StellarMap

Figure 1 : Système solaire (freepik, -)

Table des matières

2 Analyse préliminaire

2.1 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.

2.2 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

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.

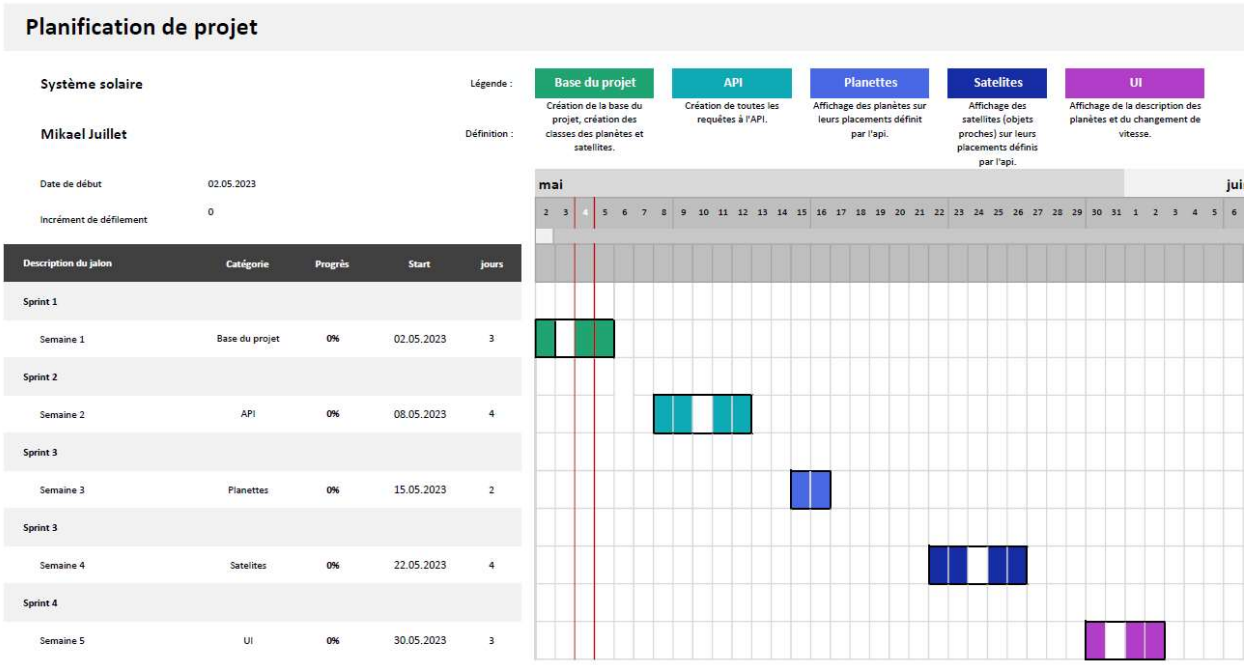


Figure 2 : Planification initiale du projet

3 Analyse / Conception

3.1 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 :

