StellarMap

Table des matières

2 Analyse préliminaire

2.1 Introduction

Ce projet est réalisé dans le cadre du 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 14h05. Il sera suivi d'une semaine de préparation à la présentation dans lequel le candidat se prépare à présenter son projet devant deux experts et le chef de projet. Cette présentation s'effectuera le 12 juin à 14h50 en salle C315 au centre professionnel de Sainte-Croix.

Lors de ce projet deux experts Monsieur Montemayor et Monsieur Sahli suivront les avancement du projet.

Le sujet sélectionné est la réalisation d'une carte 3D interactive du système solaire, il a été choisi à la suite de la proposition de ce sujet par le candidat et validation du chef de projet et des experts.

2.2 Objectifs

L'objectif de ce projet est de crée une carte interactive du système solaire sur laquelle les utilisateurs pourront sans autre naviguer dans le système et découvrir les différentes planètes et leurs satellites. 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)

2.3 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.

La planification initiale sous forme de méthodologie Waterfall (Wikipedia, 22), j'ai choisi ce format car c'est cette forme convient le mieux pour la planification initiale de projet, étant donné que ce projet dois respecter des dates de rendu précise il est possible de définir à l'avance le travail à effectuer. Cependant le travail se déroulera en format Agile via l'utilisation de sprint de 7 jours, le format Agile permet de modifier si nécessaire en cour de projet les feature demandée, le logiciel Icescrum sera utilisé pour cette planification.

Sur l'image ci-dessous, le projet est près découper en semaine et les tâche générale du projet sont près définies. En annexe se trouve au point 9.1 les détails de cette planification initiale avec les tâches et sous tâches détaillées.

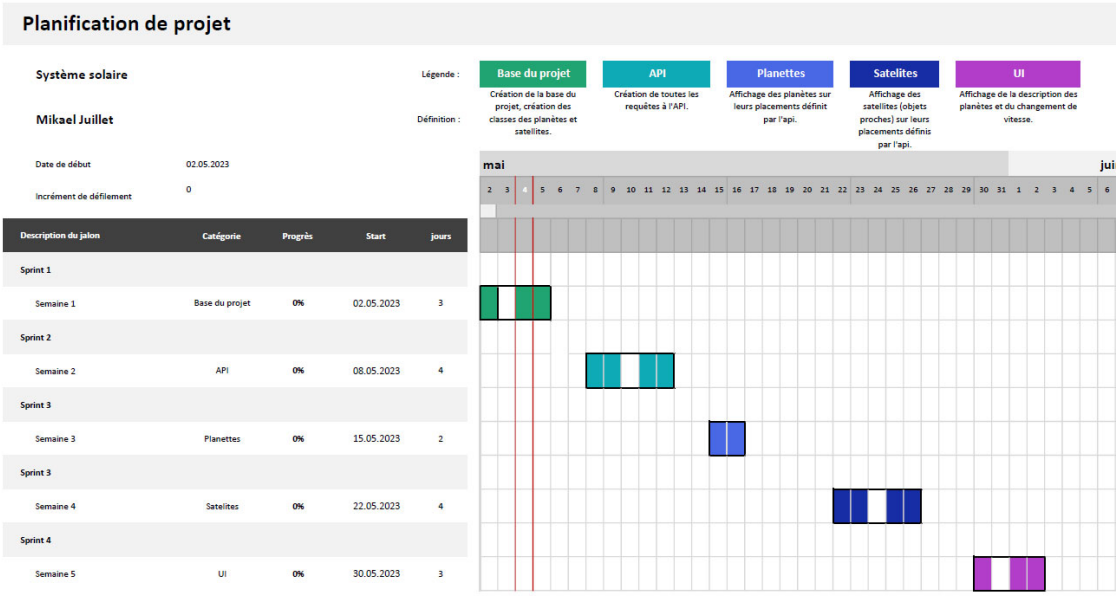


Figure 2 : Planification initiale du projet.

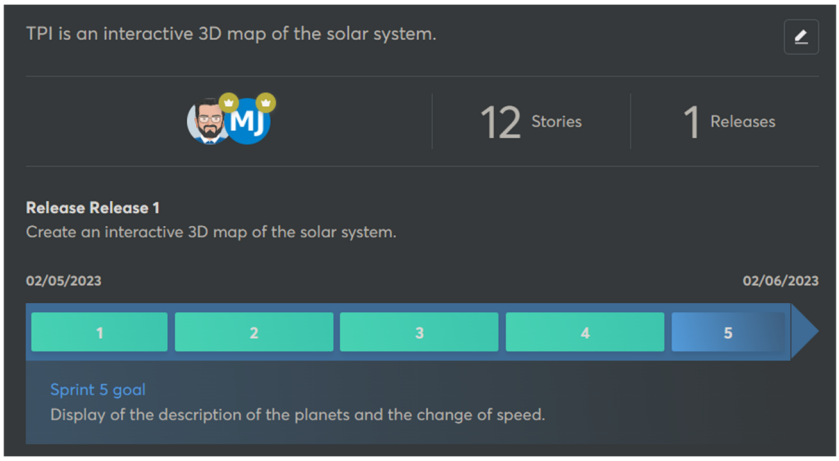


Figure 3 : Format Agile sur IceScrum du projet.

3 Analyse / Conception

3.1 Concept

Ce site est conçu pour afficher un système solaire en 3D, le visiteur peut tourner autour de la carte et ainsi voir les planètes sous d'autres angles, il a 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 ainsi que les astéroïdes à proximité de la planète terre.

Le projet sera hébergé sur la plateforme Swisscenter, les liens relatifs au projet sont les suivantes :

Site web: https://stellarmap.mycpnv.ch/
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

3.1.1 Requêtes

Il est prévu d'utiliser deux API de la Nasa, l'API « horizon view » 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.

3.1.2 Wireframe

Les Wireframe ont été créés via le logiciel Balsamiq, les maquettes dispose de trois boutons (actuel, 7j/s, 30j/s) ainsi qu'une zone de description des planètes. Le format vertical mobile affichera un message demandant aux utilisateurs de tourner leurs téléphones afin de pouvoir utiliser le site et finalement un écran de chargement s'affichera lors de l'ouverture de l'app en attendant le chargement des différents composants.

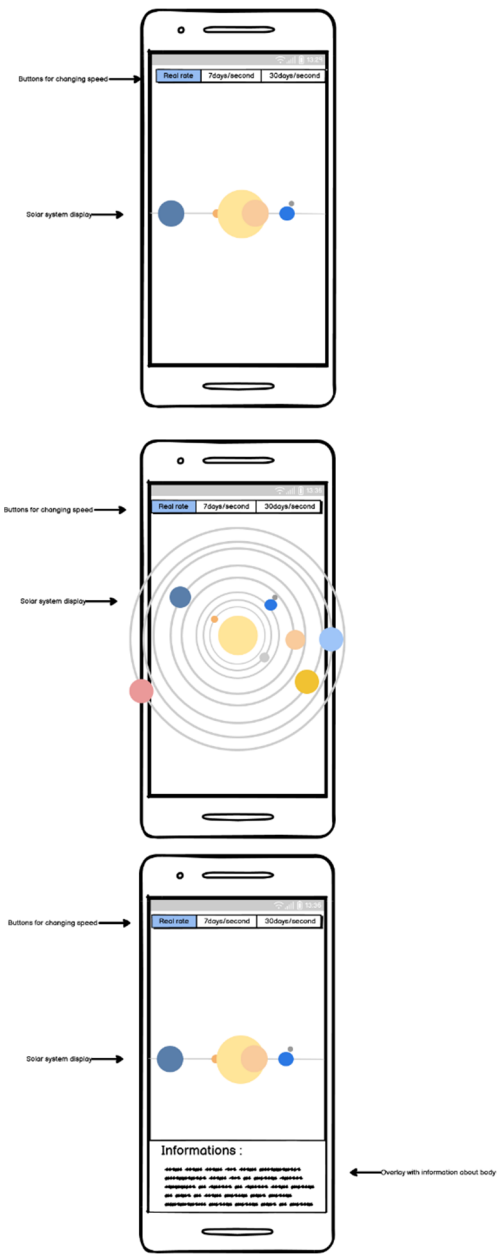


Figure 5 : Wireframe format mobile vertical.

Figure 4 : Wireframe format desktop.

