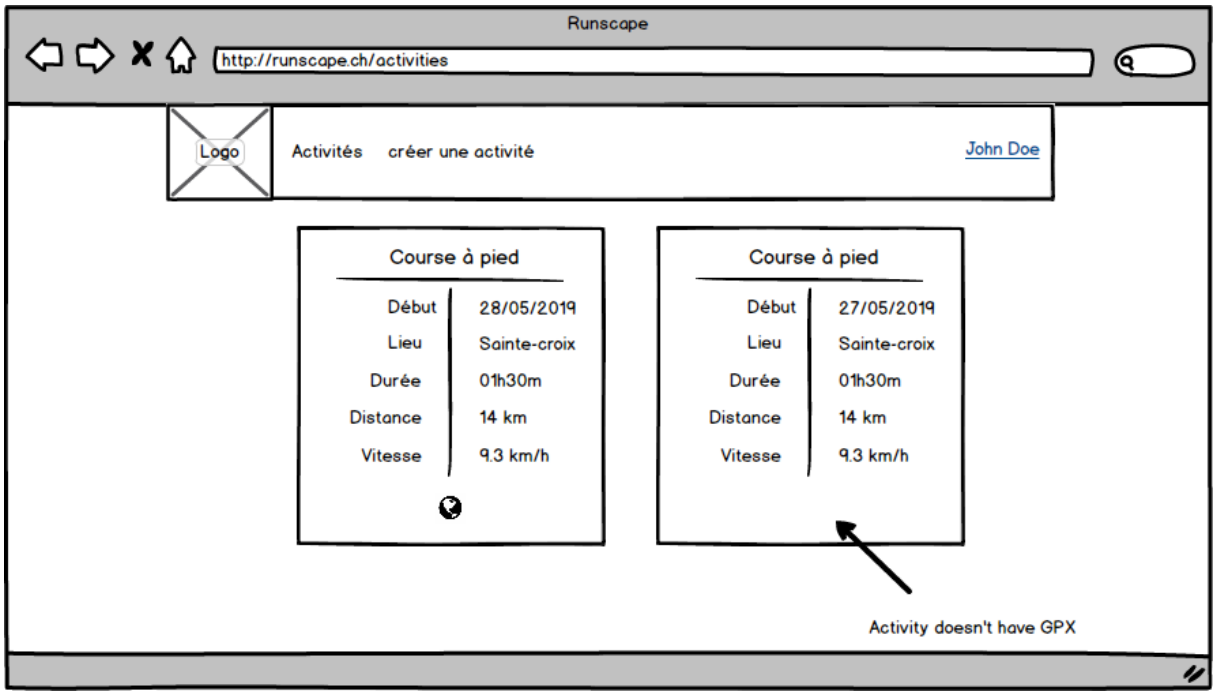
**Validation**

| Champ | Type (HTML) | Commentaires |
|---|---|---|
| **Nom** | text | |
| **Lieu** | select / select | Les champs Pays et ville sont regroupé sur la même ligne |
| **Type** | select | Les données récupérées depuis /activity-type |
| **Début** | date / time | Une agrégation du champ de date et du temps sera nécessaire en Javascript |
| **Fin** | date / time | Une agrégation du champ de date et du temps sera nécessaire en Javascript |
| **Distance** | number | |
| **Dénivelé pos.** | number | |
| **Dénivelé neg.** | number | |
| **GPX** | file | Permet de sélectionner un fichier |

Des contraintes de valeurs minimales et maximales seront ajoutés sur certains champs.

### 3.8.5 Dashboard des activités

**Wireframe**




L'icône  permet au sportif d'observer son entraînement sur une carte.

### 3.8.6 Interprétation du parcours (Story003)

Le sportif doit avoir la possibilité d'observer son activité sur une carte. Cette fonctionnalité sera disponible uniquement pour les activités crées avec un fichier GPX, car le client web ne fournit pas d'interface permettant de dessiner un parcours.

Différents services de cartographies pourraient être choisi pour l'affichage du parcours :

**Google Maps**

Google Maps propose une API nommée **Polylines** qui permet d'envoyer des coordonnées GPS à l'API. Celle-ci s'occupe ensuite du rendu sur la carte. L'API Polylines semble plus simple à utiliser que OpenLayers d'Open Street Maps. Google Maps étant plus connus, les gens ne seront pas désorientés par l'interface de la carte.

