|  |
| --- |
| API RESTful Runscape |

Table des matières

[1 Analyse préliminaire 3](#_Toc1456969)

[*1.1* Introduction 3](#_Toc1456970)

[1.2 Objectifs 3](#_Toc1456971)

[*1.3* Planification initiale 5](#_Toc1456972)

[2 Analyse 5](#_Toc1456973)

[2.1 Définition de l’audience 5](#_Toc1456974)

[2.2 Analyse Concurrentielle 6](#_Toc1456975)

[2.3 Analyse des architectures 7](#_Toc1456976)

[2.4 Étude de faisabilité 8](#_Toc1456977)

[2.5 MCD 8](#_Toc1456978)

[2.6 Stratégie de test 9](#_Toc1456979)

[2.7 Risques techniques 9](#_Toc1456980)

[2.8 Budget initial 9](#_Toc1456981)

[2.9 Planification détaillée 10](#_Toc1456982)

[2.10 Historique 10](#_Toc1456983)

[3 Conception 11](#_Toc1456984)

[3.1 Analyse de l’environnement 11](#_Toc1456985)

[3.2 Ressources de l’API 11](#_Toc1456986)

[3.3 Endpoints de l’API 12](#_Toc1456987)

[3.4 Dossier de conception 12](#_Toc1456988)

[3.5 Modèle logique des données 13](#_Toc1456989)

[3.6 Diagrammes de flux 15](#_Toc1456990)

[3.7 Cas d’utilisation 17](#_Toc1456991)

[4 Réalisation 19](#_Toc1456992)

[4.1 Configuration serveur 19](#_Toc1456993)

[4.2 Dossier de réalisation 20](#_Toc1456994)

[*4.3* Description des tests effectués 21](#_Toc1456995)

[*4.4* Erreurs restantes 21](#_Toc1456996)

[*4.5* Liste des documents fournis 21](#_Toc1456997)

[5 Conclusions 21](#_Toc1456998)

[6 Annexes 22](#_Toc1456999)

[*6.1* Résumé du rapport du TPI / version succincte de la documentation 22](#_Toc1457000)

[*6.2* Sources – Bibliographie 22](#_Toc1457001)

[*6.3* Journal de travail 22](#_Toc1457002)

[*6.4* Manuel d'Installation 22](#_Toc1457003)

[*6.5* Manuel d'Utilisation 22](#_Toc1457004)

[*6.6* Archives du projet 22](#_Toc1457005)

# Analyse préliminaire

## Introduction

Le projet consiste à réaliser une API Web permettant à une personne de se connecter, se déconnecter, gérer ses activités physiques (ajouter de nouvelles activités, les consulter, les modifier et les supprimer). Cette API ne possède pas d’interface graphique permettant à un utilisateur de communiquer plus simplement avec celle-ci.

Un deuxième projet sera entrepris par la suite pour permettre une interaction par interface graphique avec cette API, ce deuxième projet consistera à utiliser l’API dans une application mobile, des facteurs de perte de synchronisation seront à prendre en compte lors de la conception de cette API.

Plusieurs modules ont été réalisé, dans le cadre du CPNV, et me permettrons de mener à bien ce projet :

* ICT-100 (Distinguer, préparé et évaluer des données)
* ICT-104 (Implémenter un modèle de données)
* ICT-105 (Traiter une base de données SQL)
* ICT-306 (Réaliser un petit projet informatique)
* MA-07 (Linux)

## Objectifs

### Mettre en œuvre une base de données pour les activités sportives

L’API doit être en mesure de manipuler des données modélisant des activités sportives, mettant à disposition de l’utilisateur les informations suivantes :

* La date
* L’heure de départ
* Le type d’activités (vélo, natation, course à pied)
* La durée de l’effort
* La distance parcourue
* Le parcours (une représentation graphique sera attendue sur la partie cliente)
* Les dénivelés positifs et négatifs
* La vitesse moyenne

### Web Service en JSON

Le back-end devra avoir la capacité de dialoguer avec un composant client qui effectuera les actions suivantes :

* Enregistrer, modifier, supprimer une activité
* Consulter et analyser les activités sportives sur une période donnée (moyenne, nombre de kilomètres parcourus par sport)
* Importer des données en provenance de GPS (format .gpx considéré en priorité)
* S’interfacer[[1]](#footnote-1) avec au moins un acteur majeur de gestion d’activité tels que Garmin Connect, Tom tom

### Sécurisation du back-end

Le back-end devra implémenter les sécurités suivantes :

* N’être accessible que via le protocole HTTPS (self-signed certificate)
* Aucun mot de passe ne doit être présent dans la base de données ni dans les logs applicatifs[[2]](#footnote-2)
* Le code applicatif ne doit pas avoir la capacité de supprimer (ni DELETE, ni DROP) des données

### Authentification des utilisateurs

Le back-end devra implémenter les mécanismes d’authentification suivantes :

* Soit pouvoir se connecter à l’aide d’un service « 2-step verification »
* Soit s’appuyer sur les APIs de Facebook

## Planification initiale



# Analyse

## Définition de l’audience

Les personnes ciblées par l’application sont les personnes à la recherche d’une application leur permettant d’enregistrer les activités sportives qu’ils réalisent. L’application permet à n’importe quelle personne disposant d’un téléphone portable Android[[3]](#footnote-3). L’audience est donc composée de personne faisant du sport ou voulant commencé à en faire.