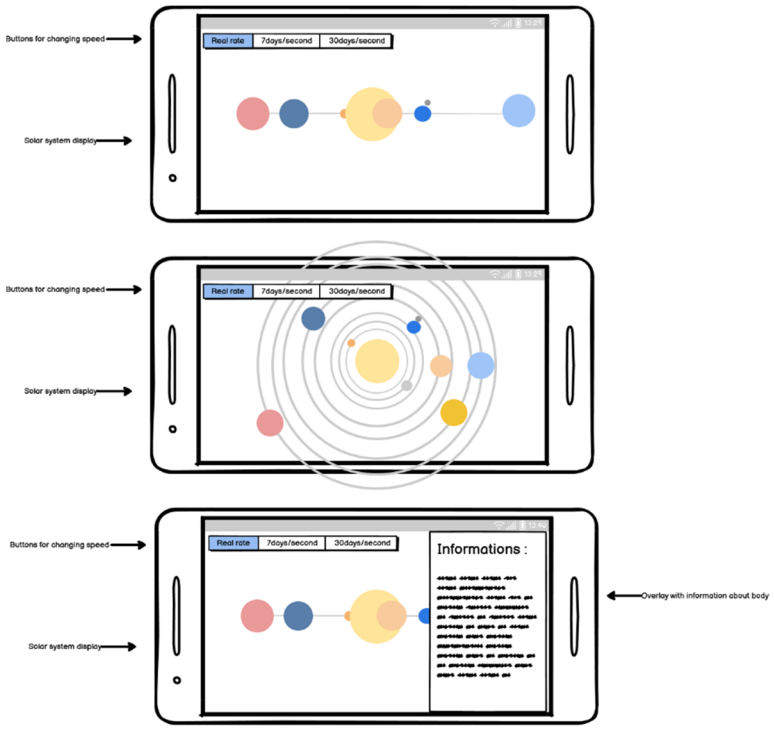


Figure 6 : Wireframe format mobile horizontal.

3.1.3 Mockups

Les Mockups ont été créé à la suite des wireframes depuis le logiciel Figma.

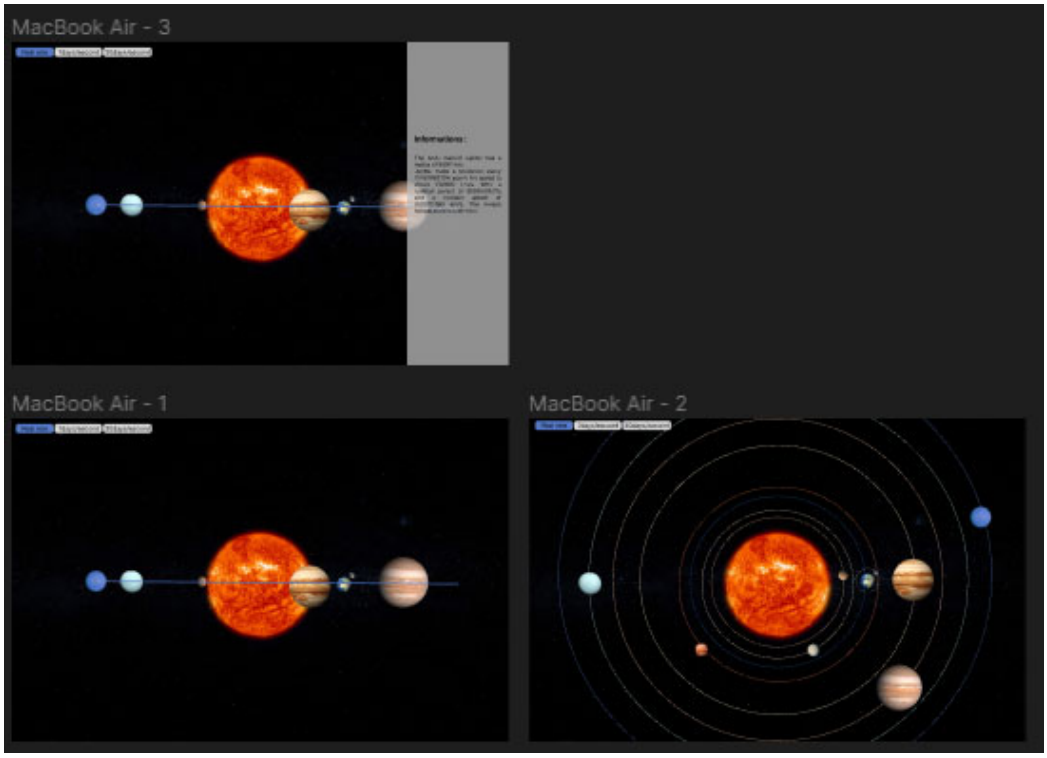


Figure 7 : Mockups format desktop

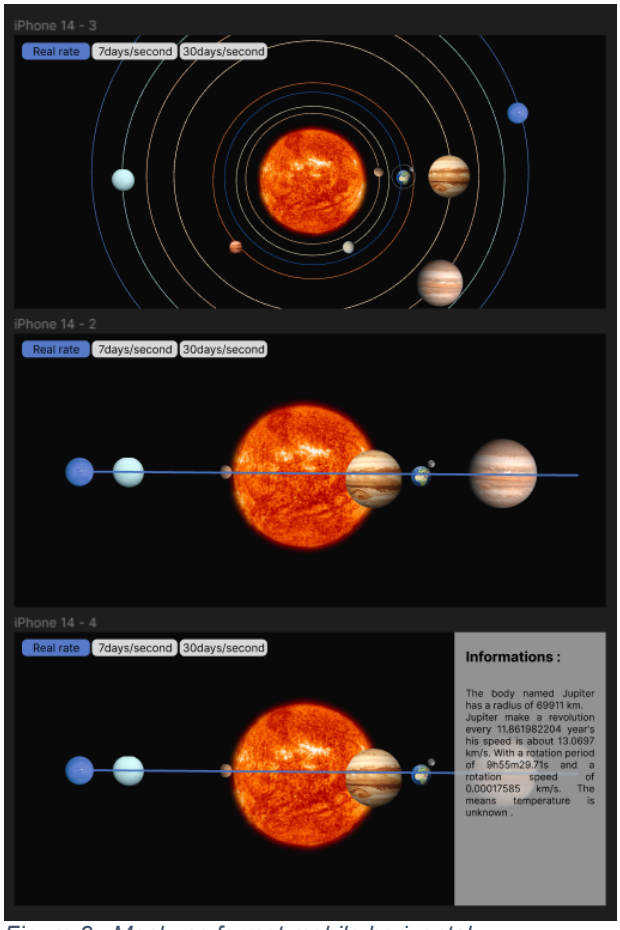


Figure 8 : Mockups format mobile horizontal.

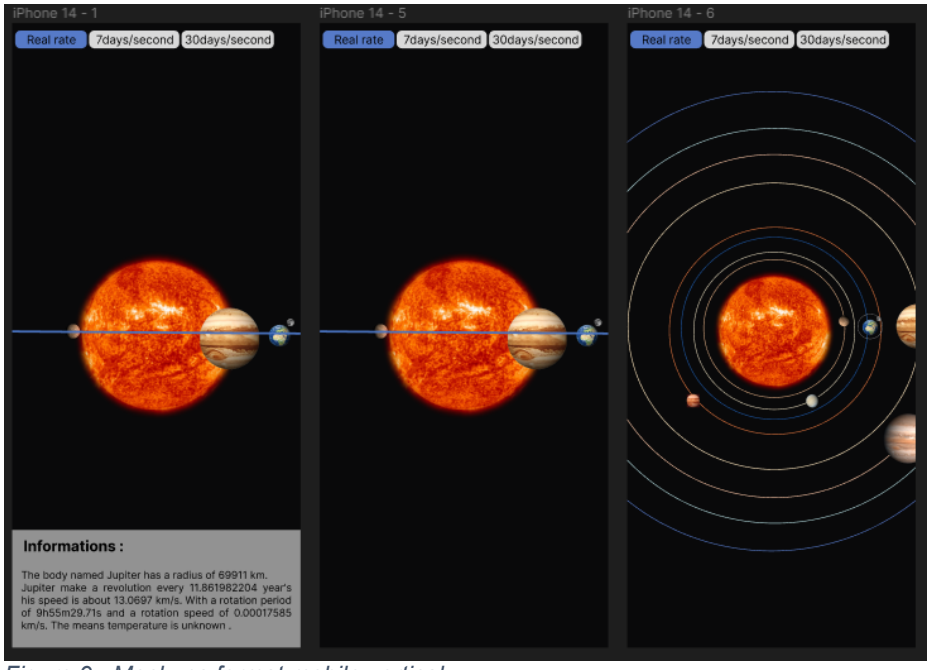
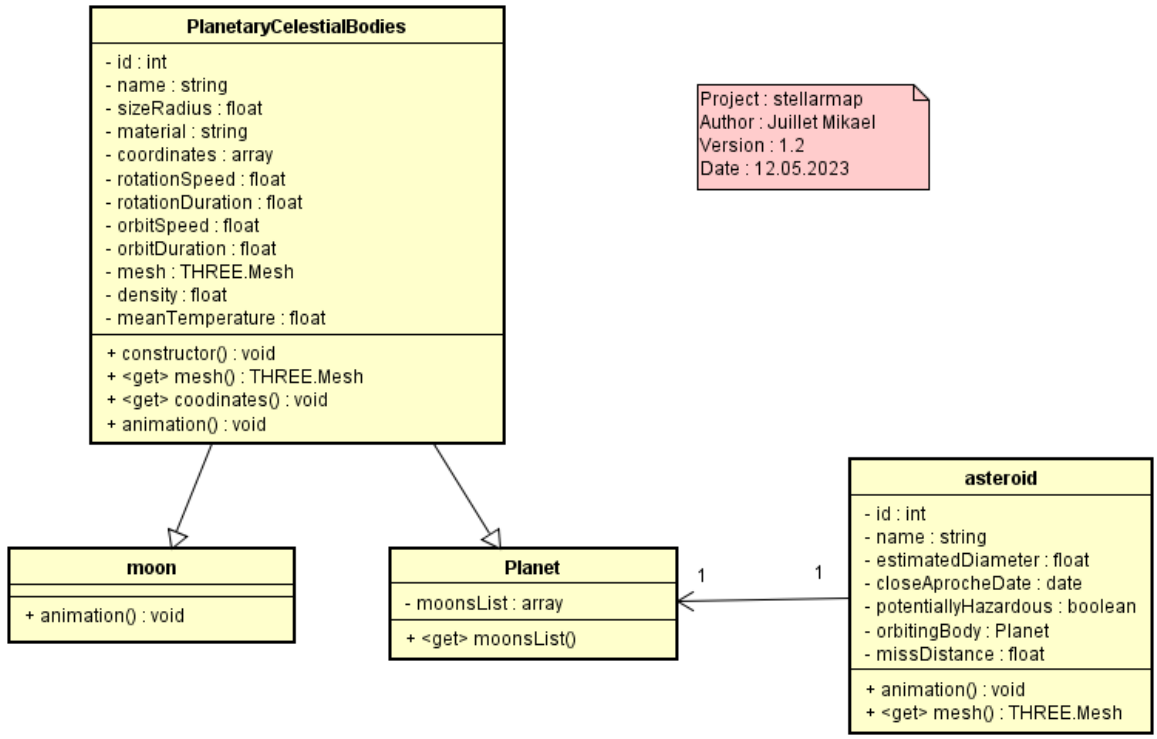