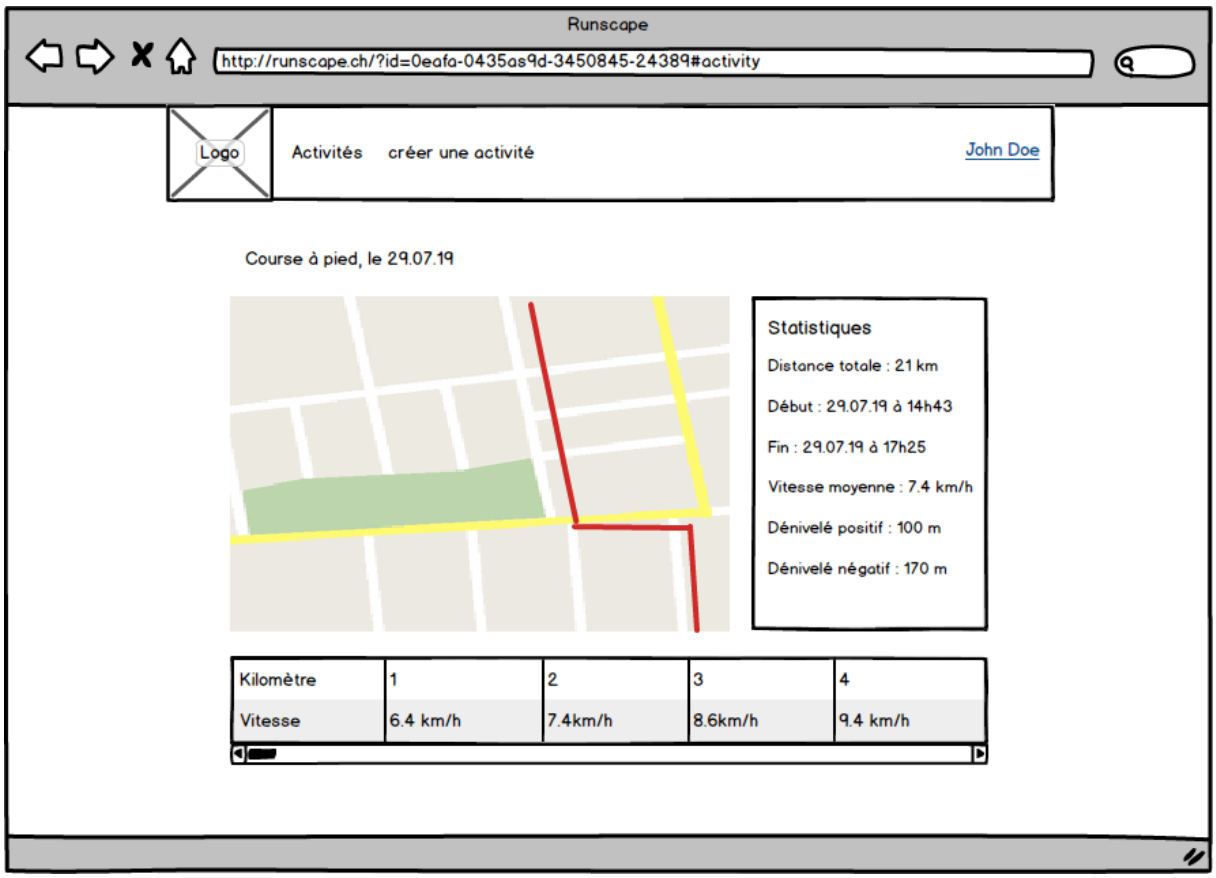
**Open Street Map**

Open Street Map propose une API nommée **Openlayers** permet d'afficher le parcours de fichiers GPX. L'usage commercial de l'API Open Street Map est cependant interdit. L'API semble plus compliquée à utiliser que celle de Google Maps.

**Choix**

Google Maps sera le service de cartographie implémenté dans l'application car il est le plus connus, les sportifs se retrouverons surement plus facilement dans l'interface. Il est envisageable d'intégrer Open Street Map par la suite comme seconde option si du temps est encore disponible pour le développement.

### Wireframe




### Référence à l'identifiant d'une activité

L'application doit pouvoir récupérer l'identifiant d'une activité sportive lorsqu'un utilisateur clique sur la vue du parcours dans la liste des activités. Le sportif pourra également enregistrer le lien vers le parcours et retourner sur celui-ci en ne passant pas par la vue de toutes les activités. Il est nécessaire de stocker les paramètres dans l'URL de la page. Les paramètres GET permettent de facilement implémenter cela.