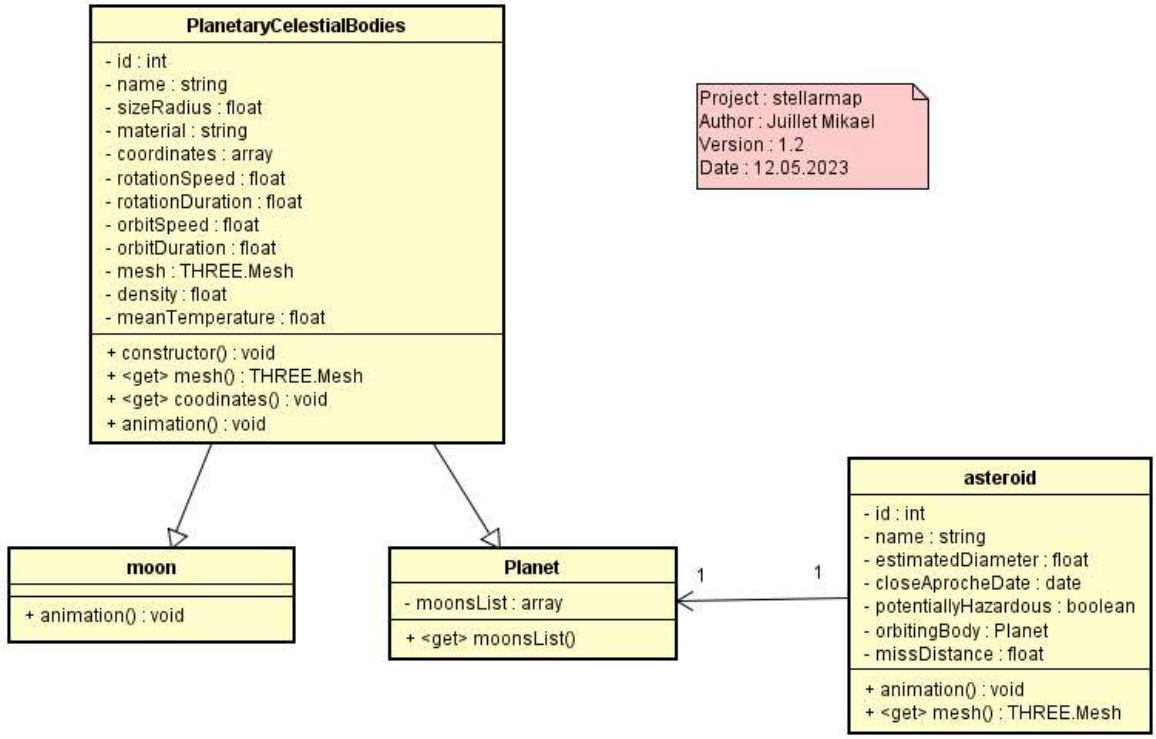
Code source : https://github.com/Juillet-Mikael/TPI
Planification du projet : https://icescrum.cpnv.ch/p/TPIJUILLET/#/project
Documentation :
https://github.com/Juillet-Mikael/TPI/blob/main/documents/documentation.docx
Journal de travail :
https://github.com/Juillet-Mikael/TPI/blob/main/documents/journaux.xlsm

3.1.1.1 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.

3.1.1.2 Maquettes

3.1.1.3 Diagramme de classe



3.1.1.4 Diagramme de séquences

3.2 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.

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

3.5 Dossier de conception

3.5.1 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

4 Réalisation

4.1 Dossier de réalisation

4.1.1 Requête API

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

```
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;
}
```

4.1.1.1 Variables d'environnement

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

4.1.1.2 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.