Figure 9 : Mockups format mobile vertical.

3.1.4 Diagramme de classe

Le diagramme de classe représente le contenu ainsi que les interactions entre les différentes classes du projet.



3.1.5 Diagramme de séquences

Le diagramme de séquence représente un cas nominal du fonctionnement complet de l'app depuis l'arrivée sur site jusqu'à l'affichage de la scène.

3.2 Stratégie de test

Deux types de tests vont être effectuer, des tests unitaires sur le filtrage des donnée et la réception des données et des tests fonctionnels 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.

Des tests seront effectués à chaque fin de sprint afin de valider le fonctionnement du site et plus spécifiquement du sprint effectué.

3.3 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.

3.4 Planification

Date de début : 2 mai 2023 à 8h50
Date de fin : 2 juin 2023 à 16h20

Rendu de documentation et journal de travail : le mardi et vendredi.

Planification finale

3.5 Dossier de conception

3.5.1 Logiciels / Framework utilisé :

Nom	Version	Utilisation
Ajax (W3schools, -)	-	Requêtes
Balsamiq Wireframe (balsamiq, 2023)	4.6.5	Wireframe
Figma (figma, 2023)	-	Mockup
FileZila	3.64.0	Déploiement
HTML, CSS	Html 5, CSS 3	Mise en page
Moment (momentjs, 2023)	2.29.4	Gestion des dates
Postman (postman, 2023)	v10.14	Requêtes
Three.js (WebGL) (three.js, 2023)	0.152.2	Rendu 3D
Visual Studio Code (visualstudio, 2023)	1.74.3	Editeur de code
Vite.js (vitejs, 2023)	4.1.1	Frontend Tooling
Vitest (vitest, 2021)	0.31.0	Tests
Merci app (merci-app, 2023)	-	Correcteur d'orthographe en français
Grammarly (grammarly, 2023)	-	Correcteur d'orthographe en anglais

4 Réalisation

4.1 Requêtes API

Pour ce projet, il est nécessaire de mettre en place deux API de la Nasa, la première s'appelle Horizon et sert à la récupération de données précises sur les objets céleste de notre système solaire. La deuxième API est Near Earth Objets et répertorie tous astéroïdes proches de la terre.

L'ensemble des requêtes s'effectuent sous forme de Get et en Https, il est important d'utiliser le https car elle garantit l'intégrité des données envoyé en chiffrant la requête, garantit l'identité du server et renforce la confiance des utilisateurs sur le site.

Il est possible de retrouver les liens des documentations des différentes API ci-dessous :

- **Horizon** : https://ssd-api.jpl.nasa.gov/doc/horizons.html
- **Near Earth Objects** (section : Asteroids - NeoWs) : https://api.nasa.gov/

L'api Near Earth Objet a besoin d'une clef API, c'est pourquoi j'ai choisi de crée un fichier .env qui contient ma clef API ainsi que les URL respectives des API.

```
VITE_API_URL_HORIZON=https://ssd.jpl.nasa.gov/api/horizons.api
VITE_API_URL_NEO=https://api.nasa.gov/neo/rest/v1/feed
VITE_API_KEY=APIKEY
```

Figure 10 : Contenu du fichier .env

Pour l'ensemble des mes API je forme mon Url via une fonction formatURL qui vas me permettre d'assembler mon URL avec les paramètres demandé par l'API.

```
function formatURL(url, parameters) {

    var fullURL = url + "?";

    Object.keys(parameters).forEach( (key, index) => {
        fullURL = fullURL + key + "=" + parameters[key]

        if (index !== Object.keys(parameters).length - 1) {
            fullURL = fullURL + "&";
        }

    });

    return fullURL;
}
```

Figure 11 : Fonction de formating des URL de requêtes.

4.1.1 Horizon system

Url de requête : https://ssd-api.jpl.nasa.gov/doc/horizons_file.html

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.

La requête est effectuée via un fetch qui vas formater l'url et récupérer soit une réponse soit une erreur.

```
export function GetHorizonSpecificBody(bodyId) {
    return new Promise(resolve => {
        const parameters = {};
        parameters.format = 'json';
        parameters.COMMAND = bodyId;
        parameters.OBJ_DATA = 'YES';
        parameters.MAKE_EPHEM = 'YES';
        parameters.EPHEM_TYPE = 'VECTORS';
        parameters.START_TIME = moment().subtract(1, 'day').format('YYYY-MM-DD');
        parameters.STOP_TIME = moment().format('YYYY-MM-DD');
        parameters.STEP_SIZE = '1d';

        fetch(formatURL(urlHorizon, parameters), {
            headers: {
                'Origin' : 'http://localhost:5173'
             }
        })
        .then(response => {
            resolve(response.json());
        })
        .catch(error => {
            console.error('Error:', error);
            resolve(error);
        });

    })
}
```

Figure 12 : Requête Horizon

Cependant lors ce que la requête est effectuée depuis le navigateur elle engendre une erreur corse. Pour résoudre ce problème, j'ai ajouté dans le 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.

Il est possible de récupérer les données via Postman.

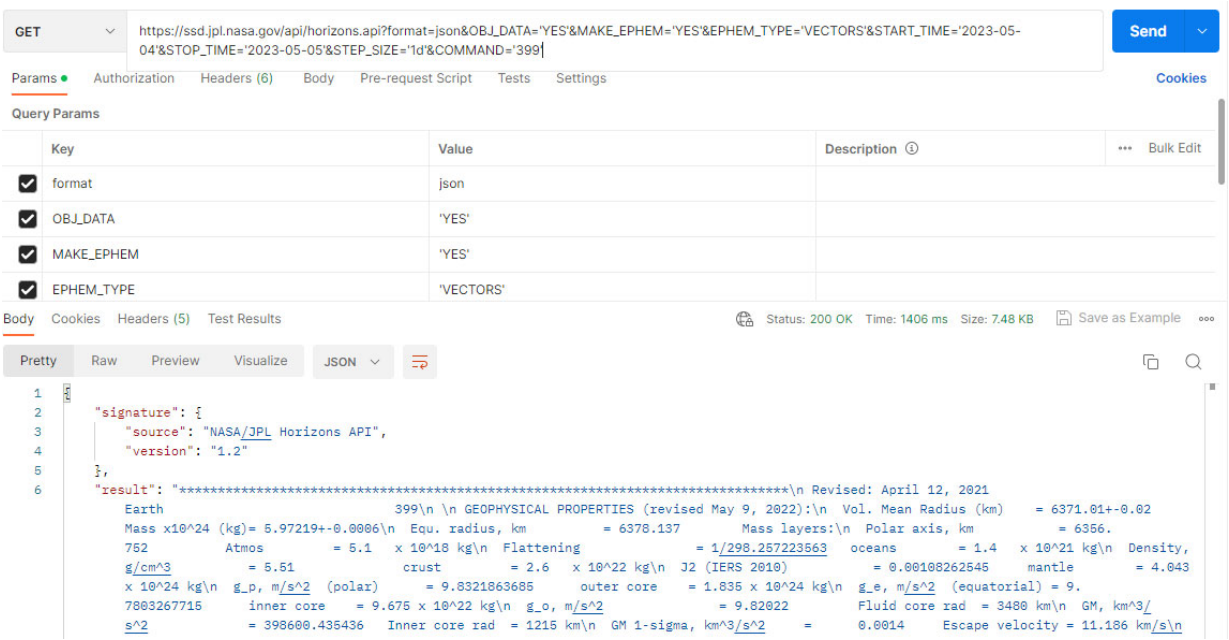


Figure 13: Données retournées via Postman.

