



Escola d'Enginyeria de Telecomunicació i
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA



Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

TRABAJO DE FIN DE GRADO

TÍTULO DEL TFG: Diseño de Web App para la visualización de satélites

TITULACIÓN: Grado en Ingeniería de Sistemas Aeroespaciales

AUTOR: José Triviño Ruiz

DIRECTOR: José Antonio Castán Ponz

FECHA: 7 de septiembre del 2022

Resumen

El objetivo del proyecto es el diseño de una aplicación web en la cual puedan visualizarse de forma intuitiva, didáctica y sencilla las diferentes características de los satélites que existen orbitando la tierra. Se trata de establecer una guía teórica y breves explicaciones sobre el funcionamiento de los mismos, así como la muestra de los mapas desde donde se pueda visualizar la información básica del satélite.

La principal fuente de información son los conocidos TLEs (*Two-Lines-Element*), un formato de datos con una lista de elementos para un objeto en un tiempo dado, que, junto al propagador (SGP4) proporciona todos los datos necesarios para la caracterización de las órbitas. La información de estos TLEs es oficialmente proporcionada de forma pública por la compañía *Celestrak* (*coworker* de *Space-Track*, proyecto inicialmente desarrollado por *US Air Force*). Además, la totalidad de componentes y librerías usadas en este proyecto son de uso público.

Existen multitud de funcionalidades que se han tratado de implementar en la medida de lo posible en la Web App: diferentes visualizados, filtrado de información, historia, comparaciones y cálculos, entre otras. Pero es el apartado de ‘familias’ de satélites el que se ha desarrollado *in extenso*, puesto que es el que tiene un especial interés de cara al uso de la página web en un futuro. Con la implementación de este apartado especial para las familias, se tratan campos como la cobertura, descripción de las órbitas o la ocupación del espacio usado.

Este proyecto no pretende más que aclarar el espacio exterior al usuario medio y ‘acercar’ la industria aeroespacial a aquellos que se interesen en el tema. Es por ello que se cataloga como un proyecto divulgativo, ya que, si bien es científico y trata cuestiones complejas, es para todos los públicos y no se requiere de profundo conocimiento previo para hacer uso del mismo. Para estudios más sofisticados y de elevada complejidad, existen herramientas más adecuadas que el proyecto que se plantea realizar.

Resum

L'objectiu del projecte és dissenyar una aplicació web on es puguin visualitzar de forma intuïtiva, didàctica i senzilla les diferents característiques dels satèl·lits que orbiten la Terra. Es tracta d'establir una guia teòrica i explicacions breus sobre el seu funcionament, així com de mostrar els mapes des d'on es pugui visualitzar la informació bàsica del satèl·lit.

La principal font d'informació són els coneguts *TLEs* (*Two-Lines-Element*), un format de dades amb una llista d'elements per a un objecte en un temps donat, que juntament amb el propagador (*SGP4*) proporciona totes les dades necessàries per a la caracterització de les òrbites. La informació d'aquests *TLE* és oficial i la proporciona públicament la companyia *Celestrak* (coworker d'*Space-Track*, projecte inicialment desenvolupat per *US Air Force*). A més, la totalitat de components i llibreries utilitzades en aquest projecte són d'ús públic.

Hi ha multitud de funcionalitats que s'han intentat implementar en la mesura del possible a la *Web App*: diferents visualitzats, filtratge d'informació, història, comparacions i càlculs, entre d'altres. Però és l'apartat de 'famílies' de satèl·lits el que s'ha desenvolupat *in extenso*, atès que és el que té un interès especial de cara a l'ús de la pàgina web en un futur. Amb la implementació d'aquest apartat especial per a les famílies, s'aborden camps com la cobertura, la descripció de les òrbites o l'ocupació de l'espai usat.

Aquest projecte només pretén aclarir l'espai exterior a l'usuari mitjà i 'apropar' la indústria aeroespacial a aquells qui s'hi interessin. És per això que es cataloga com un projecte divulgatiu: si bé és científic i tracta qüestions complexos, és per a tots els públics i no es requereix un profund coneixement previ per fer-ne ús. Per a estudis més sofisticats i de complexitat elevada, hi ha eines més adequades que el projecte que es planteja realitzar.

Abstract

The objective of this project is the design of a Web App where the main characteristics of the orbiting Earth satellites can be intuitively, didactically and easily visualized. It has been sought to establish a theoretic guide and brief explanations about how they operate, as well as a set of different maps where the basic information of the satellite can be available.

The main source of information are the well-known TLEs (Two-Lines-Element), a data format with a list of elements for a chosen object in a certain date, which, along with the propagator (SGP4) provides every necessary data for the orbit characterization. The information of these TLEs is officially and publicly provided by the *Celestrak* Company (coworker of *Space-Track*, a project initially developed by the *US Air Force*). In addition, every component and library used in this project are for public use.

Multiple functionalities were aimed to be integrated in the Web App: different views, information filtering, history, comparisons and calculations. But it is the 'families' section the one which has been developed extensively, since it can be the more attractive feature for the web users in the future. With this special feature, fields such as coverage, orbit description or the occupation of the space used are addressed.

This project intends to clarify the outer space to the average user and bring the aerospace industry closer to those who have interest in the subject. That is why this web is listed as a disclosure project, given that, although it is a scientific project and deals with complex issues, it is created for all audiences and no deep prior knowledge is required to make use of it. For more sophisticated and complex studies, there are more appropriate tools than the project that is planned to be carried out.

Agradecimientos

ÍNDICE

Resumen	I
Resum.....	II
Abstract	III
Agradecimientos	V
ÍNDICE	VII
ÍNDICE de tablas.....	X
ÍNDICE DE FIGURAS	XI
ÍNDICE DE ECUACIONES	XII
 CAPÍTULO 1: CONTEXTO	 1
1.1. Introducción	1
1.2. Stakeholders (partes interesadas).....	2
1.2.1. Desarrollador del proyecto	2
1.2.2. Director del proyecto	2
1.2.3. Usuarios finales de la aplicación	2
1.3. Estado del arte	3
1.3.1. Análisis de herramientas de visualizado de satélites existentes.....	3
1.3.2. Análisis de fuentes de información para satélites	4
1.3.3. Conclusión	5
 CAPÍTULO 2. OBJETIVOS Y LIMITACIONES.....	 6
2.1. Objetivos	6
2.1.1. Herramientas	6
2.1.2. Público	7
2.1.3. Funcionalidades extra	7
2.2. Limitaciones	8
2.2.1. De uso	8
2.2.2. Infraestructura.....	8
 CAPÍTULO 3. METODOLOGÍA	 10
3.1. Métodos de trabajo	10
3.2. Herramientas de validación.....	10
 CAPÍTULO 4. PLANIFICACIÓN	 11
4.1. Calendario	11
4.2. Recursos	11
4.2.1. Recursos humanos.....	12
4.2.2. Recursos materiales	13

4.2.3. Recursos de software	14
CAPÍTULO 5. RECOPIACIÓN DE UTILIDADES	15
5.1. Fase inicial.....	15
5.2. Entorno de desarrollo	16
5.3. Elección de herramientas y librerías	16
5.3.1. Mapa 2D	16
5.3.2. Mapa 3D	18
5.3.3. Cálculos	19
5.4. Documentación.....	20
5.4.1. Fundamentos matemáticos y científicos	20
5.4.2. Teoría de órbitas y TLEs	21
5.4.3. Satélites disponibles	27
5.5. Valoración final, gestión de alternativas y plan de acción.....	28
CAPÍTULO 6. PLIEGO DE CONDICIONES	29
6.1. Conocimientos adquiridos en la carrera como apoyo para el proyecto	29
6.2. Justificación de la elección del proyecto como adecuado para la titulación del grado de Ingeniería Aeroespacial, especialidad en Sistemas Aeroespaciales	30
6.3. Competencias del proyecto	30
CAPÍTULO 7. IMPLEMENTACIÓN	32
7.1. Lenguaje de programación	32
7.1.1. <i>HTML</i>	32
7.1.2. <i>CSS</i>	32
7.1.3. <i>JavaScript</i>	33
7.1.4. <i>JSON</i>	35
7.1.5. <i>MatLab</i>	35
7.2. Front-end.....	35
7.2.1. <i>ReactJS</i>	35
7.2.2. Framework <i>NextJS</i>	37
7.3. Navegador	38
CAPÍTULO 8. ESTRUCTURA DE LA WEB APP	39
8.1. Arquitectura	39
8.2. Capa de presentación	40
8.2.1. <i>Single Page Application (SPA)</i>	41
8.2.2. Ejemplos de diagrama de secuencia	42
8.3. Dominio	42
8.4. Datos.....	43
8.4.1. <i>API</i>	43
CAPÍTULO 9. DESCRIPCIÓN DE TAREAS.....	46
9.1. Requisitos funcionales	46
9.1.1. Disposición de satélites	46
9.1.2. Filtrado de información	47
9.1.3. Visualización de la traza de la órbita (2D y 3D)	48
9.1.4. Visualización de la órbita en 3D	51

9.2. Requisitos no funcionales.....	53
9.2.1. Estética.....	53
9.2.2. Accesibilidad.....	55
9.2.3. Sencillez	56
 CAPÍTULO 10. CONTENIDO.....	 58
10.1. Menú / Sidebar	58
10.2. Mapa 2D.....	58
10.2.1. Markers.....	58
10.2.2. Tooltips	58
10.2.3. Infobox.....	58
10.3. Mapa 3D.....	58
10.3.1. Traza.....	58
10.3.2. Órbita	58
10.3.3. Satélite.....	58
10.3.4. Infobox.....	58
10.4. Satélites.....	58
10.5. Documentación	58
10.6. About	58
 CAPÍTULO 11. PRUEBAS Y EJEMPLOS.....	 59
11.1. Tests	59
11.2. Ejemplos.....	59
11.2.1. Ejemplo de satélite en 2D.....	59
11.2.2. Ejemplo de familia de satélites 2D	59
11.2.3. Ejemplo de satélite en 3D.....	59
11.2.4. Ejemplo de familia de satélites 3D	59
 CAPÍTULO 12. CONCLUSIONES	 60
12.1. Conclusiones.....	60
12.2. Dificultades	60
12.3. Trabajos futuros	60
 CAPÍTULO 13. BIBLIOGRAFÍA	 61
 ANEXO I.....	 62
 ANEXO II.....	 63

ÍNDICE de tablas

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

CAPÍTULO 1: CONTEXTO

1.1. Introducción

El proyecto que en esta memoria queda reflejado, nace de la idea de hacer una plataforma online accesible para cualquier tipo de usuario donde se pueda interactuar con diferentes tipos de mapas y puedan verse las características principales de los satélites que hoy día orbitan la tierra.

Existen multitud de herramientas para poder acceder a esta clase de datos, pero el objetivo que caracteriza este proyecto es ensamblar un espacio de visionado donde la sencillez, minimalismo y estética prevalezcan. Es decir, la meta es que pueda ser usado por cualquier sujeto independientemente de si está familiarizado con el tema o no, y sin haber adquirido conocimientos previos sobre la ingeniería aeroespacial. Si bien se han estandarizado páginas web tales como *Flightradar24* o *Flightaware* en la población para la consulta de vuelos comerciales, no existe tal plataforma para el acceso y visionado de satélites. Es por ello que se pretende generar una web intuitiva, con funcionalidades que sean entendibles y manejables y se obtengan los resultados de forma clara.

Es de gran importancia aclarar que, pese a que la intención del proyecto sea la dicha, será una herramienta igualmente útil para las personas especializadas en el tema, ya que las diferentes opciones pueden ser realmente interesantes en estudios exhaustivos sobre el campo. Además, el hecho de que prevalezca la sencillez, hace que sea incluso más fácil de manejar que herramientas de mayor complejidad, aun siendo estas más potentes, lo cual puede suponer una ventaja importante en iteraciones repetitivas y trabajos que requieran ahorrar tiempo y maniobrabilidad.

La aplicación web que aquí se muestra se basará en los conocidos TLEs (*Two-Line-Element*), un formato de datos estandarizado en la industria aeroespacial para determinar los parámetros espacio-temporales correspondientes a los satélites. Dicho formato será tratado y explicado de forma extensa en este proyecto, puesto que es de gran importancia tener el concepto claro para entender cómo funciona la plataforma y la manera que tiene de representar órbitas y trazas.

Cabe resaltar la intención de generalizar y promover las herramientas de código abierto y de uso libre por cualquier usuario, que impulsan proyectos en todo tipo de campos y pueden tener resultados sorprendentes. Gran cantidad de proyectos no podrían haberse llevado a cabo o incluso no podrían haber siquiera nacido de no ser por esta filosofía tecnológica, por lo que es de destacar y remarcar el uso de estas prácticas.

Si se hace una búsqueda exhaustiva por la web y en programas tanto de escritorio como para móvil, es posible encontrar algunas aplicaciones en las que se puede tener acceso a gran parte de la información que se muestra en este proyecto, pero todas ellas cumplen que son complejas, de pago, limitadas o nada intuitivas. Además, el factor común es que son plataformas desactualizadas y

nada vistosas para el usuario. Aquí es donde entra en juego este proyecto de 'Satellite Viewer', así llamado de aquí en adelante para referirse al trabajo que en esta memoria se desarrolla.

1.2. Stakeholders (partes interesadas)

En este apartado se trata de esclarecer las personas que están (o estarán) involucradas en el proyecto, para determinar las partes que suponen el grupo remitente y destinatario, así como para poner sobre la mesa las intenciones de cada uno.

1.2.1. Desarrollador del proyecto

Este proyecto se desarrolla enteramente por el precursor de la idea del mismo, el autor del TFG y quien asume todas las tareas de programación, diseño, análisis, implementación, investigación y recopilación de información. El interés del desarrollador es realizar un trabajo que guste al público y que sea totalmente entendible para el usuario, tenga éstos conocimientos sobre satélites y mecánica orbital o no.

1.2.2. Director del proyecto

Mientras que el desarrollador asume la carga del trabajo, el director del proyecto José Antonio Castán tiene la función de supervisar, aconsejar y acondicionar este trabajo para que el resultado sea idóneo, aportando su experiencia tanto didáctica como en el mundo de la ingeniería, siendo sus intereses que el alumno obtenga los resultados más fructíferos y que el trabajo realizado sea vistoso y bien reconocido de cara a la evaluación.

1.2.3. Usuarios finales de la aplicación

La plataforma *Satellite Viewer*, tal y como se ha repetido con anterioridad, está pensada para toda clase de usuario interesado en el campo aeroespacial de los satélites y la mecánica orbital. Tiene especial interés para aquellos que quieran hacer algún tipo de estudio/investigación sobre posicionamiento, estado y comportamiento de las familias más numerosas en las órbitas terrestres (*Starlink*, *Iridium*, *GNSS*, *Intelsat...*), además, se tratará de hacer lo más profesional posible para que tenga valor científico y sea evaluado por el tribunal del proyecto.

1.3. Estado del arte

Es importante hacer una revisión general en la situación actual del tema y su accesibilidad. Se parte de la base de que existen programas y aplicaciones que cumplen gran parte de las funciones recogidas en el trabajo, pero es justo la parte diferencial lo que se ha de destacar (encontrar el hueco en el mercado y el porqué del proyecto).

