**StellarMap**



# Table des matières

Figure 1 : Système solaire (freepik, -)

[1 Table des matières 2](#_Toc134531991)

[2 Analyse préliminaire 3](#_Toc134531992)

[2.1 Introduction 3](#_Toc134531993)

[2.2 Objectifs 3](#_Toc134531994)

[2.3 Planification initiale 5](#_Toc134531995)

[3 Analyse / Conception 6](#_Toc134531996)

[3.1 Concept 6](#_Toc134531997)

[3.2 Stratégie de test 6](#_Toc134531998)

[3.3 Risques techniques 7](#_Toc134531999)

[3.4 Planification 7](#_Toc134532000)

[3.5 Dossier de conception 7](#_Toc134532001)

[3.5.1 Logiciels / Framework utilisé : 7](#_Toc134532002)

[4 Réalisation 8](#_Toc134532003)

[4.1 Dossier de réalisation 8](#_Toc134532004)

[4.1.1 Classes 8](#_Toc134532005)

[4.1.2 Requête API 8](#_Toc134532006)

[4.2 Répertoires 12](#_Toc134532007)

[4.3 Description des tests effectués 13](#_Toc134532008)

[4.4 Erreurs restantes 13](#_Toc134532009)

[4.5 Liste des documents fournis 13](#_Toc134532010)

[5 Conclusions 14](#_Toc134532011)

[6 Bibliographie 15](#_Toc134532012)

[7 Table des illustrations 15](#_Toc134532013)

[8 Lexique 15](#_Toc134532014)

[9 Annexes 16](#_Toc134532015)

[9.1 Planification initiale 16](#_Toc134532016)

[18](#_Toc134532017)

[9.2 Résumé du rapport du TPI / version succincte de la documentation 20](#_Toc134532018)

[9.3 Journal de travail 20](#_Toc134532019)

[9.4 Archives du projet 20](#_Toc134532020)

# Analyse préliminaire

## Introduction

Ce projet est réalisé dans le cadre du au travail pratique individuel (TPI) qui s’effectue lors de la dernière année de CFC en informatique.

Ce travail s’effectue sur une période de réalisation de 90 heures, entre le 2 mai de 8h50 au 2 juin à 15h20.

Le sujet est une carte 3D interactive du système solaire, il a été choisi à la suite de la proposition de ce sujet par le candidat.

## Objectifs

L’objectif de ce projet est de crée une carte interactive du système solaire sur le quel il sera possible de voir les 8 planètes et leurs lunes ainsi que le soleil. Il sera possible de tourner autour du soleil et d’observer les planètes sous un autre angle. Une description des planètes devra s’afficher lorsqu’un utilisateur clique sur celui-ci, de plus il sera possible d’accélérer la vitesse de déplacement des planètes.

Sept objectifs spécifiques sont à atteindre :

1. La carte s’affiche avec toutes les huit planètes.
2. L’utilisateur peut naviguer dans le système solaire.
3. Ergonomie et facilité d’utilisation du produit (Bastien et Scapin).
4. Les informations des différentes planètes s’affichent quand on clique dessus.
5. Le site est « responsive » et peut être utilisé depuis un smartphone ou une tablette.
6. L’utilisateur peut modifier la vitesse de déplacement des planètes.
7. Les angles de vue du système peuvent être déterminé par l’utilisateur.

Tout au long de mon travail je vais me conformer aux critères d’évaluation établis par le canton de Vaud (Schwab, 2018).Cahier des charges

## Planification initiale

La planification initiale se découpe en cinq sprints découpés sur cinq semaines.

Le détail de la planification est disponible à [la section 9.1](#_Planification_initiale).

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure 2 : Planification initiale du projet

# Analyse / Conception

## Concept

Ce site est conçu pour afficher un système solaire en 3D, le visiteur pourra tourner autour de la carte et ainsi voir les planètes sous d’autres angles, il aura aussi la possibilité d’accélérer le temps afin de voir le déplacement des planètes à des vitesses différentes.

Ce système se verra affiché les 8 planètes du système solaire, leurs lunes ainsi que les astéroïdes à proximité de la planète terre.

Le projet sera hébergé sur swisscenter, les liens des accès au projet sont les suivantes :

Code source : <https://github.com/Juillet-Mikael/TPI>

Planification du projet : <https://icescrum.cpnv.ch/p/TPIJUILLET/#/project>

Documentation :

<https://github.com/Juillet-Mikael/TPI/blob/main/documents/documentation.docx>

Journal de travail :

<https://github.com/Juillet-Mikael/TPI/blob/main/documents/journaux.xlsm>

#### Requêtes

Il est prévu d’utiliser deux API de la Nasa, l’api « [horizon view](https://ssd.jpl.nasa.gov/horizons/app.html#/)» permet de récupérer des informations précises sur les objets spatiaux dans notre système solaire. Horizon sera utilisé pour récupérer toutes les informations nécessaires au placement, et à la définition des planètes comme le volume, la densité, la position précise actuel, la température etc.