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

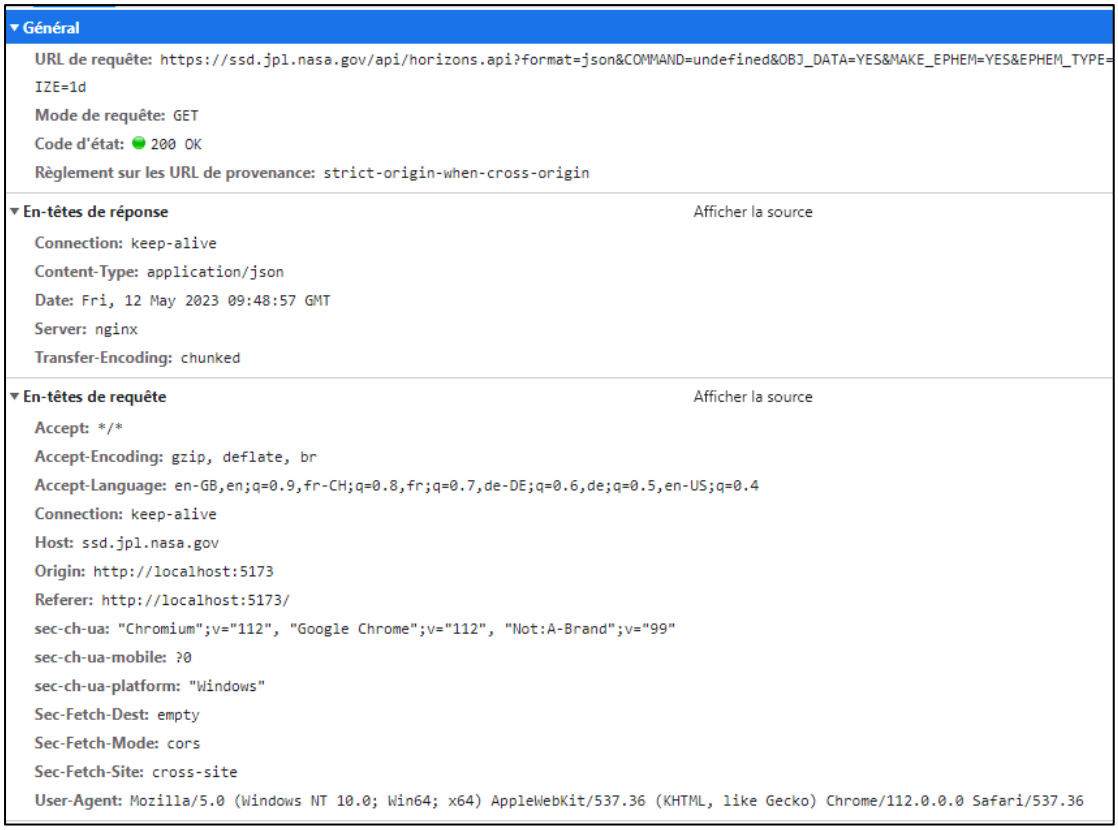


Figure 14 : Etat de la requête Horizon API sans données.

Figure 15 : 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 et de l'exportation les données dans un fichier JSON. Ainsi le moins de temps possible n'est perdu. J'ai créé une issue sur GitHub afin de maintenir le projet a jour.

La réponse de la Nasa a été expéditive (traduction du document original) :
« 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. »

Suite à ce message nous avons décidé garder le document JSON pour la réalisation de ce projet.

4.1.1.1 Récupération des données

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

Voici un exemple des données reçues après suppression des espaces, les espaces ont été enlevé car ceux-ci posent des problèmes de correspondance des regex. Pour être plus précis les regex se fient à une expression régulière et prend donc en compte les espace pour éviter des problèmes de compatibilité entre deux valeurs, il est important d'enlever le maximum de valeurs pouvant interférer.

```
**********Revised:April12,2021Mercury199 dataFilter.js?t=1683886315807:19
PHYSICALDATA(updated2021-Apr-
12):Vol.MeanRadius(km)=2440+-1Density(gcm^-3)=5.427Massx10^23(kg)=3.302Volume(x10^10km^3)=6.085Siderealrot.period=58.6463dSid.rot.rate(rad/s)=0
.00000124001Meansolarday=175.9421dCoreradius(km)=~1600GeometricAalbedo=0.106Surfaceemissivity=0.77+-0.06GM(km^3/s^2)=22031.86855Equatorialradius
,Re=2440kmGM1-sigma(km^3/s^2)=Massratio(Sun/plnt)=6023682Mom.ofInertia=0.33Equ.gravitym/s^2=3.701Atmos.pressure(bar)=
<5x10^-15Max.angulardiam.=11.0"MeanTemperature(K)=440Visualmag.V(1,0)=-0.42Obliquitytoorbit[1]=2.11'+/-0.1'Hill'ssphererad.Rp=94.4Siderealorb.p
er.=0.2408467yMeanOrbitvel.km/s=47.362Siderealorb.per.=87.969257dEscapevel.km/s=4.435PerihelionAphelionMeanSolarConstant(W/m^2)1446262789126Max
imumPlanetaryIR(W/m^2)1270055008000MinimumPlanetaryIR(W/m^2)666**********
**********Ephemeris/API_USERMonMay807:4
@:322023Pasadena,USA/Horizons**********Targetbodyname:Mercury(199)
{source:DE441}Centerbodyname:Earth(399){source:DE441}Center-
sitename:BODYCENTER**********Starttime:A.D.2023-May-
0400:00:00.0000TDBStoptime:A.D.2023-May-0500:00:00.0000TDBStep-
size:1440minutes**********Centergeodetic:0.0,0.0,0.0{E-
lon(deg),Lat(deg),Alt(km)}Centercylindric:0.0,0.0,0.0{E-
lon(deg),Dxy(km),Dz(km)}Centerradii:6378.137,6378.137,6356.752km{Equator_a,b,pole_c}Outputunits:KM-
SCalendarmode:MixedJulian/GregorianOutputtype:GEOMETRICCartesianstatesOutputformat:3(position,velocity,LT,range,range-
rate)EOPfile:eop.230504.p230727EOPcoverage:DATA-BASED1962-JAN-20TO2023-MAY-04.PREDICTS->2023-JUL-
26Referenceframe:EclipticofJ2000.0**********JDTDBXYZVXVYVZLTRGRR**********
**********$$SOE2460068.500000000=A.D.2023-May-
0400:00:00.0000TDBX=6.439857249172552E+07Y=5.346674700502940E+07Z=1.766200542283878E+05VX=5.932433425858534E+00VY=-9.068016481428440E+00VZ=-4.8
94541217796761E+00LT=2.791973394962744E+02RG=8.370125667464858E+07RR=-1.238470929543310E+002460069.500000000=A.D.2023-May-
0500:00:00.0000TDBX=6.496800946250510E+07Y=5.274906761607163E+07Z=-2.463364871165343E+05VX=7.230568569470446E+00VY=-7.535644675529745E+00VZ=-4.
893499593699156E+00LT=2.791468567349237E+02RG=8.368612232353663E+07RR=8.778380713596032E-
01$$EOE**********TIMEBarycentricDynamicalTime("TDB"orT_eph)outputwasreques
ted.Thiscontinuouscoordinatetimeisequivalenttotherelativisticpropertimeofaclockatrestinareferenceframeco-
movingwiththesolarsystembarycenterbutoutsidethesystem'sgravitywell.Itistheindependentvariableinthesolarsystemrelativisticequationsofmotion.TDBr
unsatauniformrateofoneSIsecondpersecondandisindependentofirregularitiesinEarth'srotation.CALENDARSYSTEMMixedcalendarmodewasactivesuchthatcalend
ardatesafterAD1582-Oct-15(ifany)areinthemodernGregoriansystem.Datespriorto1582-Oct-
5(ifany)areintheJuliancalendarsystem,whichisautomaticallyextendedfordatespriortoitsadoptionon45-Jan-
1BC.TheJuliancalendarisusefulformatchinghistoricaldates.TheGregoriancalendarmoreaccuratelycorrespondstotheEarth'sorbitalmotionandseasons.A"Greg
orian-
only"calendarmodeisavailableifsuchphysicaleventsaretheprimaryinterest.REFERENCEFRAMEANDCOORDINATESEclipticatthestandardreferenceepochReferencee
poch:J2000.0X-Yplane:adoptedEarthorbitalplaneatthereferenceepochNote:IAU76obliquityof84381.448arcsecondswrtICRFX-YplaneX-axis:ICRFZ-
axis:perpendiculartotheX-
Yplaneinthedirectional(+or-)senseofEarth'snorthpoleatthereferenceepoch.Symbolmeaning:JDTDBJulianDayNumber,BarycentricDynamicalTimeXX-
componentofpositionvector(km)YY-componentofpositionvector(km)ZZ-componentofpositionvector(km)VXX-componentofvelocityvector(km/sec)VYY-
componentofvelocityvector(km/sec)VZZ-componentofvelocityvector(km/sec)LTOne-waydown-legNewtonianlight-
time(sec)RGRange;distancefromcoordinatecenter(km)RRRange-
rate;radialvelocitywrtcoord.center(km/sec)ABERRATIONSANDCORRECTIONSGeometricstatevectorshaveNOcorrectionsoraberrationsapplied.Computationsby...
SolarSystemDynamicsGroup,HorizonsOn-LineEphemerisSystem4800OakGroveDrive,JetPropulsionLaboratoryPasadena,CA91109USAGeneralsite:https://ssd.jpl.
nasa.gov/Mailinglist:https://ssd.jpl.nasa.gov/email_list.ht…**********
```


Figure 16 : 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 qui a généré les regex en fonction des données que je luis ai transmis, étant donné que chaque planète a des données différentes il était plus rapide de demander à ChatGPT de générer les regex en fonction des données voulues. (OpenAI, 2023)

Comme exemple le code ci-dessous qui cherche un texte « Vol. Mean Radius (km) = » qui est suivit d'une suite de chiffres.