1.3.1. Análisis de herramientas de visualizado de satélites existentes

Es inconcebible hablar del mundo aeroespacial, satélites y mecánica orbital sin hacer una mención especial y honorífica al software de AGI conocido como *STK* 12. Este es, sin duda, el más completo de los existentes en el campo, y tiene aplicaciones educativas y profesionales, cursos disponibles, infinitas funcionalidades y su complejidad abarca desde lo más sencillo del campo hasta los cálculos y análisis más laberínticos. La herramienta dispone de una versión gratuita durante 14 días y además es posible tener una licencia si así se solicita con motivos educativos tras presentar la debida documentación que lo justifique.

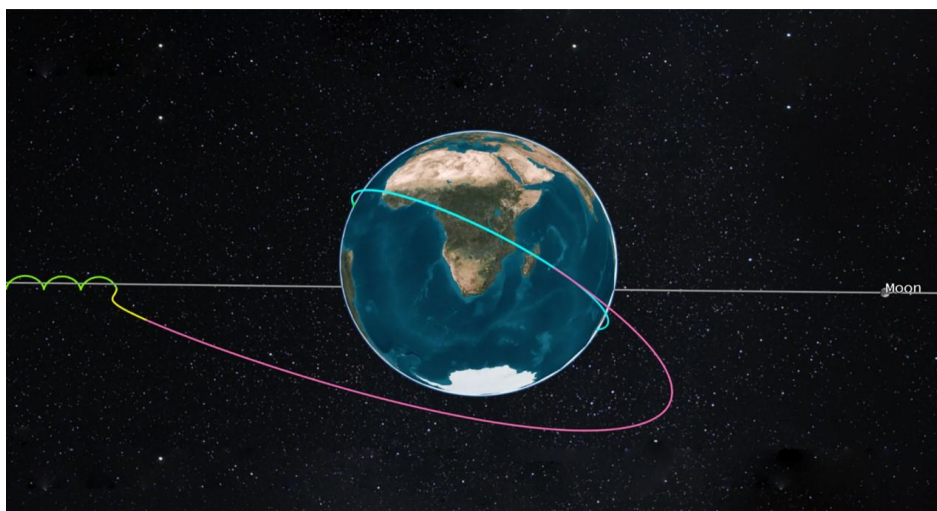


Fig. 1.1 Ejemplo de *STK Astrogator*, AGI [1].

El inconveniente de este software recae precisamente en que se trata de una herramienta que no es para nada accesible para el usuario medio, una de las bases fundamentales de este proyecto. Si bien no se trata de hacer una herramienta tan potente ni compleja, en el proyecto aquí expuesto se cubre ampliamente mejor el tema de la facilidad, rapidez y estética, de las cuales carece. Y es que no se trata sólo de que el software requiera un potente hardware, espacio digital, licencia y un amplio conocimiento de mecánica orbital,

sino que, además, no es posible acceder al mismo desde una página web, lo que limita mucho el uso.

Además de este software, existen otras alternativas que satisfacen superficialmente parte de las intenciones de *Satellite Viewer*. De hecho, la propia compañía de la que se hace uso de la información pública de los TLEs (*Celestrak*) posee su propia herramienta de filtrado y de visionado de satélites y órbitas. Una vez más, el problema de este es que no es nada familiar para el usuario que no tenga conocimientos sobre el tema, y, el visionado, aparte de no ser altamente agradable visualmente, dejará de estar disponible a partir del 1 de septiembre del 2022.

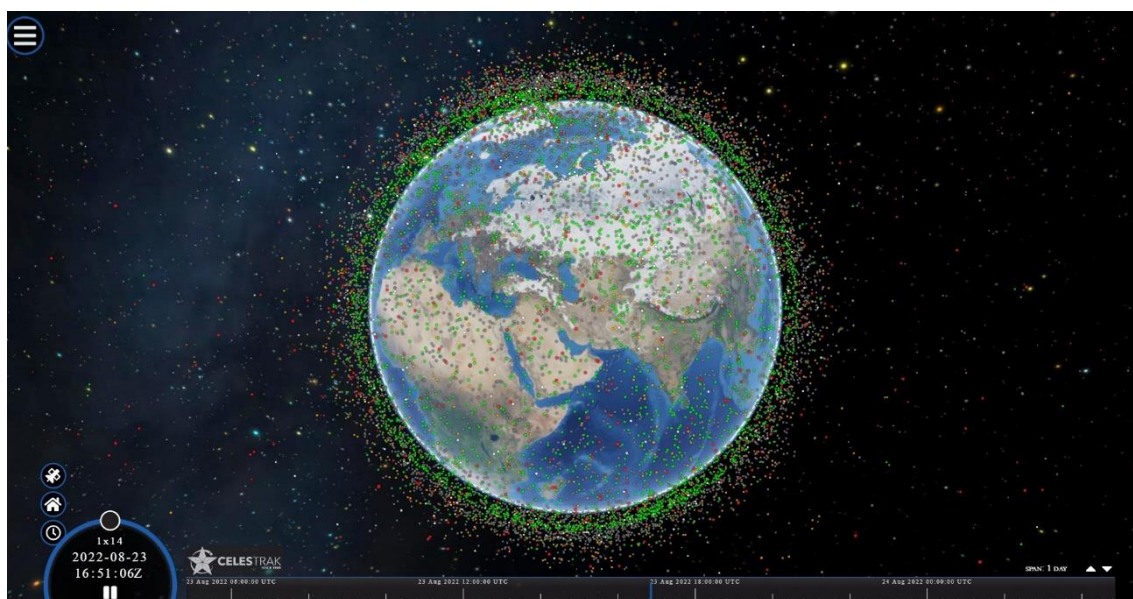


Fig. 2.2 Ejemplo de visualizado en *Celestrak* [2].

Cabe destacar que esta página web ha sido la base fundamental en la cual se ha apoyado este proyecto, ya que la documentación aquí recogida es extensa, y clara, y está dotado de multitud de ejemplos. Si bien no se trata de hacerle competencia, ha sido una gran inspiración y se ha tratado de satisfacer las funcionalidades de las que carecía esta plataforma.

1.3.2. Análisis de fuentes de información para satélites

Mientras que las alternativas para el visualizado del panorama satelital no son muy abundantes, existen multitud de fuentes de información para completar la documentación necesaria para generar el software. Este proyecto tiene una fuerte carga matemática y requiere fundamentos físicos referidos a la mecánica orbital para una comprensión íntegra, por lo que es de vital importancia para la plataforma tener una base teórica competente, fiel a la realidad y comprensible.

Uno de los problemas a la hora del filtrado de información es precisamente el caso en el que las fuentes son numerosas y, por tanto, no es fácil elegir la adecuada. En este caso, siguiendo la línea de trabajo establecida inicialmente, se procederá a usar la página proveedora de TLEs (*Celestrak*) como guía primaria para la estructuración de la documentación, ya que tiene fácilmente accesible toda la información recopilada sobre los campos de TLEs, mecánica orbital, historia satelital o incluso referencias a procesos que están aún en marcha.

Esta plataforma es escogida como ejemplo y base debido a algunas características que la hacen completamente única [3]:

- Ha estado en funcionamiento desde 1985, fecha en la cual nació el proyecto y momento en el cual era usado para proporcionar de forma pública la información de los satélites no clasificados. Esta función la hacía mediante los *NASA Prediction Bulletins*¹, que eran recibidos de primera mano de la administración para la publicación en esta plataforma.
- Nace como proyecto divulgativo (*Bulletin Board System*²), cuyo objetivo era la publicación de información satelital y astronómica. El fundador sirvió durante más de tres décadas para el ejército del aire de los Estados Unidos y tenía, además de la cualificación necesaria, la documentación y apoyo del organismo clave para desarrollar el proyecto.
- No sólo cuenta con todo el apoyo de la comunidad, sino que se establece como fuente principal de obtención de datos satelitales a escala global. La plataforma cobró más importancia a raíz de la desaparición de *Satellite Times*, y es, sin duda, la mejor opción en cuanto a filtrado y búsqueda.

Por todas estas razones, Celestrak se encuentra en el punto de mira en el campo aeroespacial y está perfectamente justificada la estructuración y construcción en base a su conocimiento, divulgación y contenido.

1.3.3. Conclusión

Una vez se ha esclarecido el estado del arte del proyecto, cabe resaltar el hecho de que todo esto está basado en una gran recopilación de información y una elección personal del desarrollador. Es decir, podrían haberse tomado otros caminos igualmente respetables y válidos para construir *Satellite Viewer*, pero esta ha sido la vía escogida.

CAPÍTULO 2. OBJETIVOS Y LIMITACIONES

2.1. Objetivos

Puesto que un proyecto carece de sentido sin objetivos iniciales, en este apartado están recogidas las metas que se tratan de alcanzar en la puesta a punto de esta aplicación web. Si bien todas ellas son importantes, no dejan de ser, como se ha mencionado, metas. En todo proyecto existen objetivos iniciales que pueden estar sujetos a cambios, mejoras o incluso cancelaciones, por lo que esto consiste realmente en una recopilación de las características buscadas inicialmente en la aplicación.

2.1.1. Herramientas

Tal y como el título del proyecto reza, la página deberá estar dotada de herramientas de visionado, objetivo primordial y primario de este proyecto. La intención es hacer un visor estético, intuitivo y agradable al usuario, así como potente y rápido a la hora del renderizado de información seleccionada.

Dichas herramientas de visionado serán de dos clases: mapa 2D y mapa 3D. Ambas tendrán las mismas funcionalidades y serán directamente accesibles desde la pestaña de inicio de la página web, resaltando así su importancia y haciendo ver al usuario que es el motivo por el que la web es creada.

En cuanto a las funcionalidades esenciales que se plantean incluir, están la de visionado del satélite en un tiempo dado (generalmente en el momento en el que se haga la petición), la caracterización de la posición (latitud, longitud, altitud), de la órbita [4] (semieje mayor, excentricidad, inclinación, ascensión recta del nodo ascendente, argumento del perigeo, anomalía verdadera) y una breve indicación sobre el satélite o la familia de satélites en cuestión, ya sea la identificación NORAD [5] o el nombre estandarizado.

Además del apartado gráfico, existen dos cuestiones que se consideran como objetivos esenciales en este trabajo: el filtrado y exposición de los cuerpos que orbitan la tierra (apartado Satélites) y la documentación (apartado de Documentación).

En cuanto al apartado satélites, se pretende mostrar al usuario de forma clara y breve algunas de las familias más importantes que existen en la actualidad, los la clasificación de los mismos en función de los tipos de órbita, propósito o fecha, y algunos ejemplos visuales de estos mostrados mediante las herramientas de visualizado 2D y 3D.

Por último, la documentación recogerá las publicaciones más relevantes en cuanto a la teoría satelital esencial se refiere, así como teoría complementaria sobre mecánica orbital, historia aeroespacial e incluso esta misma memoria, que pueda servir como ilustración de cómo la plataforma ha sido creada, las

funciones que tiene, cómo se han de emplear y qué objetivos y limitaciones recoge.

2.1.2. Público

La plataforma será creada en local por el desarrollador y mostrada al tribunal de evaluación del Trabajo de Fin de Grado y su publicación recaerá únicamente en la decisión del desarrollador, siendo su criterio el decisivo a la hora de la salida al público de esta. Es importante destacar que el proyecto se desarrolla en un marco legal donde toda la información es pública, las librerías son de uso público y todo aquello que conforma el proyecto es de carácter público, por lo que en cualquier momento está preparada para ver la luz, no siendo la legalidad un problema para esta.

Una vez lanzada, si es que esto ocurre, la intención es meramente didáctica y divulgativa. Este proyecto desea acercar la industria aeroespacial a aquellos que así lo desean y mediante un software sencillo de usar. Por lo que los objetivos relacionados con el público se centran en la satisfacción de las inquietudes que el consumidor tenga relacionadas con este campo.

2.1.3. Funcionalidades extra

En el capítulo 1 se mostraban algunas de las razones por las que este trabajo se ha puesto en marcha, por lo que existe justificación más que de sobra como para crear el proyecto. No obstante, existen una serie de funcionalidades que no son fácilmente accesibles en otros softwares y que se tratará de implementar en la medida de lo posible para hacer destacar aún más su valor de cara al público.

En primer lugar, no es trivial hacer la comparativa de ejes inerciales y no inerciales en la representación de las órbitas, puesto que estas requieren de altas capacidades de visión espacial y un fuerte conocimiento físico. Desde el lado del desarrollador y como experiencia personal, se desea implementar una funcionalidad de comparativa entre una representación y otra, que deje totalmente clara la diferencia entre ambas y que ayude a la comprensión.

La cobertura es un parámetro que interesa globalmente a personas relacionadas con el campo aeroespacial y personas no relacionadas con el mismo, por lo que es una rama que tiene interesantes salidas en este proyecto y que podría acabar desarrollando toda una nueva plataforma de cara al futuro de este proyecto, por lo que se tratará de implementar en la medida de lo posible.

Por último, y, también relacionado con la cobertura, se pretende hacer alusión a los satélites que conforman el sistema ADSB¹ (constelación GNSS [6]) e implantar apartados en los que se pueda interactuar con los mismos, de tal forma que se integre la industria aeronáutica y aeroespacial en la misma plataforma.

1. A D Survie B: sajfsadjalsdjlwjd [6]

2.2. Limitaciones

No es hasta que quedan esclarecidos los objetivos fundamentales que comienzan a ser más evidentes las limitaciones que conforman este trabajo. Y es que, las intenciones se proyectan de forma ambiciosa y genuina, y es en el futuro (tras la aparición de contratiempos y adversidades) cuando la realidad se ve totalmente plasmada y es posible la visión global de la plataforma, así como sus limitaciones. No obstante, hay parámetros controlables y limitaciones bien acotadas que se pueden apuntar desde el más primitivo de los comienzos.

2.2.1. De uso

Pese a que la aplicación se construya en las mejores condiciones posibles y dentro de los conocimientos más avanzados que el desarrollador conoce, no es sencillo estimar la magnitud del público al que irá dirigido en un futuro. Esto supone una limitación de uso que viene marcada por la vida útil y actualización de las librerías, la consistencia de la página a numerosas entradas simultáneas, las adaptaciones futuras y el apoyo que reciba.

La falta de experiencia por parte del desarrollador puede comprometer sustancialmente el producto final de este proyecto, siendo crucial la constante consciencia de la meta que se desea obtener y los caminos que se han de recorrer. Sin embargo, existen multitud de complicaciones que pueden ocurrir en este largo recorrido: falta de conocimientos de programación, errónea elección de librerías, implementación de cálculos inexactos, lentos procesados, etcétera. Todas estas situaciones tienen solución, pero pueden suponer un fatídico resultado si no se tienen continuamente en cuenta.

Pese a que se cree con la intención de que llegue al mayor número posible de personas, se parte de la base de que ya existen softwares extremadamente potentes y estandarizados en el panorama, tal y como se explicó en el apartado 1.3.2. de esta memoria. Eso supone que, aunque se haya diseñado para suplir carencias de otras alternativas, se es consciente de la superioridad estructural y la abundancia de funcionalidades con las que cuentan. En cuanto a herramientas se refiere será una plataforma útil y rápida, pero habrá gran cantidad de usos que no estarán cubiertos, lo que supone una limitación clara.