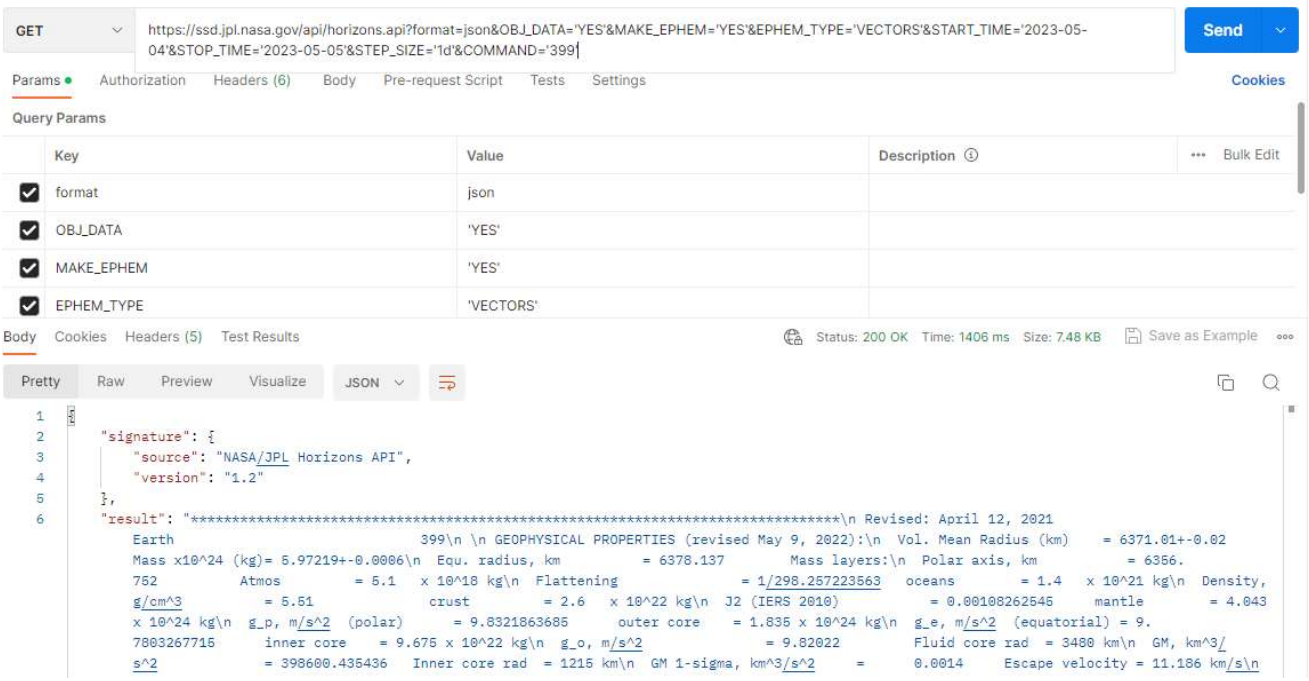


Figure 3: Données retournées via Postman.

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

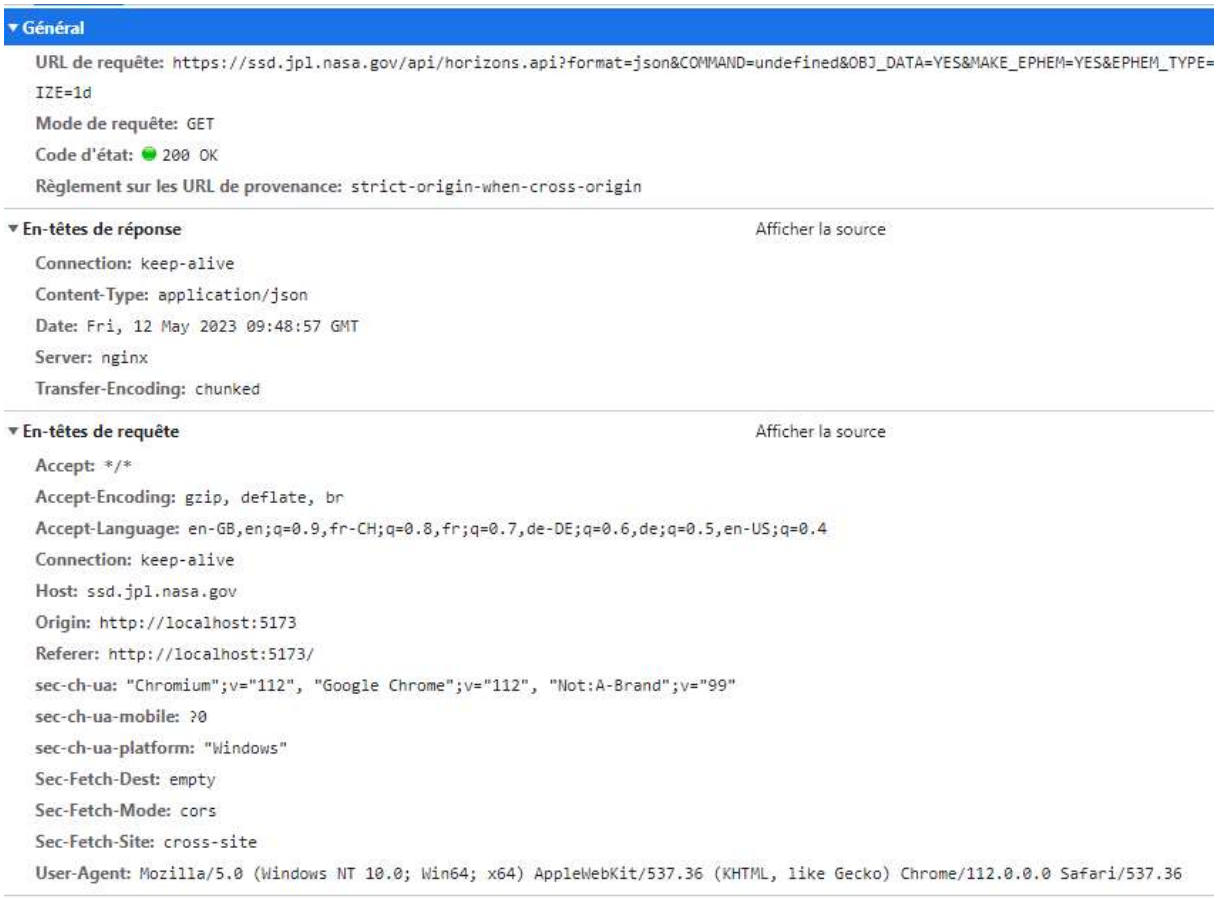


Figure 4 : Etat de la requête Horizon API sans données.