Dans le regex nous avons plusieurs modifications possibles du texte :

1. [Mm] cherche la lettre M en majuscule ou minuscule.
2. [Rr] cherche la lettre R en majuscule ou minuscule.

Il existe une multitudes d'autres modifications possibles qui permettent de rechercher des données plus ou moins complexes.

```
const regexMeanRadius = /Vol\.[Mm]ean[Rr]adius\(km\)=([\d.]+)/
const matchMeanRadius = dataWithoutSpace.match(regexMeanRadius);
```

```
// Return data in object form
return {
  id : matchNameID[2],
  name : matchNameID[1],
  sizeRadius : matchMeanRadius[1],
  coordinate : {
    x : matchPlacment[1],
    y : matchPlacment[2],
    z : matchPlacment[3]
  },
  rotationSpeed : matchRotationRate[1],
  rotationDuration : matchRotationDays[2],
  orbitSpeed : matchOrbitalSpeed[2],
  orbitDuration : matchOrbitPeriod[2],
  oblliquity : matchObliquity[2],
  density : matchDensity[1],
  meanTemperature :  matchMeanTemp ? matchMeanTemp[2] : "Unknown"
};
```

Figure 17 : Exemple de l'utilisation de regex

4.1.2 Near Earth Objects

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

La requête Near Earth Objects demande peu de paramètres, il suffit de passer dans l'url une date de début, une date de fin et une clé api donnée par la Nasa.

La fonction de requête ci-dessous montre qu'il est effectué un fetch de mon url formatée en passant url et les paramètres, puis la réponse est récupérée et retournée sous format json. Si une erreur apparait elle sera récupérée via le catch. (developer.mozilla.org, 2022)

```
export function GetNearEarthObjects(apiKey, startDate, endDate) {
    return new Promise(resolve => {
        const parameters = {};
        parameters.start_date = startDate;
        parameters.end_date = endDate;
        parameters.api_key = apiKey;

        fetch(formatURL(urlNeo, parameters))
        .then(response => {
            resolve(response.json())
        })
        .catch(function (err) {
            resolve("Something went wrong!", err);
        });

    })
}
```

Figure 18 : Fonction de création fetch des objets proches.

Voici les données récupérées :

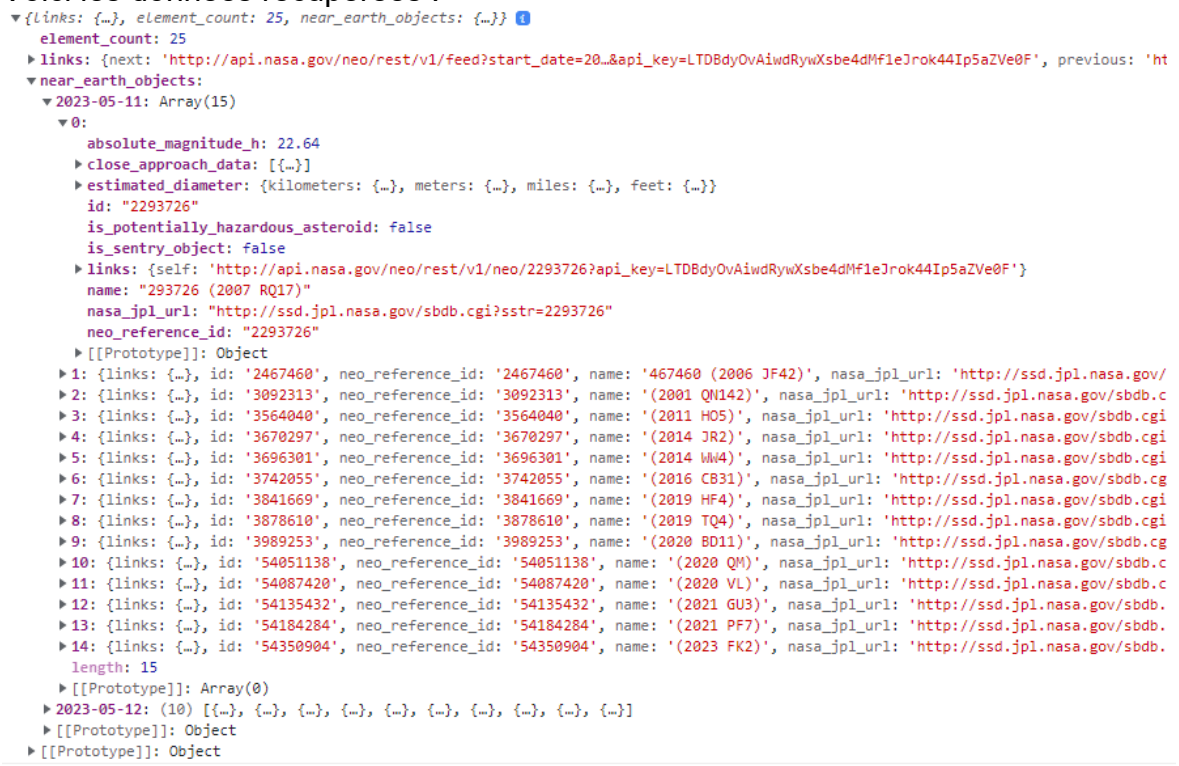


Figure 19 : Données récupérées via a l'api NEO.

4.2 3D Affichages et animations

Ce projet utilise la librairie Three.js qui permet d'afficher sur des pages web du contenu en 3D.

Three.js est utilisé car il contient une énorme communauté du à sa grande popularité, il est facile à apprendre, il est compatible avec une large gamme de navigateurs et c'est une librairie que j'ai déjà expérimenté.

Threejs a besoin de 3 choses minimum :

- Une scène
- Une caméra
- Un moteur de rendu

La scène est une sorte de conteneur, elle permet de définir les éléments qui doivent être affiché ainsi que leurs placements. La caméra c'est elle qui va « filmer » ce que qu'il y a dans la scène et le moteur de rendu a pour objectif d'afficher la scène dans le navigateur via WebGL. (Bradley, 2022)

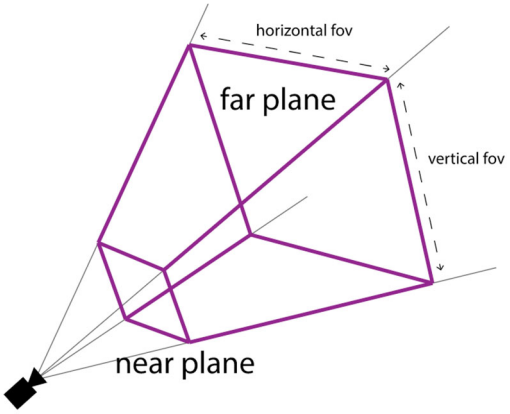


Figure 20 : Exemple de ce qu'est une scène. (Punkasem, 2017)

Il faut aussi ajouter une fonction d'animation, cette fonction crée une boucle qui fera afficher la scène via le moteur de recherche à chaque fois que l'écran est actualisé.

Quand un écran fait 60 images par seconde (frames per second ou FPS) c'est-à-dire qu'il y a 60 images qui sont affichées en une seconde.

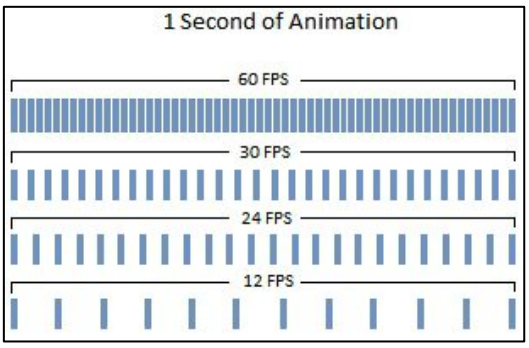


Figure 21 : Schéma d'animation par seconde.

Ce projet utilise programmation orientée objet (POO) qui réparti en plusieurs classes les éléments de mon projet.

- Renderer
  - Il est utilisé pour initialiser le rendu et géré l'animation.
- PlanetaryCelestialBody
  - Utiliser pour gérer les objets célestes planétaires.
- Planet
  - Hérite de PlanetaryCelestialBody.
- Moon
  - Hérite de PlanetaryCelestialBody.
  - Ajoute un paramètre orbitingBody qui définit planète autour de la quel elle orbite.
- Asteroid
  - Utilisé pour gérer les astéroïdes.