### Pseudo code carte

```
map = new google.maps.Map(mapElement);
points;
foreach gpxPoint in gpxPoints {
    point = new google.maps.LatLng(gpxPoint.lat, gpxPoint.lon);
    points.push(point);
}
trackLine = new google.maps.Polyline({
    path: points;
});
trackLine.setMap(map);
```

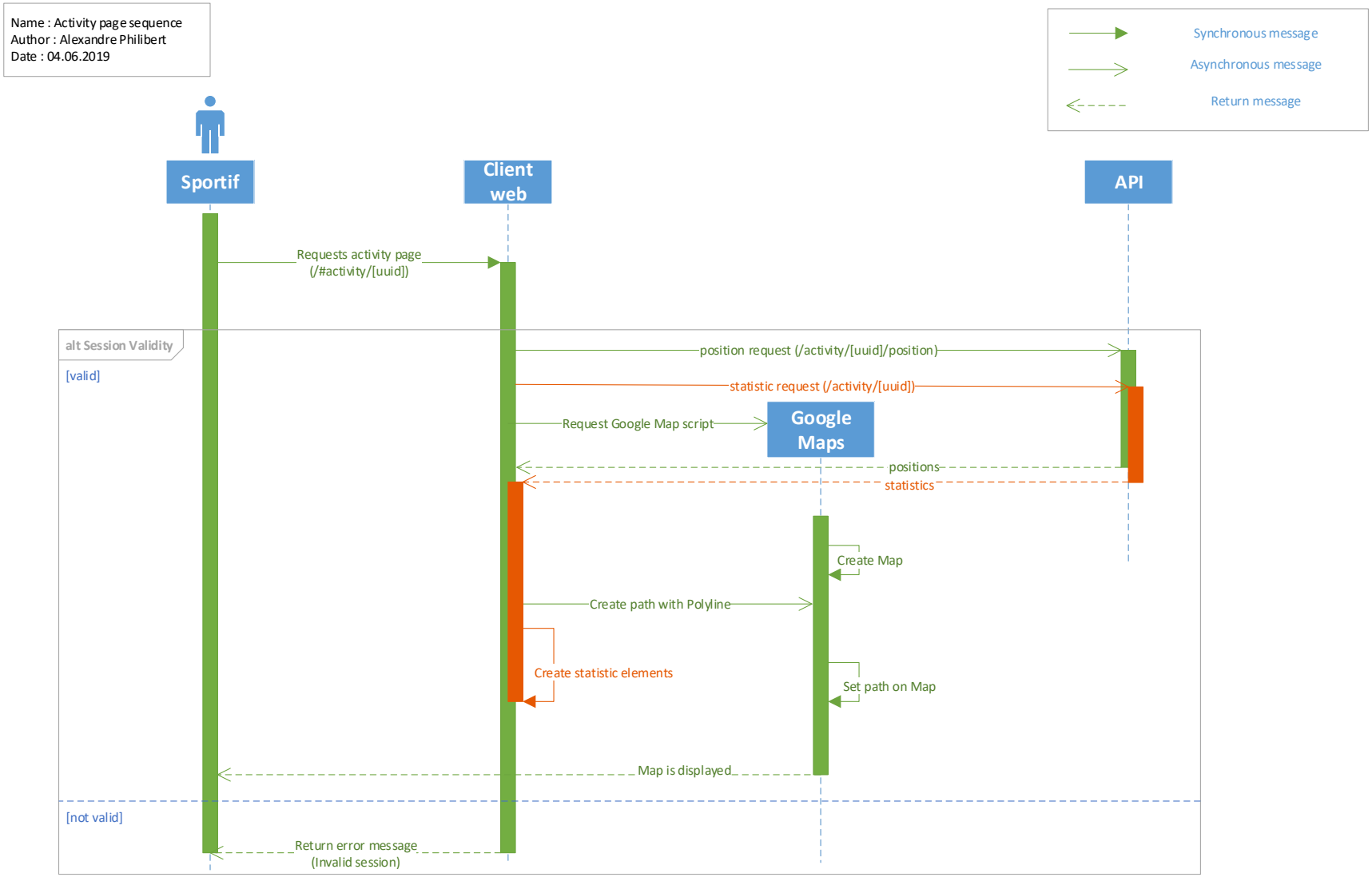
Réalisé à l'aide de :

https://developers.google.com/maps/documentation/javascript/examples/polyline-simple

### Statistiques

L'ajout de statistiques à la page nécessite de récupérer des attributs de la ressource activity qui est déjà appelé sur la page. Il pourrait s'avérer utile de pouvoir transmettre des objets entre les activités. Il serait dans ce cas plus nécessaire d'effectuer une requête sur la ressource activity si celle-ci aurait déjà été chargée précédemment (Lors de l'appel sur toutes les activités par exemple).