2.2.2. Infraestructura

Aunque se haga continuamente referencia a alternativas de este proyecto, no se puede obviar el hecho de que estos productos vienen de la mano de grandes corporaciones y de compañías que cuentan, entre otras cosas, de numerosos empleados, dispositivos, hardware y apoyo de otras entidades. Como se ha repetido en algunas ocasiones, no se desea competir con ellas, pero no tendría sentido no mencionar el hecho de que este proyecto se está desarrollando por sólo una persona (el desarrollador de este proyecto).

Además de las limitaciones de personal, y lo que supone la reducción de la creación de toda esta infraestructura a una sola persona, existen limitaciones desde la propia máquina que sea responsable de procesar todo este software, hasta la falta de conocimiento en campos de estética, captación del usuario, informática avanzada, base de datos robusta, cliente siempre en línea, entre otras.

La propia aplicación web es desarrollada en local para su total composición y sólo en el caso de que el desarrollador así lo estime o le sea requerido, será lanzada a dominio público en internet. En el momento en el que esto ocurre, nace la necesidad de comprar/alquilar un dominio web, pagar la base de datos y servidor, mantener las librerías que dependen de las visitas (librerías ‘gratuitas’ como las de *Google Maps* no suponen coste alguno para pequeños desarrolladores, pero empiezan a ser de pago exponencial a medida que se aumenta el uso de la misma) y hacerse cargo de la seguridad e integridad de la página, que queda expuesta a posibles ataques cibernéticos. Todo esto establece una limitación muy comprometedora cuando es sólo una persona trabajando en esta infraestructura, razón principal por la que se crea desde el primer momento en local.

CAPÍTULO 3. METODOLOGÍA

3.1. Métodos de trabajo

La aplicación se desarrolla enteramente en el entorno *Visual Studio Code*, software del que se habla más extensamente en el apartado de recursos del capítulo 4, y se utilizará GIT [7] como sistema de control de versiones. Más concretamente, la web *GitHub*, un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y comprado por *Microsoft* en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo sea posible descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo [8].

Los cambios y versiones que se incorporan en la aplicación son enviados y debidamente supervisados por el director del proyecto, para confirmar que son justificados y verificados, así como forma de aportación de segundas opiniones sobre el trabajo realizado.

Por razones evidentes, la metodología seguida es la satisfacción inmediata de los objetivos indispensables del proyecto, su verificación y testeo, y, por último, la introducción en el mismo de las funcionalidades denominadas como 'extra'. Las cuales se desean incorporar a la plataforma, pero no son de crucial relevancia en el completo desarrollo, y que han sido así caracterizadas bien por su menor importancia o por la gran complejidad que éstas albergan.

3.2. Herramientas de validación

Los cambios y modificaciones de la aplicación conllevan su correspondiente validación y testeo, pero esto no siempre es completamente suficiente, la validación es crucial para la 'salud' de esta plataforma y suponen un resultado óptimo a largo plazo. Por esta razón, se hace un proceso exhaustivo de validación que consta de la íntegra interacción con la aplicación para la comprobación de ningún error directo o indirecto en la misma, y un chequeo de la terminal del compilador y de la consola del navegador al final de la resolución.

CAPÍTULO 4. PLANIFICACIÓN

4.1. Calendario

El proyecto se ha planificado para una duración estimada de aproximadamente tres meses y medio. La idea surge a mediados de mayo, fecha en la cual comienza la puesta en marcha (aunque es oficialmente matriculado y formalizado a fecha del 4 de julio, fecha en la que acabaron los exámenes finales y se puede hacer la debida reunión). Por otro lado, la fecha de entrega se sitúa antes de la fecha límite de depósito de TFG, establecida como el 8 de septiembre del mismo año. El comentario más importante a resaltar en cuanto a esta planificación es que el tiempo es altamente limitado y el proyecto requiere de una dedicación absoluta en estos meses establecidos, esa es la razón por la que se han escogido estas fechas.

La planificación inicial del proyecto viene recogida en la figura 4.1. y, si bien es inicial y puede cambiar radicalmente, intenta plasmar las fechas de dedicación diseñadas, y los tiempos pensados para cada una de las partes.

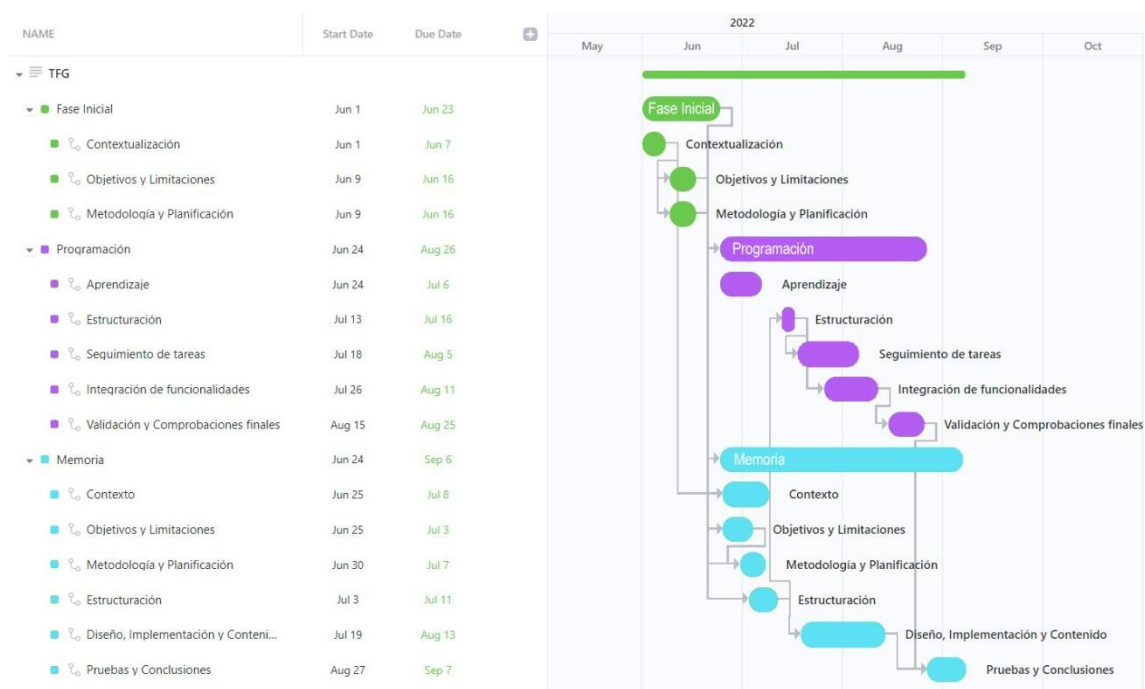


Fig. 4.1 Diagrama de Gantt del proyecto.

4.2. Recursos

En este apartado se pretende hacer una exposición de los bienes materiales, de los que se dispone al comienzo del proyecto. Es fundamental ser consciente de

las condiciones de las que se parte y de los recursos con los que se puede contar.

4.2.1. Recursos humanos

Como se menciona en apartados anteriores, la totalidad del proyecto se desarrollará por parte de una sola persona, el desarrollador, por lo que carece de sentido realizar un desarrollo extenso. Sin embargo, este mismo producto desarrollado profesionalmente por una compañía habría requerido de un personal más numeroso, puesto que los campos de la legalidad, puesta a punto, marketing, diseño y administración quedan limitados o reducidos en esta situación.

No obstante, se cuenta con el apoyo del director del proyecto, que es el encargado de supervisar y garantizar que *Satellite Viewer* es conformado de forma óptima.

En un entorno profesional de programación, se habría contado con el apoyo de analistas, arquitectos de software, programadores, testers y un jefe de proyecto al cargo de todo esto, lo que habría supuesto un aumento considerable en los costes del mismo. De forma resumida y, siguiendo los sueldos del estudio [9] de 2020 de LinkedIn (basado en miles de puestos de trabajos en el sector), se ha generado este presupuesto.

Tabla 4.1 Estimación de los costes del desarrollo del producto.

Tarea	Tiempo (h)	Personal	Sueldo (€/h)	Total (€)
Fase Inicial	126			3,528
Contextualización	42	Jefe de Proyecto	28	1,176
Objetivos/Limitaciones	42			1,176
Metodología/Planificación	42			1,176
Programación	336			4554
Estructuración	72	Arquitecto/Analista	17	1,224
Seguimiento de tareas	108	Programador	12.5	1,350
Int. de funcionalidades	96			1,200
Validación/Comprobación	60	Tester	13	780
Proyecto	462			8,082

Este análisis de coste estimado, refleja un supuesto coste total de 8,082 euros, donde puede apreciarse que sólo se ha tenido en cuenta el departamento de programación. En el momento en el que se añadiera personal de marketing, diseño y marco legal, el presupuesto sería radicalmente distinto y, evidentemente, también los resultados.

4.2.2. Recursos materiales

Para este subapartado, se puede reducir el listado de recursos meramente al dispositivo utilizado para la programación y desarrollo de la aplicación (recursos de hardware utilizados), el cual se trata de un ordenador portátil *MSI GS66 Stealth*. Dicho portátil, además, se usa con un monitor externo que será tenido en cuenta para el cálculo y desglose de costes, tanto de amortización como de consumo.

El portátil y el monitor externos serán integrados en el cálculo de los costes del proyecto, de la siguiente forma:

$$Amortización = \frac{Precio\ total}{Vida\ útil} = \begin{cases} \frac{1000€}{4\ años * \frac{250días}{año} * \frac{8h}{día}} = 0.13€/h \\ \frac{200€}{10\ años * \frac{250días}{año} * \frac{8h}{día}} = 0.01€/h \end{cases} \quad (4.1)$$

Con estos costes de amortización [10], se obtiene un total de 0.14€/h por el ordenador y el monitor externo, que, con las horas estimadas de proyecto establecidas en 462, puede obtenerse un coste total por un valor de 64,68€.

No obstante, estos no serán los únicos costes, puesto que los costes indirectos relacionados con el consumo y otros asuntos deben tenerse también en cuenta, lo cual viene recogido en la siguiente tabla.

Tabla 4.2 Estimación de los costes indirectos del desarrollo del producto.

Recurso	Cantidad	Precio	Total
Internet	3 meses	20 € / mes	60€
Ordenador	0.13kW*462h	0,567 €/kWh	34,03€
Monitor	0.08kW*462h		20,94€
Luz	0.1kW*462h		26,18€
Transportes	5 viajes facultad	1,5 €/viaje	7,5€
Impresión	3x90 carillas	0,05€/carilla	13,5€
Total			162,15€

Los costes [11] totales en el subapartado correspondiente a los recursos materiales ascienden a un total de 226,83€.

4.2.3. Recursos de software

Una de las características principales que se exigió como requisito en la idea de desarrollar esta plataforma es la de crear todo en base a software libre, con acceso global y dotado de librerías públicas, por lo que el coste relacionado al software de *Satellite Viewer* podría limitarse a las licencias adquiridas para el portátil, que se resumen en el Sistema Operativo y programas de la familia Adobe. Por suerte, la licencia de Windows está contenida en la amortización del ordenador, debido a que éste venía incluido en el dispositivo. Por otro lado, los programas de Adobe ofrecen licencias gratuitas para estudiantes, ya sea por parte de la universidad o de forma independiente con las justificaciones necesarias, por lo que ambas pueden no considerarse en este subapartado.

El software en el que se crea el proyecto es casi en su totalidad *Visual Studio Code*, una herramienta gratuita al servicio de cualquier usuario interesado, así como el navegador *Google Chrome*, también gratuito y altamente estandarizado.

CAPÍTULO 5. RECOPIACIÓN DE UTILIDADES

5.1. Fase inicial

Una vez contextualizado el proyecto, la recopilación de utilidades supone una pieza clave para el correcto desarrollo del mismo. Trabajar de forma ordenada y pautada es fruto de una buena organización inicial, y la determinación de las fases de forma escalonada favorece el reparto equitativo de carga de trabajo y que la línea crítica del proceso no se sature.

En la fase inicial de este trabajo se pretende generar la contextualización, los objetivos y limitaciones y la metodología y la planificación, tal y como viene descrito en el Diagrama de Gantt (Figura 4.1). Para ello, es necesaria la investigación sobre el estado del arte en primer lugar, y la comparación con el panorama actual. Una vez se ha hecho un estudio exhaustivo sobre el panorama existente, es cuando se puede apuntar las bases del proyecto, las razones por las que surge y los puntos fuertes que le hacen destacar sobre el resto. En definitiva, es crucial la ubicar la situación de la que se parte.

Aunque los objetivos del proyecto (o parte de ellos) son incluso generados antes de la puesta en marcha (debe existir un motivo inicial para comenzar éste), no son hasta que el contexto está enteramente definido que el listado de objetivos puede establecerse totalmente. De forma paralela aparecen las limitaciones: a través del desarrollo del producto se ven de forma detallada las limitaciones que lo caracterizan, con la diferencia de que incluso más limitaciones pueden aparecer a posteriori.

Las pautas que se han de llevar a cabo son las siguientes:

- Investigación exhaustiva sobre software aeroespacial
- Recopilación de conocimientos necesarios para realizar el trabajo
- Comparativa entre alternativas posibles
- Validación y comprobación de documentación veraz
- Filtrado de documentación relevante
- Generación de un orden de actuación en el desarrollo del producto
- Estudio de las limitaciones iniciales del proyecto
- Establecimiento de una metodología común
- Seguimiento de la planificación y fidelidad a la metodología diseñada

Si bien algunas pautas pueden llegar a parecer demasiado obvias, es demasiado común la vulneración de las mismas, y el resultado es el mismo para todas: ralentización del proceso, pérdida de eficacia y deterioro de la calidad del proyecto.

La intención del desarrollador del proyecto es tener las proyecciones en diferentes dimensiones de las órbitas satelitales, puesto que es el punto fundamental de esta plataforma, no obstante, es evidente que no se puede

empezar por esa parte sin antes generar una estructura de la página, una elección correcta del entorno del usuario y un mínimo de la línea estética.

5.2. Entorno de desarrollo

Para comenzar a construir la plataforma, el primer paso es la puesta a punto de todo el entorno de programación. Tal y como en apartados anteriores se manifestó, el software del que se hará uso para la generación del todo el código será el ampliamente conocido *Visual Studio Code*, ‘un editor de código fuente desarrollado por *Microsoft* para *Windows*, *Linux*, *macOS* y *Web*. Incluye soporte para la depuración, control integrado de *Git*, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código’ [12]. La razón de la elección de este editor es sencilla: es muy completo, admite todos los lenguajes existentes, es sencillo de instalar y configurar y es gratuito. Es por ello que en el mundo de la programación es el más usado y estandarizado.

Tras la instalación (versión 1.70.2) de este editor de texto, se hace también uso de un paquete de personalización de texto llamado ‘*Prettier*’, el cual no es más que un generador de formato para el código: establece una serie de colores, espacios y estilos para el código generado, haciéndolo más estético y fácilmente entendible para el usuario. Además, se escoge un tema deseado por el desarrollador (en este caso *Night Owl* y *Dark+*), que fomentan la concentración al programador y ayudan a combatir los problemas visuales derivados de la luz constante y las horas prolongadas de trabajo frente a pantallas.