## Analyse Concurrentielle

### Définition des concurrents

Les concurrents qui ont été retenus pour cette analyse proposent tous des applications mobile Android et iOS, deux des trois concurrents possèdes plus de 10 millions de téléchargements de leur application sur le Play Store de Google, le troisième concurrent possède 1 million de téléchargement. Les applications comparées sont, par conséquent des applications utilisés par énormément de personnes et reconnues pour leur maitrise du métier.

Les éléments analysés ne prennent pas en compte l’interface graphique de l’application, il ne s’agit ici que de fonctionnalités brutes proposée par l’application.

Les 3 applications analysées sont :

* Garmin Connect
* Tom Tom Sports
* Adidas Runtastic

### Analyse des fonctionnalités proposées

Le tableau ci-dessous contient une liste non-exhaustive des fonctionnalités proposées par les applications analysées. Le but est de définir quelles fonctionnalités semblent indispensables et quelles fonctionnalités semblent superflues.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Garmin Connect | TomTom Sports | Adidas Runtastic |
| Changements unité (miles, mètres, kg, lb) | X | X | X |
| Distance parcourue | X | X | X |
| Ajout manuel d’activités | X |  | X |
| Ajout d’un type d’activité | X |  |  |
| Calories brulées pendant l’effort | X |  | X |
| Personnalisation de la longueur de foulée | X |  |  |
| Record personnels | X |  | X |
| Interactions sociales intégrées | X |  | X |
| Abonnement mensuel permettant d’ajouter des fonctionnalités |  |  | X |
| Visualisation du parcours effectué | X |  | X |
| Altitude et dénivelé de l’activité | X |  | X |
| Défis ou objectifs | X | X | X |
| Gestion d’équipements (chaussures, …) | X |  | X |
| Connection à des services tierces | X | X | X |
| Délégation de la connexion (ex : par Google) |  |  | X |

## Analyse des architectures

Plusieurs architectures d’API Web existent, les 2 principales étant le protocole SOAP et l’architecture REST.

|  |  |
| --- | --- |
|  | Avantages |
| REST | Supporte d’autres format que l’XML (JSON par exemple) |
| Le serveur n’enregistre pas l’état du client, ce qui permet de distribué les charges sur plusieurs serveurs |
| Plus facile à intégrer que le SOAP |
| Plus utilisé que SOAP dans le contexte des Web Services |
| SOAP | Peut amener à une meilleure sécurité selon l’implémentation (signature par certificat XML par exemple) |
| Standardisé |
|  |
|  |

|  |  |
| --- | --- |
|  | Désavantages |
| REST | Architecture étant très stricte |
|  |
|  |
| SOAP | Ne supporte pas le JSON |
| Plus difficile à implémenter |
|  |

L’avantage d’utilisé l’architecture REST et de s’imposer un ensemble de contraintes pour faciliter l’évolutivité du système dans le cas où l’API venait à être distribuée sur plusieurs serveur par exemple.

Le SOAP n’étant plus autant utilisé que le REST et ne permettant pas l’’utilisation du JSON, j’ai choisi l’architecture RESTful pour mon API.

## Étude de faisabilité

### Faisabilité système

Ayant déjà réalisé l’installation d’un serveur NodeJS et d’une base de données MySQL sur un Raspberry Pi en dehors de ce projet, aucun problème ne devrait survenir lors d’une nouvelle installation.

### Faisabilité logicielle

Des modules NodeJS seront utilisés lors de la réalisation du Web Service, voici une liste de plusieurs modules qui seront sûrement utilisés :

* fs (accès au système de fichier)
* https (serveur HTTPS)
* url (gestion des URLs)
* querystring (gestion des querystring dans les URLs)
* util (utilitaires)
* jsonwebtoken (gestion des tokens de connection)

Ces modules sont, soit des modules de bases, soit des modules très connus dans l’écosystème NodeJS, il est donc très facile de se documenter sur ces modules. Aucun problème de faisabilité de devrait être rencontré lors de la réalisation du Web Service.

## MCD



## Stratégie de test

### Étendue des tests

Les tests seront effectués sur tous les endpoints de l’API. Les tests seront réalisés à l’aide du logiciel Postman.