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 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.

4.1.1.2.1 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)


```
**********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)=~1600GeometricAlbedo=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 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.

```
const regexPlacment = /(-?\d+(?:\.\d+)?(?:E[+-]?\d+)?)Y=(-?\d+(?:\.\d+)?(?:E[+-]?\d+)?)Z=(-?\d+(?:\.\d+)?(?:E[+-]?\d+)?)/
const matchPlacment = texteSansEspaces.match(regexPlacment);

// Return data in object form
return {
  id : matchNameID[2],
  name : matchNameID[1],
  sizeRadius : matchMeanRadius[1],
  material : matchNameID[1] + ".png",
  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 7 : Exemple de l'utilisation de regex

4.1.1.3 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.

```
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())
        })
        .then(data => {
            resolve(data.data);
        })
        .catch(function (err) {
            resolve("Something went wrong!", err);
        });

    })
}
```

Figure 8 : Fonction de création fetch des objets proches.

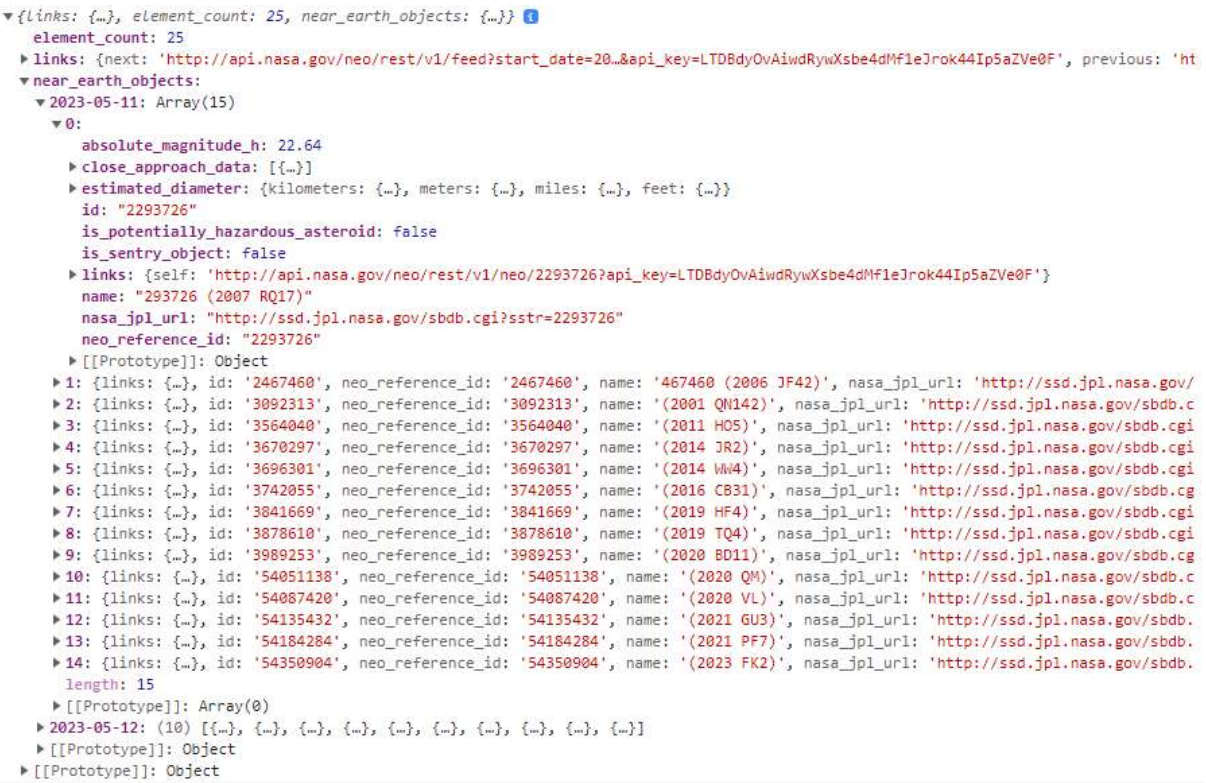


Figure 9 : Données récupérées via a l'api NEO.