Con la completa instalación y configuración del editor es más que suficiente para comenzar a generar código para la aplicación, pero es importante estar familiarizado con el entorno de la consola, que puede revisarse en la parte inferior del programa o en la consola del navegador que se use para programar la web app.

5.3. Elección de herramientas y librerías

Como primera aproximación, se han de esclarecer las herramientas que se necesitarán para la generación de *Satellite Viewer*, para lo que se debe elegir, en ocasiones, si bastará con la elección de una herramienta/librería existente o si será necesario generar alguna propia. Es por esta razón por la que se crea este apartado, donde se tratará de esclarecer todo aquello que será usado en algún punto de la programación.

5.3.1. Mapa 2D

La primera visualización disponible en la página web es la de un mapa en dos dimensiones donde las coordenadas latitud y longitud deben ser fácilmente

representadas. Por esta razón se ha de introducir un mapa en la pestaña que sea lo más visual posible. Una vez recopiladas las opciones más importantes en base a las características que el desarrollador busca, se exponen las opciones que se han barajado de algún modo u otro para hacer uso en esta plataforma:

- **Leaflet.** Librería caracterizada por la ligereza, la sencillez de su implementación y programación y el soporte del que dispone para dispositivos móviles. Se trata de una seria opción debido a que la ligereza en la página es algo crucial para la interacción usuario-página. No obstante, su dependencia con el uso de *plugins*, el soporte de sólo un par de sistemas de referencia y la excesiva simpleza del mapa hacen que existan opciones mejores.
- **ArcGIS API.** Librería creada por *Esri*, empresa internacional dedicada especialmente al soporte de software sistemas geográficos [12]. Está dotada tanto de mapas 2D como de 3D, además de ser gratuita sujeta a determinados usos. Es una opción interesante, especialmente por el hecho de poder usarla para los dos tipos de representación, pero sus limitadas extensiones gratuitas y el acceso restringido a funcionalidades de interacción con el mapa limitan mucho las ilustraciones que se desean.
- **MapBox GL JS.** Inicialmente, el mapa escogido para el desarrollo del proyecto, no sólo porque es muy completo, fácilmente personalizable y sencillos de usar, sino porque también se trata de un mapa cada vez más elegido por desarrolladores, del que existe gran cantidad de documentación y ejemplos, lo cual facilita aún más su programación y configuración.



Fig. 5.1 Ejemplo de proyección personalizada en *MapBox* [13].

El inconveniente de esta librería es la limitación de la versión gratuita, que se estima escasa para las proyecciones deseadas de la plataforma.

- **Google Maps API.** Finalmente, la opción más completa, estandarizada y totalmente gratuita para pequeños proyectos [14]. Pese a que esté basado en un servicio de pago por uso, la compañía estadounidense está comprometida con el concepto de software libre y continuamente crea proyectos para desarrolladores y que el campo siga avanzando en el camino correcto. Al ser una aplicación de pequeña escala, su uso es totalmente gratuito y con todas las funcionalidades (en el mapa 2D), por lo que lo convierte en el candidato perfecto para ser usado como proveedor de mapas del proyecto. Además, no sólo cuenta con una cantidad inmensa de proyectos y ejemplos en la web, sino que tiene una propia plataforma creada para desarrolladores donde explica en detalle los usos e implementaciones de todas las herramientas que lo conforman.

Tras la recopilación de información y la elección del mapa a usar, las tareas a llevar a cabo son la de implementación de marcadores en ubicaciones concretas, la ilustración de trazas, la personalización del mapa y la interacción con el usuario, todo ello detallado en posteriores capítulos.

5.3.2. Mapa 3D

Al igual que se ha hecho un cribado para la obtención del proveedor de mapas en dos dimensiones, se debe hacer lo propio para el proveedor de mapas en tres dimensiones. La diferencia en este caso es que no hay tantas alternativas y, si sólo se tienen en cuenta aquellas gratuitas, existen aún menos. Sin embargo, en este subapartado se pretende mostrar las opciones que se han valorado y el porqué de la elección de una en concreto, además de las tareas que ha conllevado y las que le suceden.

- **Cesium.** Librería de código abierto que se sitúa dentro de las mejores librerías de *JavaScript* para crear mapas web [14]. Tiene infinidad de funcionalidades, usa *WebGL* para mostrar los gráficos en 3D y está diseñado para ser operativo en todas las plataformas. En los inicios del proyecto se ha hecho uso de esta librería como la principal proveedora de mapas en tres dimensiones, pero el inconveniente encontrado es que la librería *ReactJS* (de la que se habla en detalle en el capítulo 9) crea conflictos indeseados y dificulta su utilización, por lo que finalmente se ha descartado.
- **ArcGIS API.** Como se comenta en el subapartado del mapa 2D, esta librería está dotada también de visionado en tres dimensiones. Si bien se destaca que existen limitaciones en cuanto al uso gratuito, su fácil incorporación y completa documentación en el entorno *ReactJS* la han convertido en la opción elegida, por su fácil personalización e implementación de objetos.

Tras la discusión sobre las alternativas planteadas (existen más, pero no fueron tenidas en cuenta por diversas razones) y la elección de la librería a utilizar, en el apartado de descripción de las tareas (capítulo 9) se comenta el proceso seguido para la configuración dentro de la página, así como el desglose de los procesos seguidos para la incorporación a *Satellite Viewer*.

5.3.3. Cálculos

Como es de esperar, las operaciones que se requieren para obtener datos satelitales a partir de la información proporcionada de *Celestrak* no son para nada sencillas, siendo, en numerables ocasiones, fuera del alcance de este proyecto y de los conocimientos matemáticos y físicos obtenidos en la titulación. Por suerte, existen dos librerías en *JavaScript* que se encargan de realizar estos cálculos y proporcionar los resultados convenientes, aunque en ciertas situaciones tengan que tratarse con cautela y aplicar correcciones.

La librería principal y ampliamente reconocida en la comunidad es *Satellite.js* (debidamente documentada en el Anexo I), encargada de la gestión del propagador *SGP4/SDP4* y de los cálculos pertinentes. Dicha librería es prácticamente idéntica a la librería de Brandon Rhode *SGP4* para Python, y no es más que una encapsulación de la misma en el entorno de *JavaScript*. Entre las atribuciones de esta librería se encuentran personalidades como Nikos Sagias (profesor de la Universidad de Peloponeso), David Vallado (autor de '*Fundamentals of Astrodynamics and Applications*', 2000) o T.S. Kelso (creador de *Celestrak*) y cuenta con un total de hasta 16 contribuidores reconocidos, con actualizaciones y ajustes casi semanales [15].

Gracias a esta librería, es posible la obtención casi inmediata de parámetros como la posición, velocidad, altitud, datos de observación y un largo etcétera. Lo que supone una herramienta crucial en la aplicación que se desea desarrollar. Todos estos parámetros, por supuesto, deben ser debidamente introducidos y comprobados, ya que el resultado depende de una cantidad de datos de entrada muy elevada, y deben ser minuciosamente generados.

Esta librería comentada, a su vez, tiene unos tiempos de procesado en ocasiones elevados para obtener resultados que se quieren de forma casi inmediata. Por esta razón, y para obtener los parámetros de forma más clara y rápida, se hace uso de una segunda librería llamada *tle.js*, la cual depende directamente de la primera, pero está desglosada en funciones concretas que facilitan aún más el uso. Además, esta librería está desarrollada enteramente por David Calhoun, uno de los desarrolladores principales de la mencionada inicialmente, lo que le da la fiabilidad y sentido al proyecto [16].

La segunda librería será la utilizada en la mayor parte del proyecto, puesto que los parámetros básicos que se quieren obtener (latitud, longitud, elevación) son fácilmente accesibles con las funciones concretas de este paquete. La ventaja es la exactitud y veracidad de los resultados combinado con un producto más

liviano, sencillo y de menor complejidad. Sin embargo, habrá cálculos que deberán hacerse usando la librería de *Satellite.js*.

Los detalles de esta librería se encuentran en el Anexo II. Los detalles de la obtención de los resultados y la metodología aplicada vienen desarrollados en el capítulo 9 de descripción de tareas, puesto que en este subapartado se pretende mostrar las librerías escogidas y la razón por la que se ha hecho.

5.4. Documentación

Es imposible generar una plataforma completa y veraz sin una base teórica fundamentada y debidamente argumentada, es por ello que se requiere de una elección de documentación existente para poder forjar un apoyo consistente. Dicha investigación tiene como resultado este apartado, donde se procura esclarecer las fuentes utilizadas.

5.4.1. Fundamentos matemáticos y científicos

Si bien para proceder al cálculo de los elementos orbitales se da por hecho la previa adquisición de conocimientos matemáticos y científicos, no está de más hacer una recopilación de conceptos básicos de los que se parte. Todos ellos en orden ideal pero no obligatorio y con la intención de facilitar el entendimiento de la obtención de los resultados. Se determinan necesarios para obtener la base fundamental de la mecánica orbital, no así para interactuar con la aplicación y generar la visualización de satélites.

- Conocimientos avanzados de geometría [16]., siendo especialmente interesantes en tres dimensiones. Así como la capacidad de visualización espacial.
- Familiaridad con cambios de coordenadas [17]. Trato con coordenadas cartesianas, polares y esféricas [18]. Facilidad de comprensión de coordenadas geográficas y coordenadas proyectadas [19].
- Base de métodos de cálculo sofisticados para operaciones complejas e indirectas [20].
- Nociones sobre fundamentos de física [21].
- Conocimientos avanzados sobre movimientos relativos, sistemas inerciales y no inerciales [22].
- Base sobre leyes fundamentales de la física, leyes de Kepler, constantes gravitacionales [23].
- Notación científica, operacional y matemática [24].

5.4.2. Teoría de órbitas y TLEs

Una vez adquiridos los conocimientos marcados como fundamentales para la teoría satelital (mecánica orbital), se dispone a generar la base de la que se parte para la creación de la plataforma.

El orden que la lógica dictaría tras la lectura de este proyecto es la explicación de los infinitamente nombrados *TLEs*, no obstante, carece de sentido comenzar en por ahí, puesto que existe abundante conocimiento que debe explicarse previamente.

- 1. Mecánica Orbital

La astrodinámica o mecánica orbital es la aplicación de la balística y la mecánica celeste a los problemas prácticos relativos al movimiento de cohetes y otras naves espaciales [25]. Está basada en las leyes de Newton y en la ley de la gravitación universal. Esta pretende hacer un estudio exhaustivo sobre las trayectorias de las naves espaciales, además de la multitud de maniobras, y es la herramienta principal de los planificadores de misiones espaciales.

- 2. Coordenadas

No es posible comenzar a hablar sobre posición, velocidad y parámetros principales de la órbita sin antes discutir el sistema de referencia utilizado, las coordenadas en las que se basa y cómo se opera con ellas.

En este proyecto, y, en toda la teoría orbital, se tratará normalmente siempre con dos tipos de coordenadas: el sistema inercial o ECI (de sus siglas en inglés “*Earth-Centered Inertial*”, Inercial Centrado en la Tierra) y el sistema no-inercial o ECEF (de sus siglas en inglés “*Earth-Centered, Earth-Fixed*”, Centrado en la Tierra y Fijado a la Tierra). Ambos comparten la misma coordenada Z, o Eje de Rotación, pero difieren en que el primero de ellos ‘permite’ el giro de la tierra y no gira con ella, siendo la dirección del eje X perpendicular a la coordenada Z y orientada hacia el equinoccio vernal (Primer Punto de Aries, representado como ‘ γ ’). En el caso del sistema ECEF, la dirección del eje X es también perpendicular al eje Z, pero siempre estará fijada con el Meridiano de Greenwich, lo que supone que los ejes ‘giren con ella’, de ahí su nombre de centrado a la tierra y fijado a la tierra. Los ejes Y de ambas representaciones se construyen con un ángulo de 90° con respecto al eje X y sobre el plano ecuatorial, pero no coinciden generalmente por la misma razón que los ejes X no coinciden [26].

Aunque aún no sea trivial plasmarlo, en las representaciones 3D de las órbitas que se harán en la aplicación existirá una diferencia radical en la vista de ambos ejes de referencia, siendo la indicación de estos crucial para la correcta lectura de la información. Por otra parte, en la representación 2D sólo se pretenderá visualizar la traza del satélite, por lo que el mapa estará estático para la vista del usuario, siendo semejante a la representación ECEF.

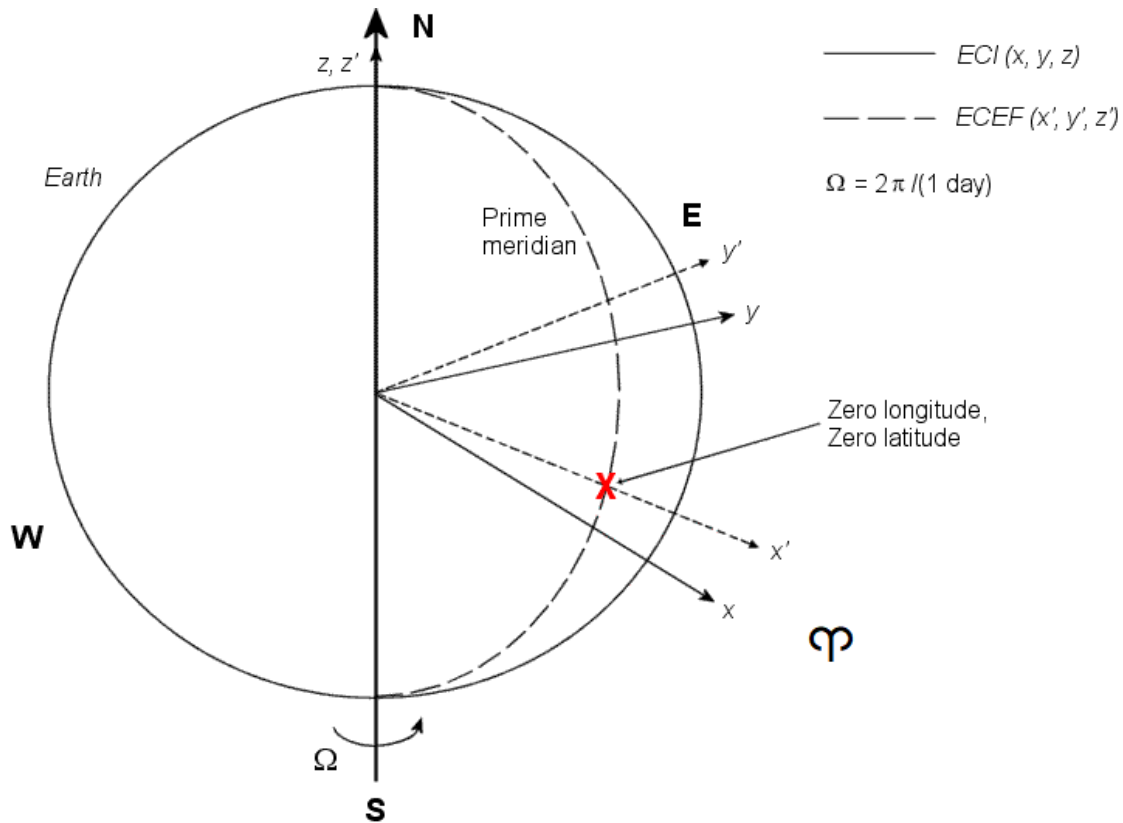


Fig. 5.2 Comparativa entre los sistemas de referencia ECI y ECEF [27].