### Testeurs

* Alexandre Philibert : [alexandre.philibert@cpnv.ch](mailto:alexandre.philibert@cpnv.ch)

### Type de tests

* Unitaire
* Intégration
* Fonctionnels

### Liste des tests

* Vérification du fonctionnement des use cases (décrit ci-dessus)
* Vérification du fonctionnement des scénarios (décrit ci-dessus)

### Données de test à prévoir

Aucunes données de test ne sont prévues

## Risques techniques

|  |  |  |
| --- | --- | --- |
| Risque | Raison | Solutions ou actions |
| Authentification stateless | Jamais utilisé auparavant | Tentative d’acquérir les connaissances nécessaire, sinon changement vers une authentification non-stateless. |

## Budget initial

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

## Planification détaillée



## Historique

# Conception

## Analyse de l’environnement

J’ai choisi de travailler avec le logiciel Notepad++, j’utilise l’auto complétion proposée par Notepad++ et les plugins pour d’autres fonctionnalités comme le FTP.

Lorsque je rencontre des problèmes, je peux utiliser la console de développement de Chrome pour accéder à un débuggeur NodeJS.

## Ressources de l’API

|  |  |
| --- | --- |
| Ressource | Utilisation |
| User | Stocke les données d’un utilisateur |
| Activity | Stocke une activité sportive réalisé par un utilisateur |
| Subscription | Stocke l’abonnement d’un utilisateur (date de début, date de fin, …) |
| Position | Stocke une position GPS à un temps donné |

## Endpoints de l’API

|  |  |  |
| --- | --- | --- |
| Fonction | Endpoints | Verbe HTTP |
| Création compte | /user | POST |
| Modification compte | /user/{userid} | PUT |
| Connexion | /session | GET |
| Création d’activité | /user/{userid}/activity/ | POST |
| Lecture d’activité | activity/{activityid} | GET |
| Modification d’activité | activity/{activityid} | PUT |
| Suppression d’activité | activity/{activityid} | DELETE |
| 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 |
| Ajout abonnement | /subscription/ | POST |
| Modification abonnement | /subscription/{subid} | PUT |
| Lecture abonnement | /user/{userid}/subscription | GET |

## Dossier de conception

### Choix du matériel physique

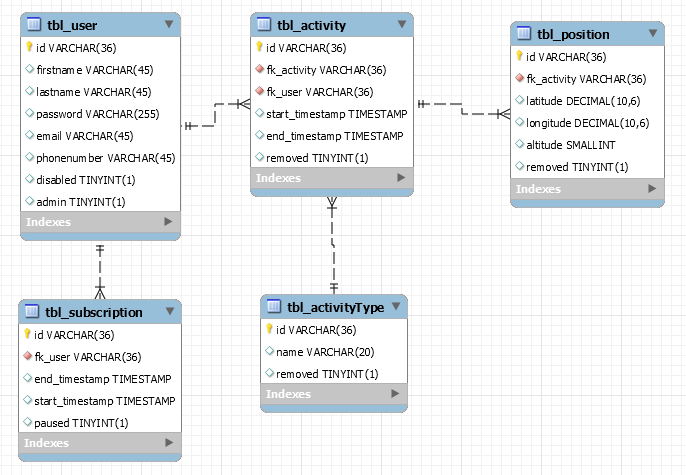
Un Raspberry Pi (modèle 3b) sera utilisé pour hébergé l’API, j’ai choisi d’utilisé un Raspberry Pi car j’ai déjà effectuer l’installation de NodeJS et MySQL dans d’autres projets. Il reste très simple de migrer l’API vers un serveur plus performant lorsque le nombres de requêtes deviendra trop important, par la similarité de la distribution linux Raspbian à Debian.

### Version des logiciels

|  |  |
| --- | --- |
| NodeJS | v8.11.1 |
| MySQL | ver 15.1 Distrib 10.1.37-MariaDB |
| Raspbian | stretch 9.4 |

## Modèle logique des données

Le modèle présenté dans cette section n’est pas définitif, il permet d’avoir une idée globale de l’architecture de la base de données. Des modifications peuvent être apportées lorsque l’architecture ne s’avère plus adaptée.