## Diagramme de séquence



### 3.9 Structure de fichiers

```
/
|    app.js      contains the listener for the website and api
|---api
|    |    app.js
|    |
|    |---node_modules    modules for the api
|    |---src
|         |    router.js   router entrypoint (handling all routes)
|         |    util.js    string handling, object handling, etc...)
|         |
|         |---config
|         |        db.js   database config file (see README.md)
|         |
|         |---middlewares on all routes (error, formdata, auth)
|         |---models      model of API objects (Activity, etc..)
|         |---routes      using endpoints and calling models
|
|---node_modules         modules for the project
|---website
     |    app.js          distribute static content to client
     |
     |---node_modules    module for the website
     |---public
     |    |    boot.js     web client boot sequence
     |    |    global.js   global objects
     |    |    index.html  single HTML file
     |    |    router.js   router object
     |    |    util.js     utilities (cookie handling)
     |    |
     |    |---img
     |    |---pages
     |    |
     |    |---scripts     depedencies (modules used on client side)
     |    |        gpsDistance.js
     |    |
     |    |---styles
     |---src              PRE-TPI pages (not used)
```

## 3.10 Cas d'utilisations

### 3.10.1 Sportif

| | | |
|---|---|---|
| En tant que sportif | Je veux enregistrer une activité sportive sans GPX à l'aide de l'API | Dans le but de suivre ma progression |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Soumet une requête POST à "user/{userid}/activity" |
| | 2 | API : Valide la requête |
| | 3 | API : Valide l'autorisation de création de la ressource |
| | 4 | API : enregistre les informations dans la base de données |
| | 5 | API : Retourne l'état de l'enregistrement au membre |
| **Extensions** | 2a | Autorisation non valide<br>API : Retourne une erreur d'autorisation |
| | 3a | Format de date de début invalide<br>API : Retourne une erreur de format de date |
| | 3b | Format de date de fin invalide<br>API : Retourne une erreur de format de date |
| | 3c | ID de type d'activité inexistant<br>API : Retourne une erreur de format d'activité |
| | 3d | Champs fichier GPX vide, autres champs vides<br>API : Retourne une erreur de champs manquant |

| | | |
|---|---|---|
| En tant que sportif | Je veux enregistrer une activité sportive sans GPX à l'aide de l'interface graphique | Dans le but de suivre ma progression |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Remplis le formulaire de création d'activité sur l'interface graphique sans le champ GPX |
| | 2 | Client web: Valide le formulaire |
| | 3 | Client web: Soumet la requête à l'API |
| | 4 | API : Valide la requête |
| | 5 | API : enregistre les informations dans la base de données |
| | 6 | API : Retourne l'état de l'enregistrement au client web |
| | 7 | Client web: Traite la réponse de l'API et redirige le sportif sur la page des activités |
| **Extensions** | 2a | Champs non remplis<br>Client web: Informe le sportif des champs non remplis |
| | 4a | Autorisation non valide<br>API : Retourne une erreur d'autorisation |
| | 4b | Format de date de début invalide<br>API : Retourne une erreur de format de date au client |
| | 4c | Format de date de fin invalide<br>API : Retourne une erreur de format de date au client |
| | 4d | ID de type d'activité inexistant<br>API : Retourne une erreur de format d'activité au client |
| | 4e | Champs nécessaires vides<br>API : Retourne une erreur de champs manquant au client |

| | | |
|---|---|---|
| En tant que sportif | Je veux enregistrer une activité sportive avec GPX à l'aide de l'API | Dans le but de suivre ma progression |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Soumet une requête POST à "user/{userid}/activity" |
| | 2 | API : Valide la requête |
| | 3 | API : Valide l'autorisation de création de la ressource |
| | 4 | API : enregistre les informations dans la base de données |
| | 5 | API : Retourne l'état de l'enregistrement au membre |
| **Extensions** | 2a | Autorisation non valide<br>API : Retourne une erreur d'autorisation |
| | 2b | Format de GPX invalide<br>API : Retourne une erreur de format GPX invalide |
| | 3a | Format de date de début invalide<br>API : Retourne une erreur de format de date |
| | 3b | Format de date de fin invalide<br>API : Retourne une erreur de format de date |
| | 3c | ID de type d'activité inexistant<br>API : Retourne une erreur de format d'activité |
| | 3d | Champs fichier GPX vide, autres champs vides<br>API : Retourne une erreur de champs manquant |

| | | |
|---|---|---|
| En tant que sportif | Je veux enregistrer une activité sportive avec GPX à l'aide de l'interface graphique | Dans le but de suivre ma progression |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Remplis le formulaire de création d'activité sur l'interface graphique avec le champs GPX |
| | 2 | Client web: Valide le formulaire |
| | 3 | Client web: Soumet la requête à l'API |
| | 4 | API : Valide la requête |
| | 5 | API : enregistre les informations dans la base de données |
| | 6 | API : Retourne l'état de l'enregistrement au client web |
| | 7 | Client web: Traite la réponse de l'API et redirige le sportif sur la page de visualisation |
| **Extensions** | 2a | Champs non remplis<br>Client web: Informe le sportifs des champs non remplis |
| | 4a | Autorisation non valide<br>API : Retourne une erreur d'autorisation |
| | 4b | Format de date de début invalide<br>API : Retourne une erreur de format de date au client |
| | 4c | Format de date de fin invalide<br>API : Retourne une erreur de format de date au client |
| | 4d | ID de type d'activité inexistant<br>API : Retourne une erreur de format d'activité au client |
| | 4e | Champs nécessaires vides<br>API : Retourne une erreur de champs manquant au client |