4.2.1 Classe Renderer

La classe Renderer est utilisé pour créer le rendu, il est défini dans celui-ci lors de la construction de la classe qu'il construit la scène, la caméra, la lumière, le renderer, la Skybox, le soleil et les planètes. Il prend comme paramètre un canvas ainsi qu'une liste d'objets à ajouter dans la scène.

Création de la skybox lors de la création de la géométrie, il est défini une valeur de rayon de -20000 afin que les faces de la sphère soit retournée vers l'intérieur.
Ce procédé est inspiré de ce site : https://codinhood.com/post/create-skybox-with-threejs

```
// Skybox creation
var skyboxGeometry = new THREE.SphereGeometry(-20000, 64, 16); // -20000 for facing interior
const skyboxTexture = new THREE.TextureLoader().load("./src/assets/images/Skybox.jpg");
const skyboxMaterial = new THREE.MeshBasicMaterial({ map: skyboxTexture });
const skybox = new THREE.Mesh( skyboxGeometry, skyboxMaterial );
this.#scene.add(skybox);
```

Figure 22 : Création de la skybox.

La création du soleil est classique, il est imporant de préciser que la taille du soleil est arbitraire sinon les planètes serait trop petites pour être visualisé c'est ce pourquoi la valeur de sa taille est de 8.

```
// Sun creation
var sunGeometry = new THREE.SphereGeometry(8, 64, 16);
const sunTexture = new THREE.TextureLoader().load("./src/assets/images/Sun.jpg");
const sunMaterial = new THREE.MeshBasicMaterial({ map: sunTexture });
this.#sun = new THREE.Mesh( sunGeometry, sunMaterial );
this.#sun.position.set(0,0,0);
this.#sun.rotation.x = 360;
this.#scene.add(this.#sun);
```

Figure 23 : Création du soleil.

La création des bodies (corps célestes) sont crée via un paramètre définit lors de la création de la classe qui vas boucler autour d'une liste de corps et récupérer le système de chaque corps.

```
// Create bodies
this.#bodiesList.forEach(bodies => {
    const bodiesSystem = bodies.planetarySystem;
    this.#scene.add(bodiesSystem);
    bodies.createOrbit();
});
```

Figure 24 : Création des planètes.

Il dispose d'une méthode animate qui se charge d'ajouter l'animation des objets, la rotation du soleil et de dessiner le rendu.

```
animate() {
    this.#bodiesList.forEach(bodies => {
        bodies.animation();
    });

    this.#sun.rotation.y += 0.00007292115;

    this.#renderer.render( this.#scene, this.#camera );
}
```

Figure 25 : Fonction d'animation du renderer

4.2.2 Classe PlanetaryCelestialBody

La classe PlanetaryCelestialBody est utilisée pour créer les objets célestes planétaires. Elle contient les méthodes de création de mesh, d'animation, de création d'orbite et de placement du système.

Elle prend pour paramètre :

- Un id
- Un nom
- Le radius de la taille
- Un fichier de texture
- Des coordonnées
- Une vitesse de rotation
- Une durée de rotation
- Une vitesse orbitale
- Une durée orbitale
- Une température moyenne

Les corps célestes planétaires définissent tout corps en orbite autour du soleil.

Une mesh est objets basés sur un maillage polygonal triangulaire.

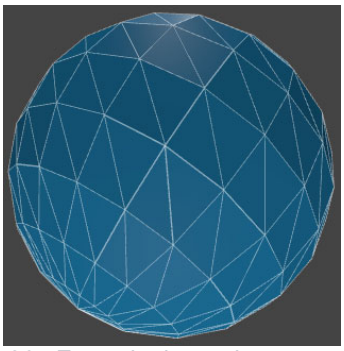


Figure 26 : Exemple de mesh sur une sphère

Cette méthode sert a créé une mesh en fonction de la taille et de la position du corp.

```
#createMesh() {
  const geometry = new THREE.SphereGeometry(this.sizeRadius / 1000, 64, 16 );
  const texture = new THREE.TextureLoader().load("./src/assets/images/" + this.textureFile);
  const material = new THREE.MeshBasicMaterial({ map: texture });
  const mesh =  new THREE.Mesh(geometry, material);
  mesh.position.set(parseFloat(this.coordinates.x) / 5000000, parseFloat(this.coordinates.y) / 5000000, 0);
  mesh.rotation.set(3, 3, 0)
  return mesh;
}
```

Figure 27 : Création physique des planètes

La méthode d'animation définit le déplacement du corp sur elle-même ainsi que autour de sont centre comme le soleil.

```
animation() {
  const deltaTime = this.#clock.getDelta();

  this.mesh.rotation.x = 190
  this.mesh.rotation.y += this.rotationSpeed * 10;

  const rayon = (Math.sqrt((Math.pow(this.coordinates.x, 2) /5000000 ) + (Math.pow(this.coordinates.y, 2)/5000000)))
  const distanceFactor = 2 * Math.PI * rayon; //km périmèter
  const timeFactor = this.orbitDuration * 365 * 24 * 60 * 60; //years => seconds

  const vitesse =  distanceFactor / timeFactor;
  this.#pointPivot.rotation.z += vitesse;

}
```

Figure 28 : animation des planètes

4.2.3 Classe Planet

La classe Planet hérite uniquement de la classe PlanetaryCelestialBody.

4.2.4 Classe Moon

La classe Moon hérite elle aussi de la classe PlanetaryCelestialBody mais elle contient en plus une propriété nommée orbitingBody qui permet de définit la planète autour de la quel elle orbite.

4.2.5 Classe Asteroid

Les astéroïdes n'héritent d'aucunes classes car elle particulière et n'a pas le même comportement que les lunes ou planètes.
En effet la classe Asteroid ne contient pas les mêmes parmètre et n'a pas d'animation dédiée.

Elle prend comme paramètre :

- Un nom
- Un id
- Une taille estimée
- Une date d'approche
- Si l'astéroïde est dangereux
- La planète d'orbite
- La distance manquée

```
constructor(id, name, estimatedDiameter, closeAprocheDate, potentiallyHazardous, orbitingBody, missDistance) {
  this.id = id;
  this.name = name;
  this.estimatedDiameter = estimatedDiameter;
  this.closeAprocheDate = closeAprocheDate;
  this.potentiallyHazardous = potentiallyHazardous;
  this.orbitingBody = orbitingBody;
  this.missDistance = missDistance;
}
```

Figure 29 : Constructeur de la classe Ateroid

Elle contient une seul méthode qui est la création de la mesh, elle définit la taille via le paramètre reçu dans le constructeur et définit la position en calculant la distance avec la position de la planète autour de la quel elle orbite.

```
#createMesh() {
  const geometry = new THREE.SphereGeometry(this.estimatedDiameter);
  const texture = new THREE.TextureLoader().load("/src/assets/images/Asteroid.jpg");
  const material = new THREE.MeshBasicMaterial({ map: texture });
  const mesh = new THREE.Mesh(geometry, material);
  mesh.position.set(
    this.orbitingBody.coordinates.x / 5000000 - this.missDistance / 5000000,
    this.orbitingBody.coordinates.y / 5000000 - this.missDistance / 5000000,
    0
    );
  return mesh;
}
```

Figure 30 : Création de la mesh de la classe Asteroid.

4.2.6 Responsive

Le responsive est définit premièrement par le css qui vas proposer si l'écran est plus petit que 600 pixel de large de tourner le l'écran ou d'agrandir la fenêtre, suite à quoi lors de l'utilisation de l'app la fonction ci-dessous permet de changer dynamiquement la taille du renderer ce qui évitera d'avoir des bordures blanches non voulues.

```
// Resize render when listen to resize
window.addEventListener('resize', () => {
    this.#camera.aspect = window.innerWidth / window.innerHeight;
    this.#camera.updateProjectionMatrix();
    this.#renderer.setSize(window.innerWidth, window.innerHeight);
    this.animate();
});
```

Figure 31 : Code permettant au redimensionnement du renderer

4.2.7 Mouvement utilisateur

Pour le mouvement de l'utilisateur, j'utilise une fonction de threejs qui permet de définir le déplacement de la caméra.

Documentation threejs : https://threejs.org/docs/#examples/en/controls/OrbitControls

```
// Add users controls
const controls = new OrbitControls( this.#camera, this.#renderer.domElement );
controls.enablePan = false
controls.maxDistance = 10000
controls.update();
```

Figure 32 : Code permettant le mouvement de l'utilisateur

4.3 Accélération

L'accélération s'effectue en fonction de trois possibilités, aucune accélération elle est a la vitesse normal des planètes, accélération de 7 jours par seconde ce qui indique que 7 jours se passe en une seconde puis 30 jours par seconde.

4.4 Description

La description est une zone de texte sur la droite de l'écran qui affiche une petite description de l'objet cliqué.

4.5 Déploiement

Le déploiement s'effectue sur Swisscenter, afin de parvenir à la connexion au serveur j'ai créé une clé ssh qui me permet de me connecter de façon sécuriser au serveur via FileZila.