4.1.1.4 Favicon

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

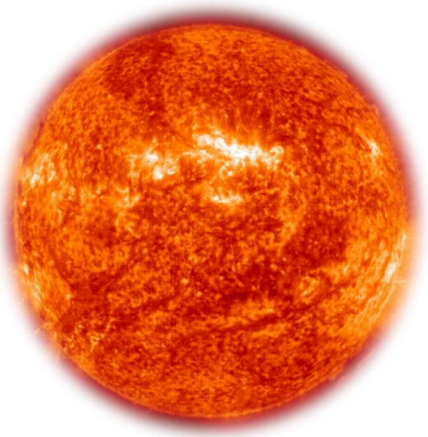


Figure 10 : Favicon

4.2 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
  -

4.3 Description des tests effectués

```
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);

})
```

4.4 Erreurs restantes

4.5 Liste des documents fournis

5 Conclusions

6 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

7 Table des illustrations

8 Lexique

R

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

9 Annexes

9.1 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

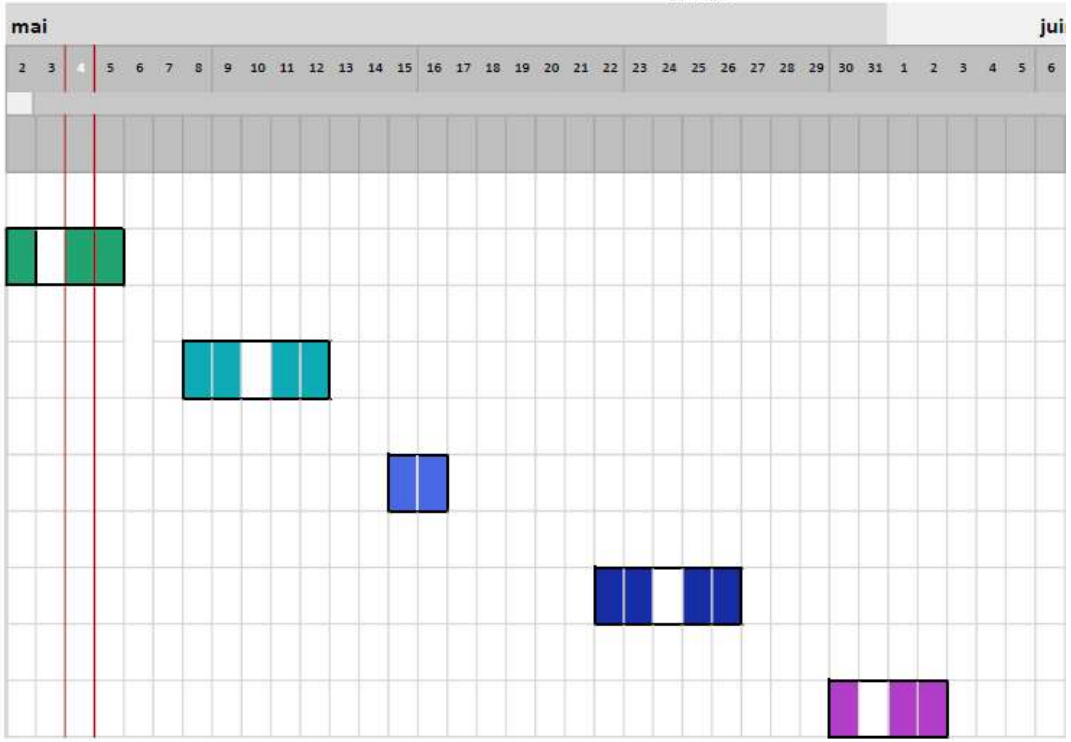
Incrément de défilement 0

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

Jours
Lundi
Mardi
Mercredi
Jeudi
Vendredi
Samedi
Dimanche

Horaire

P1	P2	P3	P4	P5	P6	P7	P8	P9

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.

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

9.3 Journal de travail

9.4 Archives du projet

Media, … dans une fourre en plastique