| | | |
|---|---|---|
| En tant que sportif | Je veux consulter la liste des pays | Dans le but de choisir le pays où j'ai réalisé mon activité |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Soumet une requête POST à "country" |
| | 2 | API : Valide la requête |
| | 5 | API : Retourne la liste des pays |

| | | |
|---|---|---|
| En tant que sportif | Je veux consulter la liste des localité d'un pays | Dans le but de choisir la localité où j'ai réalisé mon activité |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Soumet une requête POST à "country/{countryid}/place" |
| | 2 | API : Valide la requête |
| | 3 | API : Retourne la liste des localités du pays |
| **Extensions** | 2a | L'identifiant du pays (countryid) n'existe pas<br>API : Retourne une liste vide |

| | | |
|---|---|---|
| En tant que sportif | Je veux consulter la liste des activités que j'ai réalisées | Dans le but d'analyser mes performances |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Se rend sur la page d'affichage des activités qu'il a réalisé |
| | 2 | Client web : Valide la connexion du sportif |
| | 3 | Client web : Envoi une requête à l'API pour récupérer les activités du sportif |
| | 4 | API : Valide la requête |
| | 5 | API : Retourne la liste des activités du sportif |
| | 6 | Client web: Affiche la liste des activités |
| **Extensions** | 2a | Le token n'est pas présent dans les cookies<br>Client web : affiche une page d'erreur |

| | | |
|---|---|---|
| En tant que sportif | Je veux pouvoir observer mon entraînement sur une carte | Dans le but d'analyser plus précisément mes performances |

| Déroulement | Étape | Description |
|---|---|---|
| | 1 | Sportif : Se rend sur la page d'une activité |
| | 2 | Client web : Valide la connexion du sportif |
| | 3 | Client web : Envoi une requête à l'API pour récupérer les positions de l'activité |
| | 4 | API : Valide la requête |
| | 5 | API : Retourne la liste des positions d'une activité |
| | 6 | Client web: Affiche le parcours sur une carte |
| **Extensions** | 2a | Le token n'est pas présent dans les cookies<br>Client web : affiche une page d'erreur |
| | 5a | Aucune position n'est récupérée<br>Client web : informe le sportif qu'aucune position n'a été trouvée |
| | 5b | L'API retourne une erreur<br>Client web : informe le sportif qu'une erreur est survenue |

# 4 Implémentation

## 4.1 Problèmes rencontrés & Découvertes

### 4.1.1 Format de retour API

Lors de la création de la page de dashboard contenant toutes les activités d'un sportif, je me suis aperçu que, dans l'architecture que j'avais conçue, le client web est chargé d'effectuer le mapping des ressources liées sur l'API. N'ayant pas le temps d'implémenter un nouveau système, j'ai documenté les potentielles évolutions possibles. La documentation des évolutions possibles se situe au point 3.7.1.

### 4.1.2 Multithreading

Pendant le développement de l'interface graphique est les tests, il a été trouvé que le serveur AWS répondait lentement aux requêtes. Il a d'abord été pensé que le problème venait de la nature monothread de l'application. Il a donc été décidé d'intégrer le multithreading sur la gestion des requêtes entrante pour vérifier si le problème pouvait survenir de l'architecture monothread. Après implémentation, il a été constaté que le problème ne venait pas de la gestion des threads mais semble plutôt être un problème de bande passante sur le réseau.

## 4.2 Requêtes multipart/form-data

L'envoi des fichiers GPX sur l'API a demandé beaucoup de réflexion, ces réflexions peuvent être retrouvées dans la section 3.4. Cette fonctionnalité n'était pas demandée dans le cahier des charges mais elle a permis de grandement facilité l'envoi des fichiers depuis le client web. Cette fonctionnalité permet également d'envoyer des fichiers plus volumineux, car les requêtes multipart/form-data sont envoyé en plusieurs parties par le navigateur.

## 4.3 Description des tests effectués

### 4.3.1 Calcul de la vitesse par kilomètre

Le temps d'exécution de la fonction de calcul de la vitesse par kilomètre a été calculé à l'aide de la méthode console.time(). Ce test a été réalisé sur le fichier GPX nommé "run-20181229T155040.gpx"[1] contenant plus de 20'000 positions GPS et d'une taille de 3,83Mo stocké dans la base de données et récupéré au format JSON.