Mientras que el sistema de referencia ECI puede parecer menos intuitivo, complicado y con menor número de usos, resulta que el propagador SGP4 (siguiente punto de discusión) usa coordenadas en este sistema de referencia para operar y manejar los parámetros satelitales, por lo que tiene un especial interés conocer este tipo de representación.

Cabe aclarar también que, pese a que las representaciones ECEF y ECI consten de los ejes X, Y y Z, son las coordenadas geodésicas longitud (φ) y latitud (λ) (y, en casos concretos, radio (ρ) o altura (h)) las que se usan finalmente en la determinación espacial de los puntos (no así para los cálculos, únicamente como producto final para la comprensión), pero este cambio de variables es trivial, consiste en un cambio de variables cartesianas-esféricas (o cartesianas-polares si no se habla de la traza y se tiene en cuenta la altitud). En las ecuaciones 5.1, 5.2 y 5.3 se muestran los cálculos para la conversión ECEF-cartesianas, partiendo ambas del centro de la tierra como punto de referencia [28].

$$x = R \cdot \cos \lambda \cdot \cos \varphi \quad (5.1)$$

$$y = R \cdot \cos \lambda \cdot \sin \varphi \quad (5.2)$$

$$z = R \cdot \sin \lambda \quad (5.3)$$

- 3. Propagador SGP4

Una vez definidos brevemente los sistemas de referencia usados en el campo orbital, se ha de definir el funcionamiento del cálculo de las coordenadas que se lleva a cabo en la realidad. Lo cierto es que los resultados son fruto de un estudio mucho más complejo de lo presentado anteriormente, todo ello causado por modelos simplificados de perturbaciones, o bien simplemente propagadores.

La familia de modelos se limita a cinco de ellos: SGP, SGP4, SDP4, SGP8 y SDP8. Todos y cada uno de ellos son utilizados para calcular vectores de estados de órbitas de satélites y basura espacial relativa a la tierra, y haciendo uso del sistema de referencia ECI (explicado en el punto anterior) [29]. Aunque existan todos los modelos mencionados, se suele hacer referencia a ellos simplemente con 'SGP4', puesto que es el más utilizado por la especial utilidad para los conocidos TLE (punto 5 de este subapartado).

Estos propagadores predicen el efecto de las perturbaciones causadas por la forma de la tierra (que, erróneamente, es tratada como una esfera perfecta), el rozamiento, la radiación y los efectos gravitacionales causados por otros cuerpos, tales como la luna o el sol [30]. Se estima que el propagador tiene un error de menos de 1km, pero crece diariamente entre 1 y 3 km, la razón principal por la que los datos son continuamente actualizados [31].

Dichos propagadores conllevan una carga matemática y física que se escapa de los límites de este proyecto, por lo que no se entrará en detalle en los cálculos que éste efectúa. Sin embargo, era de vital importancia remarcar algunos datos importantes del mismo, para poder entender a la perfección algunas singularidades de los datos orbitales. Entre ellas, la razón por la que los TLEs son actualizados diariamente (como se ha mencionado, el propagador conlleva errores que crecen con el tiempo, y se han de corregir). Para más información, la página de *Celestrak*, referenciada en multitud de ocasiones, tiene documentación extensa sobre propagadores, códigos originales de la compañía *NORAD* (desarrolladora del producto [32]) y librerías actualizadas sobre el propagador [33], entre las que se encuentra la que se usará para este proyecto, mencionada en el subapartado 5.3.3. de Cálculos.

- 4. Parámetros de la órbita

Hasta este punto, se ha mencionado toda la teoría que recoge la información necesaria para la visualización de objetos, las herramientas utilizadas y toda clase de documentación, por lo que es conveniente y necesario llevar al quid de la cuestión: cómo representar un satélite y cuáles son los parámetros que lo conforman.

Los elementos orbitales son aquellos parámetros necesarios y suficientes para determinar una órbita (éste usa un modelo de dos masas siguiendo las leyes de movimiento de Newton). Con esta denominación se suele hacer referencia a seis parámetros básicos, también denominados como elementos keplerianos (como

tributo a Kepler [34]), que a continuación se definen para el caso de órbitas geocéntricas [35]:

- a. *Longitud del nodo ascendente* (\varnothing o Ω). Ángulo formado entre el primer punto de Aries (γ) o eje X, también conocido como origen de la longitud y la dirección del nodo. Suele hacerse referencia a este parámetro (en casos geocéntricos como este) como Ascensión recta del nodo ascendente. Este ángulo es medido siempre hacia el este (tal y como es visto desde el norte, en sentido antihorario). Color salmón en la figura 5.3.

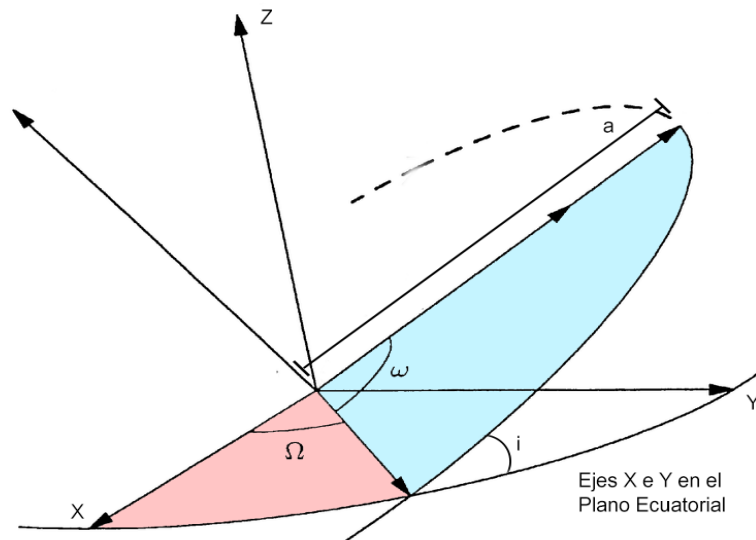


Fig. 5.3 Elementos keplerianos.

- b. *Argumento del perihelio* (ω). Su función es orientar la elipse sobre el plano orbital, puesto que mide el ángulo desde el nodo ascendente hasta el perigeo, medido en la dirección del movimiento del cuerpo que orbita. Nótese que, de no tener en cuenta la dirección del movimiento del cuerpo, se podría producir una ambigüedad. Color azul en la figura 5.3.
- c. *Inclinación orbital* (i). Ángulo formado por el plano orbital y el plano Ecuatorial.
- d. *Semieje mayor* (a). Medida del semieje mayor de la elipse que realiza el objeto orbital. En ocasiones se determina también el semieje menor, pero es el mayor el que se determina siempre y quien forma parte de los elementos keplerianos. Su cálculo es sencillo: se trata del valor medio de las distancias mínima y máxima de la elipse al foco, tal y como aparece en la ecuación 5.4.

$$a = \frac{r_{\text{máx}} + r_{\text{mín}}}{2} \quad (5.4)$$

- e. *Excentricidad* (e o ϵ). Parámetro que determina la desviación de la elipse con respecto a una circunferencia (cuanto más cercano a 0, menor excentricidad y trayectoria más parecida a la circunferencia). Para el caso de las elipses, dicho parámetro debe valer igual o más que 0 y siempre menos que 1 (una excentricidad de valor 1 se corresponde con una parábola). Tomando el semieje mayor y el semieje menor, respectivamente, como 'a' y 'b', su cálculo queda expuesto en la ecuación 5.5.

$$e = \sqrt{1 - \frac{b^2}{a^2}} \quad (5.5)$$

- f. *Anomalía media de la época* (M_0). Parte del período orbital que ya ha transcurrido, expresado como un ángulo. Ecuación 5.6.

$$M = n \cdot (t - t_0) \quad (5.6)$$

Donde 'n' es el movimiento medio (normalmente medido en °/día, o rad/día), 't' es el instante en el que se desea obtener el valor, y 't₀' es el instante de paso del planeta por el perihelio (o perigeo, anteriormente mencionado).

Además de estos valores, en ocasiones se utilizan otros datos como la *anomalía verdadera* (v), *semieje menor* (b), *excentricidad lineal* (ε), *anomalía excéntrica* (E), *longitud media* (L), *longitud verdadera* (l) y *período orbital* (T). Sin embargo, no se pondrán en uso en este proyecto, ya que no forman parte de los elementos que conforman los TLEs.

- 5. TLE

El formato de datos NORAD Two-Line-Element set o simplemente TLE, supone la codificación de una lista de elementos orbitales de un objeto orbital terrestre (y únicamente terrestre) para un instante en el tiempo, también conocido como la época (llamada así en astronomía a una fecha en concreto, en ocasiones medida en milisegundos a partir del 1 de enero del 2000 a las 00:00 según UTC [36]). Gracias a este dato, es posible la correcta representación de un satélite dado en el instante deseado. A partir de ese instante, el error puede crecer progresivamente con el tiempo, por lo que dichos TLEs son actualizados constantemente, y no son válidos para todos los tiempos posteriores si se requiere de rigurosidad y exactitud en los cálculos.

Este formato está íntimamente relacionado con los propagadores anteriormente mencionados, puesto que nacen tras la creación de los mismos, y se basan en todas sus perturbaciones para la obtención de los resultados [37].

Inicialmente se pretendió materializar mediante tarjetas perforadas [38], un recurso altamente utilizado en los primeros tiempos de la programación y la informática. Por esta razón, el formato nació como dos tarjetas de 80 columnas (lo que contendría 80 caracteres en total), pero finalmente se decidió reemplazar dicho formato por archivos de texto con dos líneas de 69 columnas en formato ASCII, precedidas de una línea destinada al título.

Un ejemplo de TLE sería el representado en la figura 5.4.

```
ISS (ZARYA)
1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11606-4 0   2927
2 25544   51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537
```

Fig. 5.4 Ejemplo de TLE de la Estación Espacial Internacional [38].

Pese a que las líneas 1 y 2 del formato tienen una extensión de 69 caracteres, el título debe estar siempre comprendido entre los 24 primeros dígitos, entre los que se admiten algunos caracteres especiales como paréntesis (como en el ejemplo de la figura 5.4).

Para la primera línea, como se ha comentado, existirá una extensión de 69 caracteres, dispuesta en el formato de la tabla 5.1.

Tabla 5.1 Formato de la línea 1 del TLE ejemplo.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	→ Carácter			
1		2	5	5	4	4	U		9	8	0	6	7	A			0	8	2	6	4	.	5	1	7	8	2	5	2	8		→ TLE				
1		2				3			4	5		6				7	8										→ Elemento									
34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	
-	.	0	0	0	0	2	1	8	2			0	0	0	0	0	-	0		-	1	1	6	0	6	-	4		0			2	9	2	7	
9									10									11									12					13			14	

Donde los respectivos valores se corresponden con los contenidos expuestos en la tabla 5.2.

Tabla 5.2 Elementos de la línea 1 del TLE ejemplo.

[01]	1-Número de línea	1	[21]-[32]	8-Día del año + fracción del día	264.51782528
[03]-[07]	2-Número de catálogo (NORAD ID)	25544	[34]-[43]	9-Primera derivada de M0*	-.00002182
[08]	3-Clasificación ('U', 'C', 'S')*	U	[45]-[52]	10-Segunda derivada de M0*	0000-0
[10]-[11]	4-Últimos dos dígitos año de lanzamiento	98	[54]-[61]	11-BSTAR término de rozamiento*	-11606-4
[12]-[14]	5-Número del año de lanzamiento	067	[63]	12-Tipo de efemérides* (decimal)	0
[15]-[17]	6-Parte del lanzamiento	A	[66]-[68]	13-Número de elemento	292
[19]-[20]	7-Últimos dos dígitos del año (época)	08	[69]	14-Checksum*	7

* BSTAR d. t.: representa una modificación del coeficiente de balística (coeficiente que relaciona rozamiento, masa y área) representado en unidades de 1/RadioTerrestre [39]; Clasificación: 'U' - 'Unclassified', 'C' - 'Classified', 'S' - 'Secret'; Tipo de Efemérides: generalmente '0', en ocasiones '2'. Ver ANEXO III; Checksum: resultado de sumar los valores contenidos en la línea, se trata de una comprobación ('Check-sum': comprobación de la suma). Ver ANEXO IV; M0: Mean Motion, o Anomalía Media, mencionada en los elementos orbitales básicos;

Por otro lado, la línea 2 del formato es generada de forma parecida, recogida en las tablas 5.3 y 5.4.

Tabla 5.3 Formato de la línea 2 del TLE ejemplo.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	→	Carácter	
2		2	5	5	4	4			5	1	.	6	4	1	6		2	4	7	.	4	6	2	7		0	0	0	6	7	0	3		→	TLE	
1		2							3								4								5										→	Elemento

35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
1	3	0	.	5	3	6	0		3	2	5	.	0	2	8	8		1	5	.	7	2	1	2	5	3	9	1	5	6	3	5	3	7
6									7									8								9				10				

Tabla 5.4 Elementos de la línea 2 del TLE ejemplo.

[01]	1-Número de línea	1	[35]-[42]	6-Argumento perigeo (grados)	130.5360
[03]-[07]	2-Número de catálogo (NORAD ID)	25544	[44]-[51]	7-Anomalia media (grados)	325.0288
[09]-[16]	3-Inclinación (grados)	51.6416	[53]-[63]	8-Movimiento medio (grados/día)	15.72125391
[18]-[25]	4-Asc. recta nodo ascendente (grados)	247.4627	[64]-[68]	9-Número revolución en la época	56353
[27]-[33]	5-Excentricidad (decimal*)	0006703	[69]	10-Checksum	7

* Tanto para la Línea 1 como la 2, los valores asumidos como decimales están expresados de tal forma que los dos primeros símbolos representan: 0,(dígitos-2)*10^-(últimos dos dígitos);

De esta forma, queda totalmente definido el formato de datos conocido como TLE, y es más sencillo de entender la razón por la que toda la documentación teórica previa era necesaria, puesto que es la base principal de conocimiento para poder ser comprendido y correctamente operado.

5.4.3. Satélites disponibles

Las formas de acceder a estos TLEs son oficialmente dos: *Celestrak* y *Space-Track*. Ambas corporaciones están dotadas de la información pública (existe información clasificada y elementos militares de acceso restringido) sobre los satélites que orbitan la tierra. La fuente de información es la Fuerza Espacial de los Estados Unidos, la cual lleva supervisando, controlando y almacenando datos de los objetos orbitales desde el principio de los tiempos de la carrera espacial. Si bien el canal de comunicación oficial de este organismo es *Space-Track*, ambas alternativas son igualmente válidas y veraces, tal y como es explicado en el subapartado 1.3.2, donde se comenta la peculiaridad de *Celestrak*.

Para este proyecto se usarán indistintamente ambas fuentes, y, en ocasiones, puede hacerse alusión sólo a alguna de ellas, pero en cualquier caso ambas son igualmente válidas y correctas, limitándose meramente a una decisión del desarrollador y de los requisitos de la plataforma.