La deuxième API est « Near Earth Object » qui permet de récupérer la liste des objets proche de la terre à un temps donnée, elle sera utilisée pour placer approximativement les astéroïdes sur la carte car aucune donnée de placement précise ne peut être récupérer via cette api.

#### Maquettes

#### Diagramme de classe

Une image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement

#### Diagramme de séquences

## Stratégie de test

Je vais essentiellement être effectuer manuellement pour la partie visuelle, j’ai choisi de faire cela car je ne dispose que de peu de temps pour crée mon projet de plus j’ai une très faible connaissance de la création de tests en Javascript.

Une exception sera faite pour ce qui est de la partie API en semaine 2, j’effectuerais des tests unitaires sur le filtrage des donnée et la réception des données.

J’effectuerai à chaque fin de sprint une série de tests dès l’implémentation de l’api en semaine numéro 2.

Les tests en rapport à la vue comme pour le déplacement des planètes seront des tests fonctionnels. Ces tests ont pour objectif de s’affurer du bon déroulement de l’affichage du système solaire.

## Risques techniques

Les risques techniques sont mon manque de connaissance à l’utilisation de Three.js (three.js, 2023) et Ajax (W3schools, -), malgré de solides base acquises grâce à la préparation au TPI, j’ai tout encore besoin de beaucoup me référer aux documentations.

## Planification

## Dossier de conception

### Logiciels / Framework utilisé :

|  |  |  |
| --- | --- | --- |
| **Nom** | **Version** | **Utilisation** |
| Visual Studio Code (visualstudio, 2023) | 1.74.3 | Editeur de code |
| Balsamiq Wireframe (balsamiq, 2023) | 4.6.5 | Wireframe |
| Figma (figma, 2023) | - | Mockup |
| HTML, CSS | Html 5, CSS 3 | Mise en page |
| Vite.js (vitejs, 2023) | 4.1.1 | Frontend Tooling |
| Three.js (WebGL) (three.js, 2023) | 0.152.2 | Rendu 3D |
| Ajax (W3schools, -) | - | Requêtes |
| Moment (momentjs, 2023) | 2.29.4 | Gestion des dates |
| Vitest (vitest, 2021) | 0.31.0 | Tests |

# Réalisation

## Dossier de réalisation

### Requête API

Les deux requêtes sont placées dans le dossier model dans un fichier appelé requests.js.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

#### Variables d’environnement

Un fichier .env a été crée pour stocker les données sensibles tel que la clé api.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

#### Horizon system

Url de requête : <https://ssd.jpl.nasa.gov/api/horizons.api>

Certains paramètres sont absolument obligatoires en voici la liste (Nasa, 2022) :

* **Format**, sera retourné en Json.
* **Command**, représente l’id de l’objet ciblé (ex. terre = 399).
* **Objet\_data**, représente les données de l’objet tel que la période de rotation.
* **Make\_Ephem**, représente les données de placement des planètes.
* **Start\_time**, la date de départ des données en format années-mois-jour.
* **End\_time**, la date de fin des données en format années-mois-jour.
* **Step\_size**, la durée séparant les informations de placement de l’objet ciblé.

Cette requête engendre une erreur corse c’est pourquoi j’ai ajouté dans mon header un Origin mais l’erreur ne se résout pas. (Braiam, 2017) (Simplified, 2021) Je ne parviens pas à récupérer les données pourtant le lien est valide et retourne des données.

Erreur Corse :

|  |
| --- |
| No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled. |

J’ai pu récupérer les données depuis Postman.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure 3: Données retournées via Postman.

Mais malgré un retour de requête avec un code d’état de 200 je n’avais aucune donnée.

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Figure 4 : Etat de la requête Horizon API sans données.



Figure 5 : Erreur d'affichage des données reçues.

Après en avoir discuté avec mon chef de projet Monsieur Benzonana, nous avons décidé de l’envoi d’un e-mail au support de la Nasa, puis j’exporterai les données dans un fichier json que je traiterai ainsi afin de ne pas perdre du temps. Si le temps le permet je reviendrais éventuellement sur cette partie en fin de projet, en plus de ceci j’ai créé une [issue](https://github.com/Juillet-Mikael/TPI/issues) sur GitHub afin de maintenir le projet.

La réponse de la Nasa a été expéditive :

« Les API de la NASA ne peuvent être intégrées à aucun site Web non-NASA. Donc, je crains que vous n'ayez besoin de trouver une autre méthode. C'est la politique de la NASA que nous sommes tenus de suivre. Vous pourrez peut-être appeler l'API à partir d'un script en dehors de votre serveur Web, puis déterminer comment connecter votre application Web à ce script. »

Pour donner suite à cette réponse nous avons décidé mon chef de projet et moi de garder l’exportation des données dans un fichier json.

##### Récupération des données

Les données reçues sont sous forme de String, je dois récupérer des valeurs précise dans ces tableaux c’est pourquoi j’utilise des expression rationnelles (regex). Celle-ci vont me permettre de récupérer des valeurs en fonction d’une expression régulière dans mon fichier. (mozilla, 2023)

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 6 : Données récupérée après suppression des espaces.