Temps d'exécution : 4ms

### 4.3.2 Multithreading

Ce test a été réalisé à l'aide de JMeter[2] sur le endpoint /api/user/[userid]/activity d'un utilisateur ayant 56 activités sportives. Les paramètres de JMeter étaient :

| Paramètre | Valeur |
|---|---|
| **Number of Threads (users)** | 100 |
| **Ramp-Up Period** | 0 |
| **Loop Count** | 100 |

**Résultats**

| # Threads | #Samples | Througput req/sec |
|---|---|---|
| **1** | 10'000 | 441.4/sec |
| **8** | 10'000 | 1084.5/sec |

Une amélioration de près de 2.5 fois le nombre de requête peut être constatée dans ce scénario.

[1] Disponible en annexe

[2] https://jmeter.apache.org/

### 4.3.3 Scenarii

| | | |
|---|---|---|
| En tant que sportif | Je veux consulter la liste des pays | Dans le but de choisir le pays où j'ai réalisé mon activité |

| Déroulement | Étape | Description | Résultat |
|---|---|---|---|
| | 1 | Membre : Soumet une requête POST à "country" | |
| | 2 | API : Valide la requête | |
| | 5 | API : Retourne la liste des pays | Réussi |

| | | |
|---|---|---|
| En tant que sportif | Je veux consulter la liste des localité d'un pays | Dans le but de choisir la localité où j'ai réalisé mon activité |

| Déroulement | Étape | Description | Résultat |
|---|---|---|---|
| | 1 | Membre : Soumet une requête POST à "country/{countryid}/place" | |
| | 2 | API : Valide la requête | |
| | 5 | API : Retourne la liste des localités du pays | Réussi |
| **Extensions** | 2a | L'identifiant du pays (countryid) n'existe pas<br>API : Retourne une liste vide | Réussi |

| | | |
|---|---|---|
| En tant que sportif | Je veux enregistrer une activité sportive sans GPX à l'aide de l'interface graphique | Dans le but de suivre ma progression |

| Déroulement | Étape | Description | Résultat |
|---|---|---|---|
| | 1 | Sportif : Remplis le formulaire de création d'activité sur l'interface graphique sans le champ GPX | |
| | 2 | Client web: Valide le formulaire | |
| | 3 | Client web: Soumet la requête à l'API | |
| | 4 | API : Valide la requête | |
| | 5 | API : enregistre les informations dans la base de données | |
| | 6 | API : Retourne l'état de l'enregistrement au client web | |
| | 7 | Client web: Traite la réponse de l'API et redirige le sportif sur la page de visualisation | |
| **Extensions** | 2a | Champs non remplis<br>Client web: Informe le sportifs des champs non remplis | Réussi<br>(Les champs deviennent rouges) |
| | 4a | Autorisation non valide<br>API : Retourne une erreur d'autorisation | Réussi<br>L'erreur ne spécifie pas l'autorisation non valide |
| | 4b | Format de date de début invalide<br>API : Retourne une erreur de format de date au client | Réussi |
| | 4c | Format de date de fin invalide<br>API : Retourne une erreur de format de date au client | Réussi |
| | 4d | ID de type d'activité inexistant | Réussi |

| | | | |
|---|---|---|---|
| | | API : Retourne une erreur de format d'activité au client | |
| | 4e | Champs nécessaires vides<br>API : Retourne une erreur de champs manquant au client | Réussi |

| | | |
|---|---|---|
| En tant que sportif | Je veux enregistrer une activité sportive avec GPX à l'aide de l'interface graphique | Dans le but de suivre ma progression |

| Déroulement | Étape | Description | Résultat |
|---|---|---|---|
| | 1 | Sportif : Remplis le formulaire de création d'activité sur l'interface graphique avec le champs GPX | |
| | 2 | Client web: Valide le formulaire | |
| | 3 | Client web: Soumet la requête à l'API | |
| | 4 | API : Valide la requête | |
| | 5 | API : enregistre les informations dans la base de données | |
| | 6 | API : Retourne l'état de l'enregistrement au client web | |
| | 7 | Client web: Traite la réponse de l'API | |
| **Extensions** | 2a | Champs non remplis<br>Client web: Informe le sportifs des champs non remplis | Réussi |
| | 4a | Autorisation non valide<br>API : Retourne une erreur d'autorisation | Réussi<br>L'erreur ne spécifie pas l'autorisation non valide |
| | 4b | Format de date de début invalide<br>API : Retourne une erreur de format de date au client | Réussi |