Como información complementaria, existen reportes sobre análisis de datos, estadísticas interesantes y diferentes análisis hechos en ambas fuentes, por lo que partir de ambas puede ser en cualquier caso beneficioso y no

contraproducente. Además, en ambas se puede elegir previamente el formato en el que se desea recibir los datos, ya sean directamente TLEs en archivo de texto o bien en formato JSON para ser fácilmente adaptables a un entorno de programación.

5.5. Valoración final, gestión de alternativas y plan de acción

Discutidas las fuentes, la documentación, las herramientas que se pretenden utilizar y toda la teoría aplicada, es momento de generar la página web, el objetivo principal de esta memoria.

Aunque pueda haber fuentes de fallos, estos se han de prever con debida antelación y se ha de preparar la plataforma para estos casos. Esta es una de las razones principales por las que en todos los apartados se han discutido diferentes alternativas, para poder tener un entorno de recuperación conocido en el caso de que existiera dicha falla: en el caso de las librerías de cálculo, se han tomado dos opciones simultáneas, en el caso de la obtención de TLEs, se han obtenido dos fuentes alternativas y, por último, en el caso de los mapas, se han discutido diferentes proveedores válidos para realizar la visualización de los satélites. No se espera ningún fallo, pero siempre es conveniente haber planteado otros caminos, por si en algún momento fuera necesaria una migración a otra alternativa.

Por como ha quedado definido en este apartado de utilidades, se dispone de todos los recursos necesarios para crear la página, y se puede empezar con la descripción de las tareas, no sin antes exponer el pliego de condiciones, recogido en el capítulo 6.

CAPÍTULO 6. PLIEGO DE CONDICIONES

6.1. Conocimientos adquiridos en la carrera como apoyo para el proyecto

La asignatura que, sin duda, ha impulsado en mayor parte este proyecto, es Mecánica Orbital. Se trata de una de las dos partes (correspondiente a 3 cts.) en las que se divide el bloque obligatorio de Fundamentos de la Navegación, cursado en el tercer curso del grado de Ingeniería Aeroespacial en la Universidad de Sevilla.

En esta asignatura se hace un repaso general a la historia de la exploración espacial, se imparten las bases (coordenadas, parámetros, cálculos, transformaciones) que constituyen la mecánica orbital y se imparten de forma general conocimiento del panorama aeroespacial (transmisiones, tipos de órbitas, usos de satélites, *transferencias de Hohmann*) [40]. Gracias a estos conceptos, que forman los cimientos del campo aeroespacial, es posible desarrollar un proyecto de este calibre.

Sin embargo, esta asignatura supone la cúspide de la pirámide de conocimiento adquirido para situarse en la creación de trabajo. Son necesarias las asignaturas [41] de Matemáticas I (Álgebra), Matemáticas II (Cálculo), Matemáticas III (Geometría y Métodos Aplicados), Ampliación de Matemáticas (Matemática Diferencial y Compleja) y Métodos Matemáticos (herramientas de cálculo sistemáticas basadas en ordenador, para situaciones complejas) para obtener el conocimiento matemático necesario para comprender toda la lógica que alberga este proyecto.

Por otra parte, no habría sido posible llegar a este punto sin antes haber cursado Física I (Cinemática, Dinámica, Leyes, Campos), Física II (Fundamentos Eléctricos, Termodinámicos y Electromagnéticos) y Ampliación de Física (Mecánica Clásica, Sistemas Inerciales). Con estos tres bloques [41], se abarcan los conocimientos necesarios para asumir la lógica física que en esta memoria se recoge.

Por último, cabe destacar asignaturas que, de alguna forma, ya sea transversal o directamente, han contribuido aportando conocimientos al desarrollo de esta plataforma. En la asignatura de Introducción a la Ingeniería Aeroespacial se dedicó gran parte de la misma a exponer la historia de la exploración espacial. En la asignatura de Sistemas de Propulsión se comentó la forma en la que actúan los cuerpos orbitales para la transferencia de órbitas y su comportamiento general en el espacio. Y, sin duda, cabe destacar los conocimientos informáticos adquiridos en la asignatura de Proyectos de Gestión del Tráfico Aéreo de la Universidad Politécnica de Cataluña, que, si bien la carga teórica aprendida en esta no está en absoluto relacionada con la aquí expuesta [42], fueron las competencias adquiridas en esta las que han impulsado de forma exponencial la decantación por este proyecto. Se realizó todo un software de visualizado de aviones en un mapa, con una consola y una serie de tablas de datos, que dieron pie a la idea precursora de *Satellite Viewer*.

6.2. Justificación de la elección del proyecto como adecuado para la titulación del grado de Ingeniería Aeroespacial, especialidad en Sistemas Aeroespaciales

En la búsqueda de un Trabajo de Fin de Grado ajustado para la titulación que se trata, se barajaron todo tipo de opciones: gestión de tráfico, investigación sobre técnicas utilizadas y técnicas mejoradas para aeropuertos, creación de dispositivos electrónicos con aplicación aeronáutica o incluso un estudio histórico sobre el panorama aeroespacial. Finalmente, se decidió tomar un camino totalmente aeroespacial, debido a que el interesado (desarrollador del producto) desea formarse de cara a un futuro profesional en este campo y consistía en un campo más que justificado para la titulación.

Con la aparición de aplicaciones del estilo de *FlightRadar24*, surge la idea de realizar algo similar para la visualización de satélites, donde no existe tanta variedad y hay todo un camino de oportunidades y desarrollo. Realizando esto, además, se abarcan varios de los caminos fundamentales del grado: sistemas aeroespaciales, conocimientos de programación, cálculos matemáticos complejos aplicados a la ingeniería e investigación sobre el panorama del mundo satelital, por lo que supone una idea debidamente justificada y con una base científica ampliamente apoyada por los conocimientos adquiridos en la titulación del grado (subapartado 6.1).

6.3. Competencias del proyecto

Las competencias desarrollables en la creación de esta plataforma, se recogen en la siguiente lista:

- Conocimiento general aeroespacial: teoría de órbitas, tipos de satélites, tipos de transferencia, familias existentes, utilidades de cada una en la actualidad, atribución por países, restricciones del espacio, cantidad de basura aeroespacial, propósitos futuros, situación gubernamental entorno a este campo, bases de cobertura, vida útil de un satélite, familiaridad con los organismos encargados de la gestión del espacio, entre otras cuestiones.
- Familiaridad con cálculo complejo matricial y vectorial. Repaso y ampliación de transformaciones en diferentes tipos de coordenadas.
- Estudio y análisis de las bases de la mecánica orbital y partes de la mecánica clásica, en la que se basa.
- Amplia experiencia de programación software, basada en entorno web, familiaridad con multitud de lenguajes, tales como *JavaScript*, *HTML*, *CSS* o *Matlab*. Así como extensiones *ReactJS* y 'frameworks' como *NextJS*.

- Aprendizaje de manipulación de datos en gran cantidad de formatos: conversión texto – JSON, JSON – texto, descomposición, gestión de variables, vectores y matrices digitales.
- Aprendizaje de uso de programas de diseño, requeridos para el diseño de la página web, tales como Photoshop (edición de imagen en general), *Illustrator* (creación de iconos, mapas vectoriales en general), *Google Fonts* (gestión de fuentes de la página) o Adobe Color (para el ajuste correcto e idóneo del color de la aplicación).

Por otra parte, existen una serie de competencias personales que tienen fuerte influencia en este trabajo, más allá del conocimiento y la experiencia intelectual:

- Gestión de tiempo, organización y planificación
- Capacidad de ceñirse al plan establecido, habilidad de lidiar con contratiempos, problemas y mejoras y predisposición para recibir otros puntos de vista, así como capacidad de autocrítica.
- Compromiso de finalizar un proyecto en el que el tiempo, las herramientas y los recursos eran limitados.

CAPÍTULO 7. IMPLEMENTACIÓN

7.1. Lenguaje de programación

Es importante tener claras las funciones que se han de abarcar a la hora de empezar un proyecto, puesto que es uno de los factores que pueden afectar más a la elección de un lenguaje motor u otro. En esta memoria se está describiendo una plataforma para entorno web, por lo que se necesita un lenguaje que esté adaptado para esto. En este campo, el lenguaje que gobierna es, sin duda, JavaScript y sus inconmensurables extensiones, *frameworks* y funcionalidades. Por todo esto, el lenguaje base elegido para el desarrollo de Satellite Viewer es JavaScript, pero no será el único. En este apartado viene descrito todo lo necesario para el despliegue de la plataforma.

7.1.1. HTML

Lenguaje básico para la creación de páginas web, con el que nacieron la mayoría de páginas web que se conocen hoy día. Se trata de la programación estándar para el desarrollo de herramientas de navegador, por lo que es de especial utilidad. Con el desarrollo de lenguajes como *Cascading Style Sheets* o *JavaScript*, su escritura puede ser ‘asistida’, de modo que, con la escritura de código debidamente en estos, es posible generar el mismo producto que se obtendría editando en él [40]. Precisamente por esta razón, *HTML* no será prácticamente programado en esta plataforma, ya que los elementos serán contruidos en JavaScript, pero es necesario mencionarlo por la importancia que tiene y la gran influencia en la historia de la creación de páginas web (JavaScript es sólo el lenguaje, y el desarrollo web no deja de ser una adaptación más de las muchas que tiene, por lo que se puede afirmar que ‘simula’ y ‘sustituye’ al lenguaje *HTML*).

7.1.2. CSS

Lenguaje que compagina a *HTML* en la creación de páginas web. Mientras que el primero se encarga de los contenidos y todo el material que va a ser representado, la manera de exponerlos y referenciarlos, *CSS* tiene como propósito la inserción del estilo de los mismos.

Este lenguaje es el responsable de los estilos de la página, comenzando desde las fuentes utilizadas o su tamaño, hasta tareas más complejas como la colocación con la interacción de la página, los espacios asignados a los elementos, el modo de interacción con las barras de deslizamiento, el comportamiento con los cambios de tamaño de la ventana y un largo etcétera.

En el cuadro de texto de la figura 7.1 puede apreciarse un código de ejemplo para el ajuste de los estilos de un ‘*container*’ (o contenedor en español, donde se suele albergar el contenido de la página, valga la redundancia).

```
.container {  
  padding: 0;  
  height: '100vh';  
  width: '100%';  
  margin: 0px;  
  font-family: 'Alumni Sans Pinstripe', sans-serif;  
  font-size: '25px';  
}
```

Fig. 7.1 Código ejemplo de generación de estilos para ‘*container*’.

Analizando el código, puede ser intuitivo descifrar lo que está produciendo: está aportando márgenes, tamaño, fuente y tamaño de fuente al contenedor.

Si bien *HTML* puede ser obviado con la adaptación de *JavaScript*, *CSS* debe ser tratado y manipulado con obligatoriedad para el aporte de los estilos de la aplicación, por lo que era de estricto cumplimiento su presencia en este apartado.

7.1.3. *JavaScript*

Son varias las razones por las que *JavaScript* es el lenguaje más usado con diferencia, todo ello sin tener en cuenta que la tendencia es la aplicación web, con reducciones anuales de las aplicaciones de escritorio y software variado [39].

Entre otras, las más destacables podrían reducirse a las siguientes:

- Está dotado de una base y arquitectura comprensible, fácil de entender y que puede adaptarse desde el programador principiante hasta los niveles más altos de la profesión: se trata de un software muy flexible con una alta capacidad de adaptación.
- Se puede desarrollar software para todo tipo de dispositivos, desde móvil, Tablet u ordenador hasta videojuegos multiplataforma.
- El hecho de que exista una gran comunidad que hace uso de este lenguaje de forma habitual, hace que pueda encontrarse toda clase de documentación, tutoriales o ejemplos ya hechos por otros desarrolladores, lo que fomenta aún más su continuo crecimiento.
- Quizás el punto más fuerte en cuanto a la programación web: está dotado de *frameworks* potentes con estilos modernos, en continua evolución y

con toda clase de herramientas y efectos visuales para la interacción usuario-página.

Todas estas razones hacen que el gráfico mostrado en la figura 7.2 sea más que entendible, así como la elección de este entorno como el principal para el proyecto.

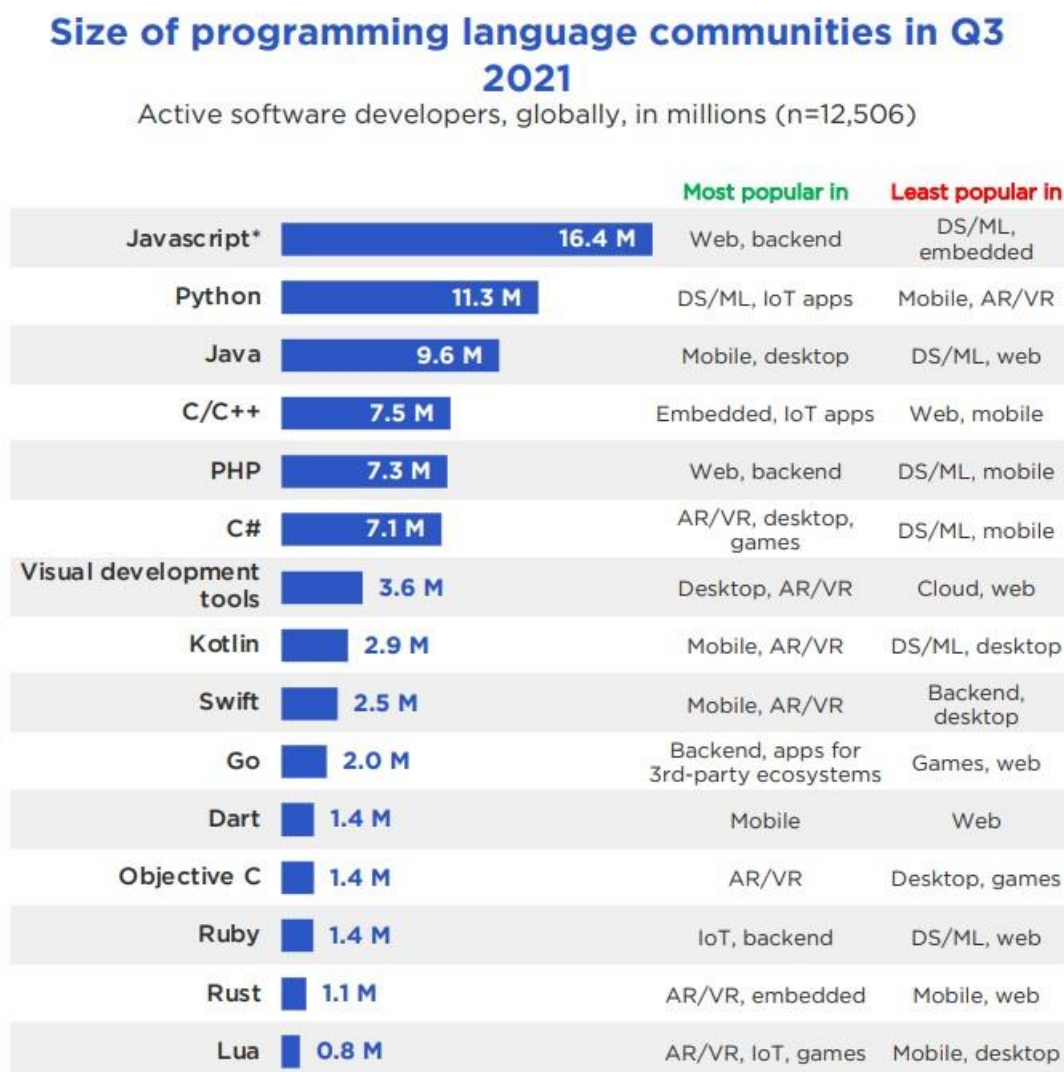


Fig. 7.2 Estadística de uso de diferentes lenguajes de programación en 2021, recogida en la 21ª Edición de *State of the Developer Nation* [38].