Site web : https://stellarmap.mycpnv.ch/

4.6 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
  -

4.7 Favicon

J'ai choisi comme favicon une image de soleil du site png art. (pngarts, -).

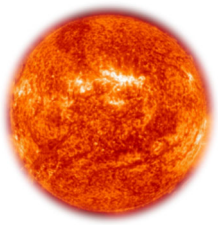


Figure 33 : Favicon

4.8 Description des tests effectués

Type	Prés-requis	Action	Attendu	Résultat
API - Horizon	Données format json.	Filtrer les données de Mercure.	id : “199”, name : “Mercury”, sizeRadius : “2440”, coordinate : { x : “6.439857249172552E+07”, y : “5.346674700502940E+07”, z : "1.766200542283878E+05" }, rotationSpeed : “0.00000124001”, rotationDuration : “58.6463”, orbitSpeed : “47.362”, orbitDuration : “0.2408467”, oblliquity : “2.11”, density : “5.427”, meanTemperature : "440"	Réussi
API - Horizon	Données format json.	Filtrer les données de Venus.	id : “299”, name : “Venus”, sizeRadius : “6051.84”, material : “Venus.png”, coordinate : { x : “1.073618289428096E+07”, y : “1.428930836016723E+08”, z : "6.291878178878881E+06" }, rotationSpeed : “-0.00000029924”, rotationDuration : “243.018484”, orbitSpeed : “35.021”, orbitDuration : “0.61519726”, oblliquity : “177.3”, density : “5.204”, meanTemperature : "735"	Réussi

API - Horizon	Données format json.	Filtrer les données de la terre.	id : “399”, name : “Earth”, sizeRadius : “6371.01”, coordinate : { x : “-1.116613234083490E+08”, y : “-9.446394564291658E+07”, z : "-4.091379678603614E+07" }, rotationSpeed : “0.00007292115”, rotationDuration : “24h”, orbitSpeed : “29.79”, orbitDuration : “1.0000174”, oblliquity : “23.4392911”, density : “5.51”, meanTemperature : "287.6"	Réussi
API - Horizon	Données format json.	Filtrer les données de Mars.	id : “499”, name : “Mars”, sizeRadius : “3389.92”, coordinate : { x : “-9.172658128265008E+07”, y : “2.475395789325977E+08”, z : "7.984203671250358E+06" }, rotationSpeed : “0.0000708822”, rotationDuration : “24.622962h”, orbitSpeed : “24.13”, orbitDuration : “1.88081578”, oblliquity : “25.19”, density : “3.933”, meanTemperature : "210"	Réussi
API - Horizon	Données format json.	Filtrer les données de Jupiter.	id : “599”, name : “Jupiter”, sizeRadius : “69911”, coordinate : { x : “7.900239544305509E+08”, y : “3.977974646088730E+08”,	Réussi

			z : "-1.643804799922679E+07" }, rotationSpeed : “0.00017585”, rotationDuration : “9h55m29.71s”, orbitSpeed : “13.0697”, orbitDuration : “11.861982204”, oblliquity : “3.13”, density : “1.3262”, meanTemperature : "Unknown"	
API - Horizon	Données format json.	Filtrer les données de Saturne.	id : “699”, name : “Saturn”, sizeRadius : “58232”, coordinate : { x : “1.377989499063393E+09”, y : “-6.338851815853779E+08”, z : "-3.764483124414477E+07" }, rotationSpeed : “0.000163785”, rotationDuration : “10h39m22.4s”, orbitSpeed : “9.68”, orbitDuration : “29.447498”, oblliquity : “26.73”, density : “0.687”, meanTemperature : "Unknown"	Réussi
API - Horizon	Données format json.	Filtrer les données de Uranus.	id : “799”, name : “Uranus”, sizeRadius : “25362”, coordinate : { x : “2.056177409121107E+09”, y : “2.306377240773124E+09”, z : "-1.704586625654197E+07" }, rotationSpeed : “-0.000101237”, rotationDuration : “17.24”, orbitSpeed : “6.8”, orbitDuration : “84.0120465”,	Réussi

			oblliquity : “97.77”, density : “1.271”, meanTemperature : "Unknown"	
API - Horizon	Données format json.	Filtrer les données de Neptune.	id : “899”, name : “Neptune”, sizeRadius : “24624”, coordinate : { x : “4.567398114374511E+09”, y : “-2.794219386842926E+08”, z : "-9.485098077934098E+07" }, rotationSpeed : “0.000108338”, rotationDuration : “16.11”, orbitSpeed : “5.43”, orbitDuration : “164.788501027”, oblliquity : “28.32”, density : “1.638”, meanTemperature : "Unknown"	Réussi
API - Neo	startDate : 2023-05-04 endDate : 2023-05-05 apiKey : Clé api valide	Envois de la requête : https://api.nasa.gov/neo/rest/v1/feed?start_date=2023-05-04&end_date=2023-05-05&api_key=APIKEY	Reçois 20 éléments de la requête.	Réussi
API - Neo	startDate : 2023-05-04 endDate : 2023-05-05 apiKey : Clé api non valide	Envois de la requête : https://api.nasa.gov/neo/rest/v1/feed?start_date=2023-05-04&end_date=2023-05-05&api_key=APIKEY	Un message d’erreur.	Réussi
				Réussi
				Réussi
				Réussi
				Réussi
				Réussi
				Réussi

Pour effectuer les tests sur les requête j'ai utilisé le Framework vitetest qui me permet de créer des tests rapidement et facilement. Tous les tests pour l'api Horizon sont semblables seules les données reçues sont différentes, cela me permet de tester que le filtrage des données est réussi.

```
test('horizonAPIFilter() should return Jupiter data', () => {

  //Given
  const jupiter = Allplanets.planets[4].result;
  const jupiterExpected = {
      id : "599",
      name : "Jupiter",
      sizeRadius : "69911",
      material : "Jupiter.png",
      coordinate : {
        x : "7.900239544305509E+08",
        y : "3.977974646088730E+08",
        z : "-1.643804799922679E+07"
      },
      rotationSpeed : "0.00017585",
      rotationDuration : "9h55m29.71s",
      orbitSpeed : "13.0697",
      orbitDuration : "11.861982204",
      oblliquity : "3.13",
      density : "1.3262",
      meanTemperature : "Unknown"
  };

  //When
  const result = horizonAPIFilter(jupiter);

  //Then
  expect(result).toEqual(jupiterExpected);

})
```

Figure 34 : Exemple de tests sur l'api Horizon

NEO

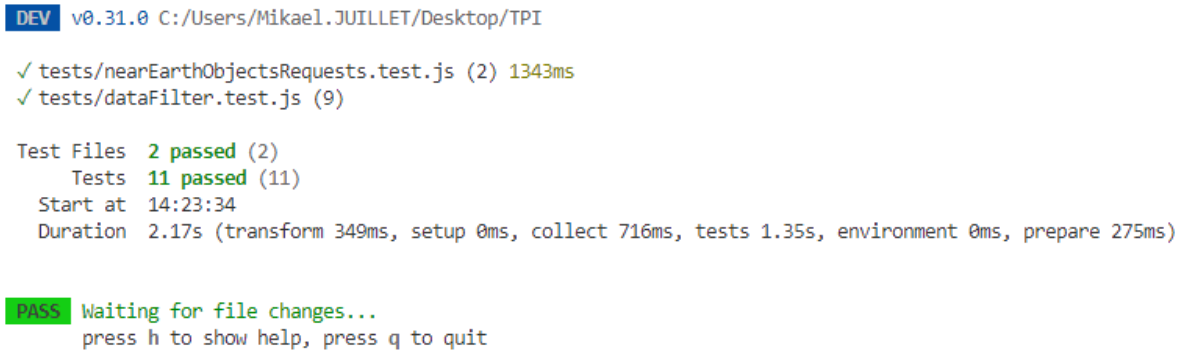


Figure 35 : Vision des tests passé.

4.9 Erreurs restantes

4.10 Liste des documents fournis

Les documents fournis sont :

- Documentation du projet
- Journal de travail
- Journal de bord
- Cahier des charges

5 Conclusions

6 Bilan personnel

7 Résumé

8 Bibliographie

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

Bradley, S. (2022). *Scene, Camera and Renderer*. Récupéré sur sbcode.net: https://sbcode.net/threejs/scene-camera-renderer/#:~:text=The%20Renderer%20displays%20the%20scene,2D%20image%20for%20the%20Canvas.