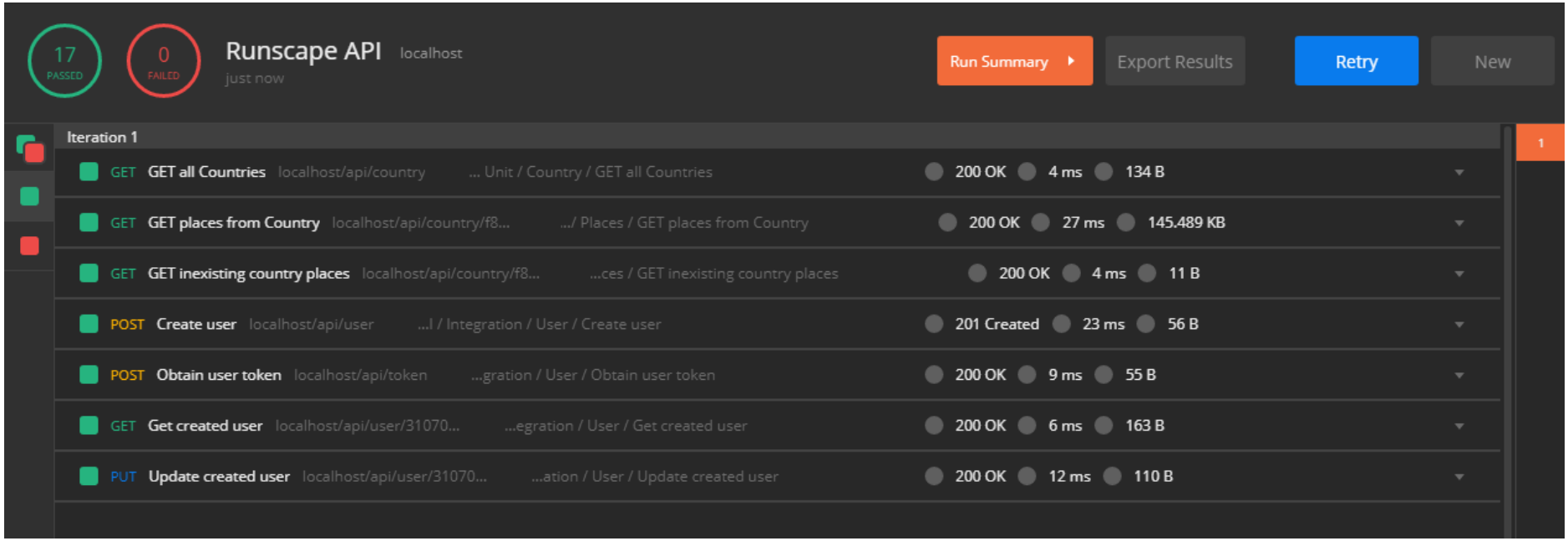
| | | | |
|---|---|---|---|
| | 4c | Format de date de fin invalide<br>API : Retourne une erreur de format de date au client | Réussi |
| | 4d | ID de type d'activité inexistant<br>API : Retourne une erreur de format d'activité au client | Réussi |
| | 4e | Champs nécessaires vides<br>API : Retourne une erreur de champs manquant au client | Réussi |

| | | |
|---|---|---|
| En tant que sportif | Je veux consulter la liste des activités que j'ai réalisées | Dans le but d'analyser mes performances |

| **Déroulement** | **Étape** | **Description** | **Résultat** |
|---|---|---|---|
| | 1 | Sportif : Se rends sur la page d'affichage des activités qu'il a réalisé | |
| | 2 | Client web : Valide la connexion du sportif | |
| | 3 | Client web : Envoi une requête à l'API pour récupérer les activités du sportif | |
| | 4 | API : Valide la requête | |
| | 5 | API : Retourne la liste des activités du sportif | |
| | 6 | Client web: Affiche la liste des activités | |
| **Extensions** | 2a | Le token n'est pas présent dans les cookies<br>Client web : affiche une page d'erreur | Réussi<br>La page not found s'affiche avec un message |

### 4.3.4 Tests Postman

Vous trouverez, ci-dessous, une liste des tests Postman :




## 4.4 Erreurs restantes

Les erreurs sont répertoriées sur Github dans l'application Issue : https://github.com/AlexandrePhilibertCPNV/TPI_PHILIBERT

# 5 Conclusion

## 5.1 Objectifs atteints

Tous les objectifs du cahier des charges ont été atteints. Il reste cependant des bugs qui sont référencés sur Github.

## 5.2 Suites possible pour le projet

### 5.2.1 Améliorer la création d'activité

La création d'activité étant l'élément central de l'application, il serait judicieux d'améliorer cette partie de l'application. Lorsqu'un sportif créer une activité à l'aide d'un GPX, celui-ci se voit redirigé sur la page de visualisation de l'activité. Il peut arriver que lorsque le sportif est redirigé sur cette page, les positions ne sont pas encore enregistrées dans la base de données. Il n'est donc pas encore possible de dessiner le parcours sur la carte.

L'amélioration à apporter peut prendre plusieurs formes :

- La redirection ne s'effectue que lorsque les positions ont été enregistré dans la base de données. Dans ce scénario le sportif verra directement le parcours lorsqu'il arrive sur la page.
- Aucune redirection vers la page de l'activité n'est faite. L'utilisateur reste sur la page de création d'activité et un message confirmant la création de son activité s'affiche.
- La redirection s'effectue directement après la création, mais le client web effectues des requêtes jusqu'à ce que les positions sont disponibles.

