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

Table des matières

[1 Analyse préliminaire 3](#_Toc526619)

[*1.1* Introduction 3](#_Toc526620)

[*1.2* Objectifs 3](#_Toc526621)

[*1.3* Planification initiale 4](#_Toc526622)

[2 Analyse / Conception 4](#_Toc526623)

[2.1 Analyse des architectures 4](#_Toc526624)

[*2.2* Concept 6](#_Toc526625)

[2.3 MCD 6](#_Toc526626)

[2.4 Stratégie de test 7](#_Toc526627)

[2.4.1 Étendue des tests 7](#_Toc526628)

[2.4.2 Testeurs 7](#_Toc526629)

[2.4.3 Type de tests 7](#_Toc526630)

[2.4.4 Liste des tests 7](#_Toc526631)

[2.4.5 Données de test à prévoir 7](#_Toc526632)

[2.5 Risques techniques 8](#_Toc526633)

[*2.6* Planification détaillée 8](#_Toc526634)

[3 Conception 8](#_Toc526635)

[3.1 Analyse de l’environnement 8](#_Toc526636)

[3.2 Ressources de l’API 8](#_Toc526637)

[3.3 Endpoints de l’API 9](#_Toc526638)

[3.4 Dossier de conception 9](#_Toc526639)

[3.4.1 Choix du matériel physique 9](#_Toc526640)

[3.4.2 Version des logiciels 9](#_Toc526641)

[4 Réalisation 10](#_Toc526642)

[*4.1* Dossier de réalisation 10](#_Toc526643)

[*4.2* Description des tests effectués 10](#_Toc526644)

[*4.3* Erreurs restantes 10](#_Toc526645)

[*4.4* Liste des documents fournis 10](#_Toc526646)

[5 Conclusions 11](#_Toc526647)

[6 Annexes 12](#_Toc526648)

[*6.1* Résumé du rapport du TPI / version succincte de la documentation 12](#_Toc526649)

[*6.2* Sources – Bibliographie 12](#_Toc526650)

[*6.3* Journal de travail 12](#_Toc526651)

[*6.4* Manuel d'Installation 12](#_Toc526652)

[*6.5* Manuel d'Utilisation 12](#_Toc526653)

[*6.6* Archives du projet 12](#_Toc526654)

# 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

*Ce chapitre énumère les objectifs du projet. L'atteinte ou non de ceux-ci devra pouvoir être contrôlée à la fin du projet. Les objectifs pourront éventuellement être revus après l'analyse.*

*Ces éléments peuvent être repris des spécifications de départ.*

## Planification initiale

*Ce chapitre montre la planification du projet. Celui-ci peut être découpé en tâches qui seront planifiées. Il s'agit de la première planification du projet, celle-ci devra être revue après l'analyse. Cette planification sera présentée sous la forme d'un diagramme.*

*Ces éléments peuvent être repris des spécifications de départ.*

# Analyse / Conception

## 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 dans certains cas (certificats XML signé 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, j’ai choisi de base l’architecture de rendre mon API RESTful.

## Concept

*Le concept complet avec toutes ses annexes:*

*Par exemple :*

* *Multimédia: carte de site, maquettes papier, story board préliminaire, …*
* *Bases de données: interfaces graphiques, modèle conceptuel.*
* *Programmation: interfaces graphiques, maquettes, analyse fonctionnelle…*
* *…*

## MCD



## Analyse fonctionnelle

Les contraintes :

* Possibilité d’ajouter des

## 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, sinon changement vers une authentification non-stateless |

## Planification détaillée

*Révision de la planification initiale du projet :*

* *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*
* *partage des tâches en cas de travail à plusieurs.*

*Il s’agit en principe de la planification* ***définitive du projet****. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l’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

* User
* Activity

## Endpoints de l’API

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

## Dossier de conception

### Choix du matériel physique

Un Raspberry Pi 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 |

*Fournir tous les document de conception:*

* *le choix du matériel HW*
* *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
* *le choix des outils logiciels pour la réalisation et l'utilisation*
* *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
* *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme…*

***Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !***

# Réalisation

## 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 !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

*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*