Vous trouverez, ci-dessous, une description des champs pouvant contenir des informations spéciales. D’autres champs ne sont pas détaillé car le contenu de ceux-ci est basique

### User

|  |  |
| --- | --- |
| Champs | Utilisation |
| password | Stocke le hash du mot de passe |
| disabled | Définit si un compte utilisateur est activé ou non |
| admin | Définit si un compte utilisateur est administrateur ou non |

### Activity

|  |  |
| --- | --- |
| Champs | Utilisation |
| start\_timestamp | Timestamp de début d’activité |
| end\_timestamp | Timestmap de fin d’activité |
| removed | Définit si une activité a été supprimée |

### Position

|  |  |
| --- | --- |
| Champs | Utilisation |
| latitude | Stocke la latitude de l’enregistrement de position avec une précision d’environ 100mm |
| longitude | Stocke la longitude de l’enregistrement de position avec une précision d’environ 100mm |
| altitude | Stocke l’altitude de l’enregistrement de positon avec comme point de référence le niveau de la mer |
| removed | Définit si un enregistrement a été supprimer |

### Subscription

|  |  |
| --- | --- |
| Champs | Utilisation |
| end\_timestamp | Timestamp de fin d’abonnement |
| start\_timestamp | Timestamp de début d’abonnement |
| paused | Définit si l’abonnement est mis en pause |

### ActivityType

|  |  |
| --- | --- |
| Champs | Utilisation |
| Name | Nom du type d’activité |
| removed | Définit si le type d’activité a été supprimer |

## Diagrammes de flux





## Cas d’utilisation

### Diagramme des cas d’utilisation



### Cas d’utilisation visiteur

|  |  |  |
| --- | --- | --- |
| En tant que visiteur | Je veux créer un compte | Dans le but de devenir un membre |

|  |  |  |
| --- | --- | --- |
| Déroulement | Étape | Description |
| 1 | Utilisateur : Soumet une requête GET à "/user" |
| 2 | API : Valide la requête |
| 3 | API : retourne une réponse |
| Extensions | 2a | Format d'email invalide  API : Retourne une erreur de format d'email invalide |
| 2b | Format de mot de passe invalide  API : Retourne une erreur de format de mot de passe |
| 2c | Champs manquant dans la requête  API : Retourne une erreur de champs manquant |

# Réalisation

## Configuration serveur

### Utilisateurs Raspbian

|  |  |  |  |
| --- | --- | --- | --- |
| Nom d'utilisateur | Fonction | Accès SSH | Accès sudo |
| pi | Utilisateur par défaut de Raspbian, il n’a pas été supprimé car des fonctionnalités de Raspbian ont besoin de cet utilisateur. | Non | Oui |
| alexandre | Administrateur du serveur | Oui | Oui |

### Paquets installés

Plusieurs paquets ont été ajoutés, en plus de la configuration par défaut de Raspbian, vous trouverez la liste des paquets ajoutés et mis à jour ci-dessous :

|  |  |
| --- | --- |
| Nom du paquet | Version |
| openssh-server | OpenSSH\_7.4p1 Raspbian-10+deb9u5, OpenSSL 1.0.2q 20 Nov 2018 |
| ufw | 0.35 |
| fail2ban | Fail2Ban v0.9.6 |
| nodejs | v8.11.1 |
| npm | 1.4.21 |

### Configuration SSH

Le fichier de configuration SSH se trouve à l'emplacement "/etc/ssh/sshd\_config", les commandes modifiées sont :

|  |
| --- |
| ChallengeResponseAuthentication no  PasswordAuthentication no  DenyUsers pi  UsePAM no |

### Configuration firewall

Le firewall utilisé sur le serveur est ufw, par sa simplicité de configuration, voici la liste des ports ouverts :

|  |
| --- |
| To Action From  -- ------ ----  443 ALLOW Anywhere  80 ALLOW Anywhere  22 ALLOW Anywhere  443 (v6) ALLOW Anywhere (v6)  80 (v6) ALLOW Anywhere (v6)  22 (v6) ALLOW Anywhere (v6) |

Seulement les ports par défauts des protocoles HTTP, HTTPS et SSH sont autorisé en version IPv4 et IPv6.

### Configuration fail2ban

Un outil de prévention contre les intrusions à été installé sur le serveur, il permet de bannir les adresses IP après un nombre de tentatives de connexion trop importantes sur le port SSH par exemple. La configuration de fail2ban à été laissé par défaut, seulement la ligne suivantes du fichier "/etc/fail2ban/jail.local" sous la section [sshd] a été modifiée :

|  |
| --- |
| maxretry = 10 |

## Dossier de réalisation

*Décrire la réalisation "physique" de votre projet*

* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*

*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…*

## Description des tests effectués

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## Erreurs restantes

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

*Développez en tous cas les points suivants:*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

## Sources – Bibliographie

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)… Et de toutes les aides externes (noms)*

## Journal de travail

## 

## Manuel d'Installation

## Manuel d'Utilisation

## Archives du projet

*Media, … dans une fourre en plastique*

1. L’utilisateur doit être en mesure de synchroniser ses activités sportives avec un autre Web Service [↑](#footnote-ref-1)
2. Stocké sous forme de hash par exemple [↑](#footnote-ref-2)
3. Pour autant que le téléphone dispose d’un GPS, gyroscope et accéléromètre. [↑](#footnote-ref-3)