### 5.2.2 Améliorer le passage de données entre les activités

Le client web utilise actuellement une solution de routage très simple. Il n'est, par conséquence, pas possible de transférer entres les activités d'autres informations que des valeurs sous forme de champs texte. Les possibilités imaginées pour permettre un meilleur changement de contexte seraient :

- La modification du routeur pour permettre l'envoie d'Objets Javascript entre les activités
- L'ajout d'une classe contenant les données à récupérer dans la nouvelle activité. Cette solution est notamment utilisée dans le framework Android.

### 5.2.3 Améliorer les ressourcées liés dans l'API

Cette fonctionnalité a été documenté dans la conception: mapping des clés étrangères 3.7.1.

## 5.3 Difficultés particulières

La principale difficulté lors du développement du projet a été la gestion du temps. Il était à certains moments nécessaire de faire des choix simplifiant l'architecture de l'application au détriment du développement d'outils plus réutilisables.

### 5.4 Bilan personnel

Ce projet m'a permis de continuer le travail que j'avais déjà effectué lors d'un projet précédent. Cela m'a permis d'aller plus loin dans mes réflexions lors de la conception car je devais m'adapter à une architecture existante.

Je fini ce projet avec de nouvelles connaissances dans le protocole HTTP avec les requêtes multi-part, l'affichage de cartes à l'aide de Google Maps et l'automatisation de tests à l'aide de l'outil Postman.

J'ai également pu apprendre certains concepts de design d'api tel que le sideloading ou l'embedding que je n'ai malheureusement pas pu mettre en place dans le temps imparti. Cela reste néanmoins un élément très intéressant permettant de grandement simplifier certaines actions sur l'application qui semble intéressant à implémenter dans une suite du projet.

Je n'ai pas rencontré de problèmes lors de ce projet. La documentation fournie par Google sur l'API Maps ressemble énormément aux documentations Android, cela m'a peut-être aidé car j'ai eu l'occasion de travailler 5 mois sur Android.

## 6 Annexes

### 6.1 Liste des documents fournis

- Planification Initiale : PHILIBERT_PlanificationInitiale.pdf
- Planification Finale : PHILIBERT_PlanificationFinale.pdf
- Configuration Postman : PHILIBERT_ConfigPostman.pdf
- Résumé : PHILIBERT_Resume.pdf
- Cahier des charges : PHILIBERT_CahierDesCharges.pdf
- Fichier GPX : run-20181229T155040.gpx
    - Disponible sur Github (/Code/misc/samples/run-20181229T155040.gpx)
- Résultat des tests : Runscape API.alexandre.json
    - Disponible sur Github (/Code/misc/tests/Runscape API.alexandre.json)

### 6.2 Liens

- Github : https://github.com/AlexandrePhilibertCPNV/TPI_PHILIBERT
- Trello : https://trello.com/b/7nNzJwwd/tpiphilibert
- Tutoriels vidéos
    - Création d'une activité avec GPX : https://youtu.be/F5W4fJvof_o
    - Création d'une activité sans GPX : https://youtu.be/3t5X0BVz684
    - Visualisation des activités (Dashboard) : https://youtu.be/nu8iXkzSPt8

### 6.3 Sources

HTTP POST multipart/form-data :

https://developer.mozilla.org/fr/docs/Web/HTTP/M%C3%A9thode/POST

Utilisation des objets FormData :

https://developer.mozilla.org/fr/docs/Web/API/FormData/Utilisation_objets_FormData

https://developer.mozilla.org/fr/docs/Web/API/FormData/append

Liste communes Suisse :

https://www.bfs.admin.ch/bfs/fr/home/bases-statistiques/repertoire-officiel-communes-suisse.assetdetail.6986904.html

Sélection d'une ligne sur N MySQL :

https://stackoverflow.com/questions/858746/how-do-you-select-every-n-th-row-from-mysql

datetime-local :

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/datetime-local#Browser_compatibility

Caching static files in memory :

https://stackoverflow.com/questions/32154656/does-express-static-cache-files-in-the-memory

multipart/form-data dans une requête XHR :

https://stackoverflow.com/questions/9395911/send-a-file-as-multipart-through-xmlhttprequest

OpenLayer API Open Street Map :

https://wiki.openstreetmap.org/wiki/Openlayers_Track_example

REST links, sideloading, embedding :

https://phauer.com/2015/restful-api-design-best-practices/

Google Maps Polyline API :

https://developers.google.com/maps/documentation/javascript/examples/polyline-simple

Postman tests :

https://blog.getpostman.com/2014/01/27/extracting-data-from-responses-and-chaining-requests/

Longueur maximale d'une adresse email :

https://www.ietf.org/rfc/rfc2821.txt