Pour formater le texte, il faut en premier définir l’expression régulière de l’élément que l’on souhaite exporter via des regex. Puis faire une reconnaissance dans le texte de l’élément spécifier par le regex et finalement récupérer la bonne valeur.

Pour crée les regex j’ai utilisé chatGPT pour générer les regex en fonction des données, étant donné que chaque planètes as des données différentes il était plus rapide de demander à chatGPT de générer les regex en fonction des données voulus. (OpenAI, 2023)

Comme exemple le code ci-dessous qui cherche trois valeurs x=, y= et z= ils sont suivis de chiffres et d’une combinaisons E+- accompagné de chiffres.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 7 : Exemple de l'utilisation de regex

#### Near Earth Objects

Url de requête : <https://api.nasa.gov/neo/rest/v1/feed>

La requête near earth object est simple, j’ai fetch les données en passant une date de début et une date de fin ainsi que la clé api définie dans le fichier env.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 8 : Fonction de création fetch des objets proches.

Une image contenant texte, capture d’écran, menu, Police

Description générée automatiquement

Figure 9 : Données récupérées via a l'api NEO.

#### Favicon

Une image contenant objet astronomique, planète, sphère, soleil

Description générée automatiquementJ’ai choisi comme favicon une image de soleil du site png art (pngarts, -).

Figure 10 : Favicon

## Répertoires

Code source : <https://github.com/Juillet-Mikael/TPI>

Planification du projet : <https://icescrum.cpnv.ch/p/TPIJUILLET/#/project>

Documentation se situe aussi dans un dossier nommé doc au sein du projet Git.

**Architecture des documents :**

* TPI
  + documents
    - journals
    - documentation
    - planification initiale
    - diagrams
      * diagrame de classe
      * diagrame de scéquence
    - instruction
  + src
    - model
    - view
    - controller

## Description des tests effectués

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Erreurs restantes

## Liste des documents fournis

# Conclusions

# Bibliographie

balsamiq. (2023, - -). *balsamiq*. Récupéré sur balsamiq: https://balsamiq.com/

Braiam. (2017, Mai 09). *No 'Access-Control-Allow-Origin' header is present on the requested resource—when trying to get data from a REST API*. Récupéré sur stackoverflow: https://stackoverflow.com/questions/43871637/no-access-control-allow-origin-header-is-present-on-the-requested-resource-whe

day.js. (2023, - -). *day.js*. Récupéré sur day.js: https://day.js.org/

figma. (2023, - -). *figma*. Récupéré sur figma: https://www.figma.com/

freepik. (-, - -). Vecteur gratuit système de système solaire classique avec deisgn plat. *Vecteur gratuit système de système solaire classique avec deisgn plat*. -, -, -. Récupéré sur https://fr.freepik.com/

momentjs. (2023, - -). *momentjs*. Récupéré sur momentjs: https://momentjs.com/

mozilla. (2023, Mai 05). *Regular expressions*. Récupéré sur developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular\_expressions

Nasa. (2022, Septembre 1). *Horizons API*. Récupéré sur ssd-api: https://ssd-api.jpl.nasa.gov/doc/horizons.html

pinia. (2023, - -). *pinia*. Récupéré sur pinia: https://pinia.vuejs.org/

Simplified, W. D. (2021, Mai 22). *Apprenez CORS en 6 minutes*. Récupéré sur Youtube: https://www.youtube.com/watch?v=PNtFSVU-YTI

three.js. (2023, - -). *three.js*. Récupéré sur three.js: https://threejs.org/

visualstudio. (2023, - -). *visualstudio*. Récupéré sur visualstudio: https://code.visualstudio.com/

vitejs. (2023, - -). *vitejs*. Récupéré sur vite: https://vitejs.dev/

vuejs. (2023, - -). *vuejs*. Récupéré sur vuejs: https://vuejs.org/guide/components/provide-inject.html#prop-drilling

W3schools. (-, - -). *AJAX Introduction*. Récupéré sur W3schools: https://www.w3schools.com/js/js\_ajax\_intro.asp

# Table des illustrations

[Figure 1 : Système solaire (freepik, -) 1](file:///C:\Users\Mikael.JUILLET\Desktop\TPI\documents\documentation.docx#_Toc134532038)

[Figure 2 : Etat de la requête Horizon API sans données. 9](#_Toc134532039)

[Figure 3 : Données récupérée après suppression des espaces. 10](#_Toc134532040)

[Figure 4 : Exemple de l'utilisation de regex 11](#_Toc134532041)

# Lexique

R

regex

Regex ou expressions rationnelles sont des motifs de combinaisons de caractères au sein de chaînes d'un texte. · 12

# Annexes

## Une image contenant texte, capture d’écran, nombre, Police Description générée automatiquementUne image contenant texte, logiciel, nombre, Page web Description générée automatiquementPlanification initiale

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementUne image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

## 

Une image contenant texte, diagramme, Caractère coloré, ligne

Description générée automatiquement

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

## Journal de travail

## Archives du projet

*Media, … dans une fourre en plastique*