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

claude aubry. (21, mai 2018). https://claudeaubry.fr/post/2018/extraits-du-livre-scrum/. Paris, Paris, France.

cpnv.ch. (-, - -). *Icescrum*. Récupéré sur Icescrum.cpnv.ch: https://icescrum.cpnv.ch/#/

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

developer.mozilla.org. (2022, Décembre 23). *Utiliser Fetch*. Récupéré sur developer.mozilla.org: https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch

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/

grammarly. (2023). *grammarly*. Récupéré sur grammarly: https://app.grammarly.com/

kjpargeter. (s.d.). Free photo starry night sky. *Free photo starry night sky.* feepick, -. Récupéré sur https://www.freepik.com/free-photo/starry-night-sky_7061153.htm#query=stars&position=2&from_view=search&track=sph

merci-app. (2023). *merci-app*. Récupéré sur merci-app.com: https://www.merci-app.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

OpenAI. (2023, - -). *Introducing GPT-4, OpenAI's most advanced system*. Récupéré sur Introducing GPT-4, OpenAI's most advanced system: https://openai.com/

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

pngarts. (-, - -). *Images Transparentes de soleil*. Récupéré sur pngarts: https://www.pngarts.com/fr/explore/123391

postman. (2023). *What is Postman?* Récupéré sur postman: https://www.postman.com/

Punkasem. (2017). *Learning Three.js*. Récupéré sur junethanaon.com: https://junethanaon.com/blog/blog-threejs.html

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/
vitest. (2021, - -). *vitest*. Récupéré sur vitest: https://vitest.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
Wikipedia. (22, Février 2023). https://fr.wikipedia.org/wiki/Mod%C3%A8le_en_cascade. -, -, -.
wrike. (-, - -). *wrike*. Récupéré sur wrike.com: https://www.wrike.com/main/

9 Table des illustrations

10 Lexique

A

Agile
: Une méthode Agile est une approche répétée et collaborative, capable de prendre en compte les besoins initiaux du client et ceux liés aux évolutions. · 5

API
: Permet d'accéder aux fonctions ou aux données d'une application à distance. · 6, 12, 15, 16

C

CFC
: Certificat Fédérale de Capacité · 4

E

erreur corse
: Produisent lorsqu'un serveur ne renvoie pas les en-têtes HTTP requis par la norme CORS. · 13

F

feature
: Feature ou fonctionnalité, exprime un objectif préci d'un projet informatique. · 5

G

Get
: Moyen pour vous de récupérer des données à partir d'une source de données à l'aide d'Internet. · 12

H

Https
: Protocole de transfert hypertexte est un protocole de communication client-serveur. · 12

I

Illustration en deux dimensions de l'interface d'une page, se concentre spécifiquement sur l'allocation d'espace. · 7

issue
: Problème, référence les problèmes relatif au projet. · 15

J

JSON

M

P

R

S

T

U

W

Bibliothèque de graphismes pour le Web est une API JavaScript permettant le rendu de graphismes en 2D ou 3D avec de hautes performances, sans avoir à utiliser de plugin. · 19

11 Annexes

11.1 Planification initiale

Description	Catégorie	Progrès	Début	Heures prévu
Sprint 1				
Diagramme de classes	Base du projet	0%	03.05.2023	0.75
Diagramme de séquence	Base du projet	0%	03.05.2023	0.75
Création de la classe planète	Base du projet	0%	03.05.2023	1.00
Création de la classe satellite	Base du projet	0%	03.05.2023	1.00
Ajout des opérations dans les classes	Base du projet	0%	05.05.2023	2.25
Ajout de vitejs	Base du projet	0%	05.05.2023	0.25
Création du fichier détenant les codes de planètes	Base du projet	0%	05.05.2023	0.50
Sprint 2				
Création d'un contrôleur	API	0%	08.05.2023	0.25
Création d'un modèle	API	0%	08.05.2023	0.25
Ajout d'un fichier .env	API	0%	08.05.2023	0.25
Récupération de la clef API	API	0%	08.05.2023	0.25
Requêtes de récupération des planètes	API	0%	08.05.2023	3.00
Requêtes de récupération des objets proches	API	0%	09.05.2023	3.00
Requêtes de récupération des images	API	0%	11.05.2023	2.25
Récupération des erreurs dans le contrôleur	API	0%	12.05.2023	0.75
Lien entre la récupération des données et les classes	API	0%	12.05.2023	0.75

Sprint 3				
Création de maquettes	Planettes	0%	14.05.2023	0.75
Ajout de three.js	Planettes	0%	14.05.2023	0.25
Création des planètes	Planettes	0%	14.05.2023	0.75
Placement des planètes	Planettes	0%	15.05.2023	0.50
Orbite sidérale	Planettes	0%	15.05.2023	0.50
Orbite autour du soleil	Planettes	0%	15.05.2023	0.75

Sprint 4				
Ajout du déplacement utilisateur	Satelites	0%	22.05.2023	2.25
Création des Satélites	Satelites	0%	23.05.2023	1.50
Ajout des lunes	Satelites	0%	25.05.2023	1.50
Placement sur la carte	Satelites	0%	25.05.2023	1.50
Ajout de l'orbite	Satelites	0%	26.05.2023	1.50
Orbite sidérale	Planettes	0%	26.05.2023	1.50

Sprint 5				
Création de maquettes	UI	0%	30.05.2023	0.75
Placement du canvas en arrière-plan	UI	0%	30.05.2023	0.75
Ajout de la desciption des planettes	UI	0%	01.06.2023	2.25
Ecoute d'un clique sur planètes	UI	0%	02.06.2023	0.75
Ajout de changement de vitesse	UI	0%	02.06.2023	1.50

Planification de projet

Système solaire

Mikael Juillet

Date de début 02.05.2023

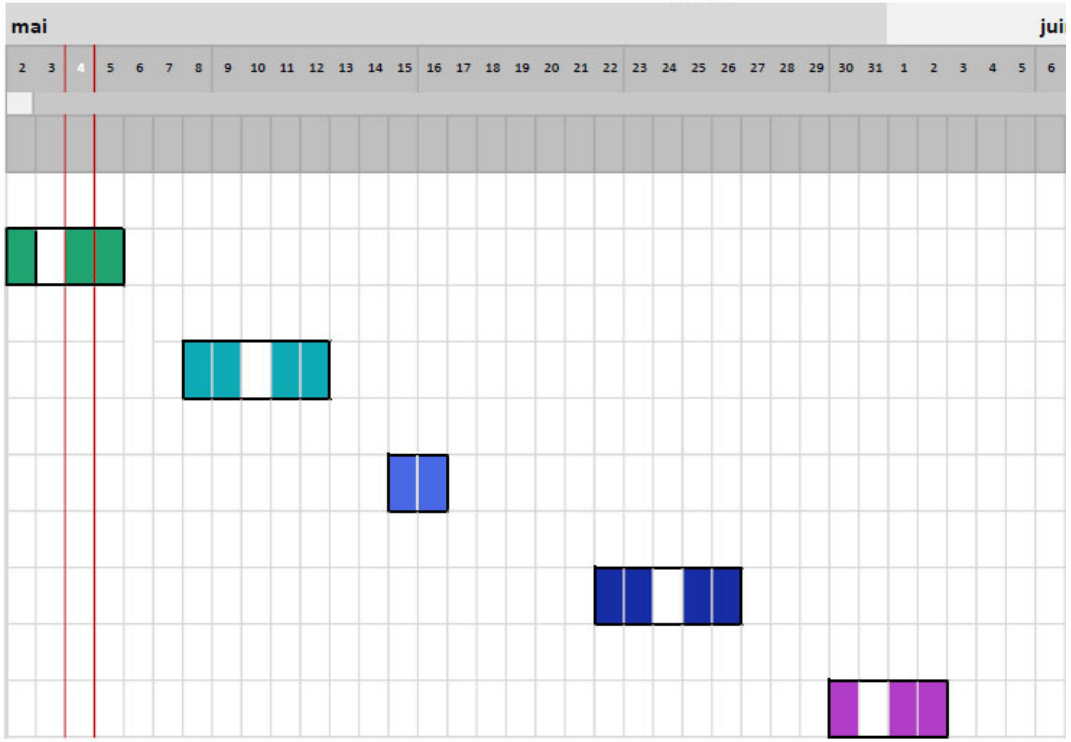
Incrément de défilement 0

Légende :

Définition :

Base du projet	API	Planettes	Satelites	UI
Création de la base du projet, création des classes des planètes et satellites.	Création de toutes les requêtes à l'API.	Affichage des planètes sur leurs placements définit par l'api.	Affichage des satellites (objets proches) sur leurs placements définis par l'api.	Affichage de la description des planètes et du changement de vitesse.

Description du jalon	Catégorie	Progrès	Start	jours
Sprint 1				
Semaine 1	Base du projet	0%	02.05.2023	3
Sprint 2				
Semaine 2	API	0%	08.05.2023	4
Sprint 3				
Semaine 3	Planettes	0%	15.05.2023	2
Sprint 3				
Semaine 4	Satelites	0%	22.05.2023	4
Sprint 4				
Semaine 5	UI	0%	30.05.2023	3



Exemple d'un sprint d'une semaine

Système solaire

Mikael Juillet

Date de début 02.05.2023

Légende: Documentation | Implémentation | Annalyse | Tests

Sprint reviews

* P = Période de 45 minutes

Horaire

Jours	P1	P2	P3	P4	P5	P6	P7	P8	P9
Lundi									
Mardi									
Mercredi									
Jeudi									
Vendredi									
Samedi									
Dimanche									

Note : Il est important de prendre en compte que la disposition changera car il y a par exemple des semaines avec seulement deux jours mais dans ces deux jours il y aura de l'analyse et des tests même s'ils ne sont pas prévus. C'est un schéma approximatif.

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

11.3 Journal de travail

11.4 Archives du projet

Media, … dans une fourre en plastique