Como se ha mencionado, este lenguaje no actuará solo en la producción de la aplicación, sino que se hará uso de la extensión *ReactJS* para la creación de los componentes de la página y del *framework NextJS*.

7.1.4. JSON

A diferencia de los subapartados anteriores, *JSON* se trata de un formato de almacenamiento de datos de lectura directa para *JavaScript* (*JavaScript Object Notation*), y no de un lenguaje de programación. Es debido al alto uso de este formato (lectura de TLEs, aporte de datos complementarios, formato base de análisis de *Celestrak* y *Space-Track*) por lo que se le hace una mención especial en este apartado. En la figura 9.3 puede verse la variable Posiciones [] como ejemplo de este tipo de formato.

7.1.5. MatLab

Lenguaje de programación meramente numérico, fácilmente generable (lenguaje de alto nivel) con la que se pueden gestionar operaciones con altas velocidades de procesamiento. Además de disponer de infinidad de extensiones, destaca la producción de gráficas analíticas de información de gran utilidad y procesamiento de datos de forma visual, de modo que es recomendable para generar todo tipo de análisis visual.

Con esta plataforma se generarán algunas gráficas relevantes del proyecto, puesto que, a pesar de ser una gran potente herramienta de cálculo, es más fácil tener los cálculos integrados en *JavaScript*. Por tanto, su uso será limitado y únicamente con fines de visualización de datos estadísticos, pero igualmente debe ser aquí mencionado, puesto que será de gran ayuda.

7.2. Front-end

En todo el primer apartado de este capítulo se pretendía hacer un repaso de los lenguajes utilizados para crear la plataforma, sin atender a usos más concretos de cada uno. En este apartado se precisará información más detallada sobre el desarrollo del entorno visual (a lo que accede el usuario), también llamado *front-end*. Para ello, se mencionan las herramientas complementarias a *JavaScript* utilizadas para la producción de *Satellite Viewer*, recogidas en los subapartados 7.2.1 y 7.2.2.

7.2.1. ReactJS

Tal y como lo define la propia compañía, se trata de ‘una librería de *JavaScript* para construir interfaces para el usuario’ [42]. Por la descripción, puede entenderse la razón por la que se ha escogido esta extensión, ya que, como se menciona en el subapartado 7.1.3, *JavaScript* es únicamente el lenguaje, y puede adaptarse según el propósito para cualquier situación. En este caso, que se trata del desarrollo web, es vital contar con una librería de interfaces usuario,

puesto que facilita mucho su programación y tiene resultados muy positivos para aquellos que acaben haciendo uso de la plataforma.

La librería escogida no es la única, de hecho, existen multitud de ellas generalmente estandarizadas en la comunidad de este lenguaje (figura 7.3), y no es ésta la más usada (sí la segunda, según esta fuente).

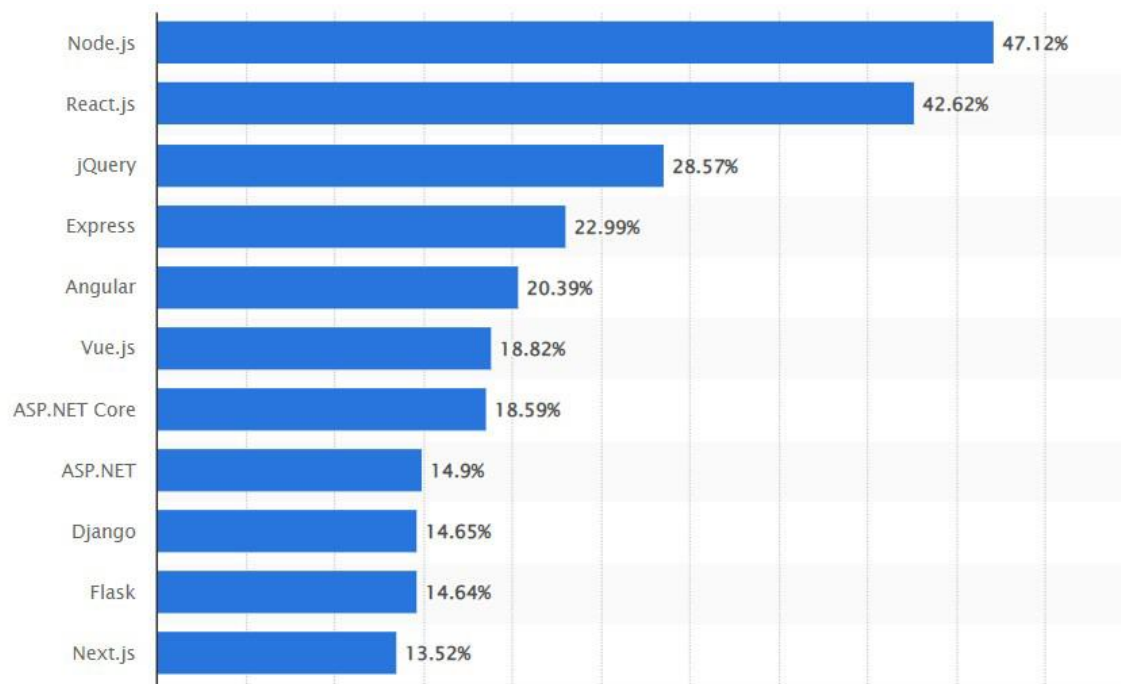


Fig. 7.3 Estadística de uso de diferentes librerías de *JavaScript* a fecha de 2022, según *statista.com* [38].

Las razones por las que se hace uso de *ReactJS* son varias, brevemente resumidas a continuación:

- Al igual que *JavaScript*, puede usarse para toda clase de plataformas.
- Flexible, versátil, con una infinidad de compatibilidades y un servidor con continua evolución, crecimiento e innovación
- A diferencia de otras librerías, evita todo lo posible la intrusión de otros lenguajes como *HTML*, que, en ocasiones, pueden ser bastante tediosos y menos visuales para programar en la actualidad.
- Su funcionalidad más destacada y, probablemente, la razón por la que se ha escogido en este proyecto: se basa en componentes y hace que su estructuración, programación e implementación sea muy sencilla, de gran ayuda para el programador y de fácil acceso para el usuario.
- Los conocidos *React hooks*, que hacen de *JavaScript* un entorno basado en variables estado, fácilmente accesibles y actualizables, ofrecen al

entorno página-usuario un sinfín de posibilidades de actualización de datos, reasignación de valores, efectos con interacciones y un largo etcétera que, en definitiva, hacen de la página un lugar más orgánico y cambiante.

- Aunque no siempre valga de referencia, sólo basta mencionar algunos proyectos en los que esta librería se ha visto envuelta para ratificar su potencial: *Instagram*, *Facebook*, *Netflix*, *AirBnb* o *Whatsapp*, entre otros [43].

7.2.2. Framework *NextJS*

El uso de la librería *ReactJS* como principal para la creación de la interfaz no supone que ésta sea la única utilizada: existen librerías complementarias que pueden generar resultados incluso mejores.

En el caso de esta aplicación, se ha optado también por el *framework NextJS*, el cual suele tener una finalidad similar a *ReactJS*: '*muchos desarrolladores tienen problemas tratando de encontrar diferencias entre ambas*' según *readixweb.com* [44]. Si bien *ReactJS* es simplemente una librería más de *JavaScript*, *NextJS* es un entorno de desarrollo construido 'por encima' de *ReactJS*.

Este *framework* tiene la característica principal que es capaz de crear, estructurar y construir todo el entorno web de usuario mediante un paquete usuario-desarrollador fácil de usar y extremadamente intuitivo de desarrollar, lo que facilita enormemente del desarrollo de la página, especialmente en los inicios de la misma, cuando se debe comenzar por los 'cimientos'. Además, permite crear renderizados (carga visual) desde el lado del servidor, y no desde el entorno del usuario, lo que hace que la experiencia sea fluida y dinámica.

Si bien se trata de una herramienta de gran utilidad que supondrá un gran paso en las etapas iniciales de este proyecto, no siempre se pondrá en uso más allá de esta función, puesto que *React* es una librería suficiente y completa. Los campos en los que suele destacar su uso son:

- Páginas web comerciales: su arquitectura es muy sencilla de construir desde este *framework*, incluso para funcionalidades como administrador, usuarios, contraseñas y un largo etcétera.
- Páginas web dedicadas al *marketing*, ya sea compañías que quieran exhibir su expansión o bien quieran hacer una presentación de cara al público de forma muy visual.
- Generar el despliegue de una página: para el que se usará en este caso.

7.3. Navegador

Toda la programación creada en la fase de desarrollo debe ser debidamente visualizada, comprobada y examinada. Para ello, además de una terminal (*Visual Studio Code*) se necesitará un compilador. En este caso, al tratarse de una página web, se hará lo propio albergando la plataforma en puertos locales de la conexión y de este modo sea accesible desde el navegador deseado.

Para el desarrollo de *Satellite Viewer* se usará en todo momento el navegador más usado en el entorno web de forma mundial: *Google Chrome*. Si bien no debería ocurrir ningún problema, cabe resaltar que, al hacer las comprobaciones oportunas siempre en este navegador, no se descarta un diseño igualmente óptimo para otro tipo de navegadores, aunque este fallo no tiene porqué ocurrir.

Por último, aunque la terminal del entorno de programación sea crucial para ver el estado del código y visualizar cualquier tipo de fallo, también será realmente útil la consola del navegador, desde donde se puede acceder a los avisos, fallos e incluso al propio código de forma directa, lo cual hace que la detección y corrección de errores sea realmente rápida e inmediata.

CAPÍTULO 8. ESTRUCTURA DE LA WEB APP

8.1. Arquitectura

Haciendo un breve resumen de los aspectos de la plataforma desglosados hasta este capítulo, se ha pasado por el contexto, la razón de su creación y las alternativas existentes (Capítulo 1), los objetivos y limitaciones que se deben asumir desde el comienzo (Capítulo 2), la metodología (Capítulo 3), la planificación (Capítulo 4), las bases en las que se apoya el proyecto (Capítulo 5), los conocimientos en la carrera que dan sentido al proyecto (Capítulo 6) y la implementación del mismo (Capítulo 7). Es, una vez todos estos capítulos han sido completamente expuestos, cuando se puede comenzar a hablar más concretamente de la plataforma, atendiendo puntos más concretos sobre la programación, componentes o tareas operacionales.

El primer paso para describir una página es especificar la estructura que ésta sigue. En el caso de *Satellite Viewer*, debido a que se pretende crear una web liviana, rápida y accesible, no estará dotada de una gran cantidad de pestañas diferentes, sino que se pretende tomar una estructura sencilla del tipo horizontal: aquella en la que las dependencias (en este caso, las diferentes pestañas) se desarrollan de forma horizontal con respecto a la página principal (véase figura 8.1).

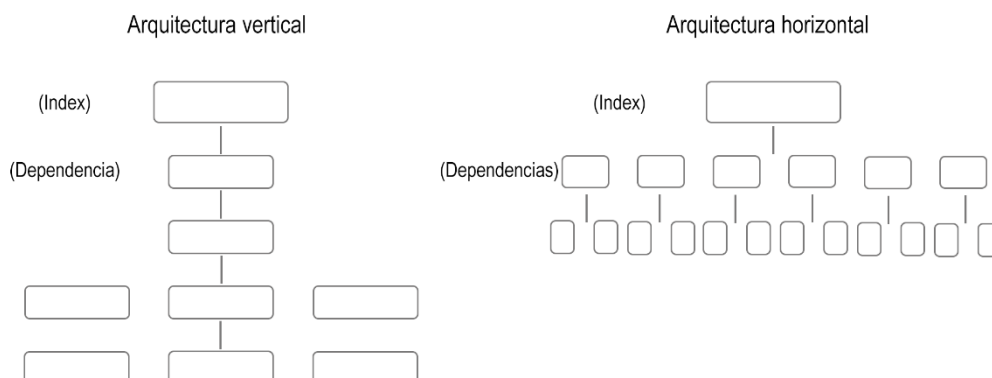


Fig. 8.1 Esquema comparativo entre arquitectura vertical y horizontal.

Las ventajas de crear la página con este tipo de arquitectura es la accesibilidad que se le proporciona al usuario a la hora de encontrar el contenido que está buscando: en tan sólo un par de '*clicks*' se encontrará frente a aquello que desea visualizar. Las páginas con arquitectura vertical o secuencial, por otro lado, necesitan varias interacciones para llegar al contenido, y, en general, suelen suponer un rechazo al consumidor, llegando a conseguir incluso el abandono de la misma por parte del interesado [46].

Además de esta arquitectura, la plataforma (por razones obvias) tendrá una organización de un solo nivel, sin jerarquías: todas las dependencias tendrán el

mismo peso y no hará falta pasar por alguna de ellas para llegar a otra. Al entrar en el menú principal se dispondrá de los accesos para todas las pestañas disponibles y, a su vez, serán disponibles todas las pestañas desde cualquiera de ellas, sea cual sea en la que se encuentre el usuario.

Este modelo de organización por niveles tiene una desventaja remarcable: no será posible llegar a un contenido en concreto a través de una URL (*Uniform Resource Locator*, coloquialmente conocida como enlace o dirección de página web), sino que será necesario interactuar con la pestaña para visualizarlo. Esto puede suponer que, para dos URLs iguales, se pueda estar visualizando contenidos totalmente distintos a la vez. Este modelo de página es conocido como SPA, de sus siglas en inglés *Single Page Application*, o, aplicación de una sola página en español. Lo cual está definido en el apartado 8.2.

8.2. Capa de presentación

La capa de presentación (*front-end*) es aquella que crea el punto de conexión con los usuarios a través de una interfaz gráfica. Se trata del punto de salida del sistema ya que representa de forma visual los datos, componentes y todo el código asociado (apartado 7.2) que contiene. Además, supone el punto de entrada, puesto que constantemente procesa las operaciones de los usuarios y trata de mostrar los resultados en el dominio del sistema. El esquema general de procesos de una página web puede verse en la figura 8.2.

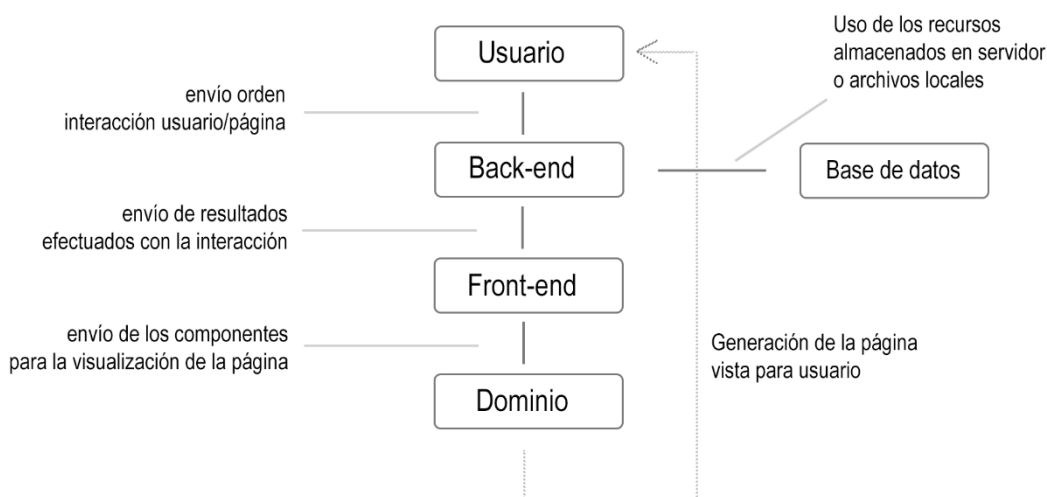


Fig. 8.2 Esquema de procesos en una página web convencional.

Como puede verse, se han de cuidar varios procesos para que el funcionamiento de la plataforma sea el óptimo, habiendo bastantes puntos donde el parámetro de rapidez de procesamiento pueda verse comprometido. Para garantizar que esto no ocurra, es necesario generar operaciones con tiempos de procesamiento rápidos (evitar bucles, repetición de operaciones y cálculos innecesarios) en el back-end,

no recargar demasiado la página con elementos difíciles de renderizar (exceso de figuras, animaciones, objetos en 3D, etc.) en el front-end y contar con un dominio cuya respuesta sea aceptable, lo cual no dependerá del proceso de creado sino del proveedor que se elija.

8.2.1. *Single Page Application (SPA)*

La forma de generar las páginas está basada en el concepto en auge *Single Page Application*, el cual consiste en la escritura de nuevos datos y elementos a raíz de la interacción del usuario sobre la misma pestaña, ahorrando tiempos de renderizado en páginas totalmente nuevas. De esta forma, sobre la misma pestaña se va visualizando la evolución de los procesos a medida que el usuario va generando órdenes, haciendo de ésta un tipo de web más dinámica, visual y sencilla.

Su inconveniente es que no es posible acceder a cierto tipo de contenido desde una dirección específica sin contenido en la caché: se deberá de generar el contenido mediante órdenes para ver los resultados deseados. Esto supone lo que previamente se comentó en el apartado de la arquitectura, existe la posibilidad de visualizar simultáneamente dos pestañas con la misma dirección y con contenidos diferentes, todo ello dependiendo de las órdenes introducidas.

En el caso de *Satellite Viewer*, esto puede traducirse a la imposibilidad de generar un link en el que se visualice un satélite en concreto, o con unos ajustes específicos. Para los propósitos que esta plataforma está creada, no supone un gran impedimento, puesto que es muy sencillo producir cualquier situación que se desee en apenas un par de ‘clicks’.

Por otro lado, sí que conlleva muchas más ventajas: transiciones considerablemente más rápidas, sin necesidad de generar nuevamente todos los componentes (especialmente útil en el caso de las pestañas de mapas, que conllevan un tiempo de renderizado y procesado) y mayor sensación de aplicación nativa y versátil. Este modelo de página es justo el desarrollado por defecto por *frameworks* como *ReactJS*, una razón por la cual fue elegido como apoyo a la programación de la aplicación.

Es debido a lo explicado con respecto este tipo de página por lo que los apartados se diseñarán siguiendo esta estructura. No obstante, sí se crearán diferentes pestañas con el fin de no sobrecargarla y hacer una diferenciación notable entre las funcionalidades creadas: mapa 2D, mapa 3D, satélites, documentación, ‘about’, etc. De forma que desde la pestaña principal o ‘index’ (llamada numerosas ocasiones en esta memoria como Menú) se accederá a la pestaña deseada y entonces se crearán las peticiones a la página según el modelo SPA. Las variables que se gestionen en este tipo de pestaña son internas, no visibles al usuario, y con siempre con un valor por defecto a la hora de visualizar la página. Siendo, además, modificables con la entrada de valores del usuario y generando los respectivos cambios que se desee.

8.2.2. Ejemplos de diagrama de secuencia

La manera de funcionar de estas pestañas es descrita más concretamente en el capítulo 9, atendiendo a tareas concretas y su modo de desarrollo. Sin embargo, es importante mencionar el uso general de las mismas y la manera de operar con ellas, atendiendo, en primer lugar, al modo de declararlas y modificarlas (figura 8.3).

```
const intervaloPorDefecto = 10; //valor por defecto a la variable

const [interval, setInterval] = useState(intervaloPorDefecto);

export default function botonIntervalEjemplo() {
  <button onClick={() => { setInterval(20) }}>
  <a>Cambia Valor</a>
  </button>
};
```

Fig. 8.3 Código ejemplo para la declaración y modificación de una variable estado.

Tras la implementación de este código básico de ejemplo, la variable 'interval' pasa a valer '20', en lugar de '10', que fue el valor asignado por defecto. Esto sólo ocurrirá una vez se haga 'click' en el botón 'Cambia valor'. Gracias a este tipo de programación, es posible variar toda clase de parámetro de la página web, y es la forma fundamental en la que se alternan los valores, los componentes y todo elemento que forme parte de la página web.

Por ejemplo, podría crearse una variable que representara si un menú está desplegado o no, aportándole un valor 'booleano' (*true* o *false*), que cambiara en el momento en el que se pulsase, y que, con la condición de *true* sea capaz de renderizar las partes del menú, y con la condición de *false* no represente nada. Esta es la base de creación y modificación de las variables estado, que son constantemente utilizadas en el desarrollo de la plataforma.

8.3. Dominio

Atendiendo a la figura 8.2, es necesario el despliegue del software creado en un dominio que sea capaz de mostrar toda la capa de presentación. Esto es posible de forma muy sencilla localmente: 'subiendo' la plataforma en un puerto local, de forma que sea accesible desde el dispositivo que lo desarrolla, y no se tenga que pasar por el tedioso proceso de optimizar una página online y abierta al público. Si bien no tiene sentido crear una página divulgativa, didáctica y destinada para todos los públicos sin el fin de subirla a una dirección de acceso público en internet, en todo el proceso de creación no es una cuestión relevante: se necesita un compilador, un programa que permita el desarrollo del código y una base de

datos local u online. Una vez el proyecto es totalmente desarrollado, tal y como se comenta en el subapartado 2.1.2, puede estar sujeto a ‘ver la luz’ del público o no, decisión que se tomará en función de las necesidades e intenciones del desarrollador.

No obstante, debido a que el objetivo final es crear esta herramienta y que esté disponible para el público, es adquirido gracias al proveedor mundialmente conocido como ‘GoDaddy’ el dominio ‘www.satellite-viewer.com’ a fecha del 29 de agosto del 2022. Este dominio es mantenido por el desarrollador y se pretende usar como página de ‘*Hosting*’ (término que se utiliza para referirse a un dominio que ‘alberga’ un software en concreto).

8.4. Datos

Por último, a la hora de concretar de forma absoluta la estructura del proyecto, es importante hablar de la forma en la que se tendrá acceso a los datos necesarios para la generación de resultados. La forma estándar de acceder a información ‘en crudo’ de bases de datos, es conocida como ‘*fetching*’, (fetch en inglés, ‘buscar’ en español), la cuál se basa en lo siguiente:

- Parte de un soporte del que obtiene la información, el *Fetch API* (o simplemente *API*, *Application Programming Interface*) mediante el cual puede hacer una petición y obtenerla de forma online.
- Las peticiones (o ‘*Requests*’) serán en un formato concreto, generalmente proporcionados por el organismo que controla dicha API y accesibles mediante un enlace.
- Existen varios tipos de *APIs*: *REST* (el más común, basado en una comunicación cliente/servidor, trabaja con *XML*, *JSON*, *PHP* y texto), *RPC* (mientras que las anteriores se usan para la transferencia de datos, estas invocan acciones o procesos) y *SOAP* (parecido a *REST*, con la diferencia de que tiene una estructura mucho más fijada, cuenta con menos flexibilidad y sólo admite trabajar con *XML*).

Por razones obvias, en este proyecto se trata de implementar una *API* del tipo *REST* de la cual se pueda obtener la información de los satélites para poder generar las gráficas y el visualizado en los diferentes mapas.

8.4.1. API

Las fuentes usadas, al igual que se comentó en el subapartado 5.4.3, serán los proveedores *Space-Track* y *Celestrak*, ambas dotadas de información veraz, fiable y continuamente actualizadas. Una ventaja que tienen las dos fuentes es que se trata de organismos de gran reputación y cuya información supone el soporte de muchas plataformas online, por lo que es un incentivo para que el servidor esté la mayoría del tiempo activo: las probabilidades de no tener acceso

a la información son muy bajas, supondría una baja en un organismo internacional y cuya pertenencia se atribuye al gobierno de los Estados Unidos y a su Ejército del Aire.

En el capítulo 5 se especifica la información necesaria para la visualización de satélites en un mapa, los conocidos parámetros orbitales. También se explica que, gracias a la compañía NORAD, estos son debidamente condensados en un formato de datos conocido como TLE donde, además de estos parámetros orbitales, se tiene información sobre el número de catálogo, el nombre, la fecha de dicho TLE y otros valores relevantes como el número de revolución en dicha fecha o la clasificación del satélite. Los proveedores *Space-Track* y *Celestrak* han estado proporcionando públicamente durante años aquellos TLEs no clasificados como secretos. Dichos datos se pueden obtener en ambas plataformas de diferentes formas: en archivo de texto, *JSON* o *XML*, entre otros [47].

Gracias a estas plataformas, se crean las herramientas de *fetching* para disponer de estos TLEs y hacer uso de ellos en la aplicación de *Satellite Viewer*. La razón de usar dos proveedores recae en que, pese a ser ambos válidos y útiles, los dos tienen características únicas que los hacen útiles para la generación del proyecto:

- Se hará uso de *Celestrak* para la obtención general de TLEs, debido a que admite más formatos de salida [48]: *TLE* (2LE y 3LE), *OMM XML*, *OMM KVN*, *JSON*, *JSON PP* y *CSV*. En general, se obtienen de forma directa los datos de un formato *JSON* (o '*JSON PP*', formato que simplemente genera una visión más sencilla para el lector, pero supone lo mismo que *JSON* para *JavaScript*), por lo que es extremadamente ventajoso obtener datos en este formato. No obstante, también se requerirán en formato *TLE* para la visualización de información relevante y la generación de elementos visuales en la plataforma.
- Se hará uso de *Space-Track* para obtener información de interés relacionada con el tema, especialmente útil para la pestaña 'Satélites', donde se muestra información de todo tipo acerca de estos objetos. Tiene vistosos análisis sobre pertenencia geográfica de elementos orbitales, datos sobre basura espacial, clasificación por tipo de órbitas o datos sobre satélites 'caídos', entre otros.

En definitiva, es crucial tener acceso a la información proporcionada en ambas páginas, por lo que se generará la puerta de acceso con código del estilo al mostrado en el ejemplo de la figura 8.4.

```
export default function assignTLE(id) {  
  const [TLE, setTLE] = useState(tlePorDefecto);  
  const [TLEisLoading, setTLEisLoading] = useState(true)  
  fetch('https://tle.ivanstanojevic.me/api/tle/' + id)  
    .then((response) => response.json())
```



```
.then((TLE) => {  
  
  const fullTle = `${TLE.name}  
    ${TLE.line1}  
    ${TLE.line2}`;  
  setTLE(fullTle); //  Guardar datos  
  
  //setTLEIsLoading(false); //  Desactivar modo "cargando"  
});  
return(TLE) }
```

Fig. 8.4 Código ejemplo para la llamada a una API y el guardado de información.

En el código de la figura, además, se implementa una función que detecta si el *fetch* se ha efectuado correctamente, devolviendo el valor 'response' que alberga, generalmente, los valores [49]:

- 1xx: *Informational* –Comunica información de protocolo.
- 2xx: *Success* –Indica que la petición del cliente fue aceptada satisfactoriamente.
- 3xx: *Redirection* –Indica que el cliente debe efectuar alguna acción adicional para completar la petición.
- 4xx: *Client Error* – Comunica error por parte del cliente en la petición.
- 5xx: *Server Error* – Comunica error por parte del servidor en la petición.

Tras la completa definición de la estructura de la página, se trata de discutir las tareas que se han de llevar a cabo para la completa generación de la página en el capítulo 9, donde se explican en detalle algunas herramientas de utilidad para la plataforma.

CAPÍTULO 9. DESCRIPCIÓN DE TAREAS

9.1. Requisitos funcionales

Una vez definida la estructura de la plataforma, que es estrictamente necesaria para el desarrollo de *Satellite Viewer*, es el momento de desglosar el plan de acción que se ha de cumplir para obtener la página web deseada. Para ello, se ha decidido hacer una división radical en el proyecto entre los requisitos funcionales (aquellos con los que el usuario interactúa, es capaz de manipular y, de algún modo u otro, son ‘tangibles’) y los requisitos no funcionales (aquellos que, pese a que puedan ser apreciados, no son personalizables, ni tienen posibilidad para el usuario de generar interacción, más allá de la visión de la página y de su comodidad en esta).

En cuanto a los requisitos funcionales, es importante tener en cuenta las acciones que, en algún momento, el usuario efectuará en la página y que suponen una gestión interna de datos. Por ejemplo: la búsqueda de un satélite en concreto, la elección del mismo, la navegación por la pantalla o el almacenamiento en la base, entre otros.

En estos subapartados se discutirá la forma en la que se pretende abordar dichas tareas, de modo que cada utilidad esté creada y pensada de forma previa, antes de integrar todo en la futura página web. La ventaja de esta discusión previa es el ahorro de futuras incompatibilidades y, si cabe, la integración de todas estas en un marco que las haga lo más favorables e íntegras posible.

9.1.1. Disposición de satélites

Puesto que la aplicación es un software de visionado de satélites, es de vital importancia una pestaña dedicada exclusivamente a la exhibición de los mismos, donde se expongan todos aquellos rastreados, las familias existentes, los usos que estas tienen o el tipo de órbita que tiene cada uno.

Además, aparte de una pestaña única dedicada a esta visualización, se requiere de un menú lateral con el que se pueda interactuar en los mapas, para la visualización de datos y la ubicación de trazas, puntos e información relevante. A partir de este subapartado, a este menú lateral de búsqueda y utilidades se le atribuirá, por decisión del desarrollador y con el objetivo de que quede todo perfectamente clarificado, el nombre de *InfoBox*.

La herramienta InfoBox se creará con extremada cautela, puesto que se pretende que sea una herramienta única para ambos modos de visualizado (mapa 2D y mapa 3D). Se busca que tenga un diseño acorde con la página web y que contenga buscador, filtros de búsqueda, el listado de satélites disponible y el listado de familias.

Si bien se le pueden incluir más herramientas en un futuro, las mencionadas son requisitos fundamentales, y se establecen dentro del marco de objetos de obligada presencia en la plataforma aquí mostrada. A modo de explicación, para que clarificar el funcionamiento de la herramienta selectora InfoBox, se muestra la figura 9.1.

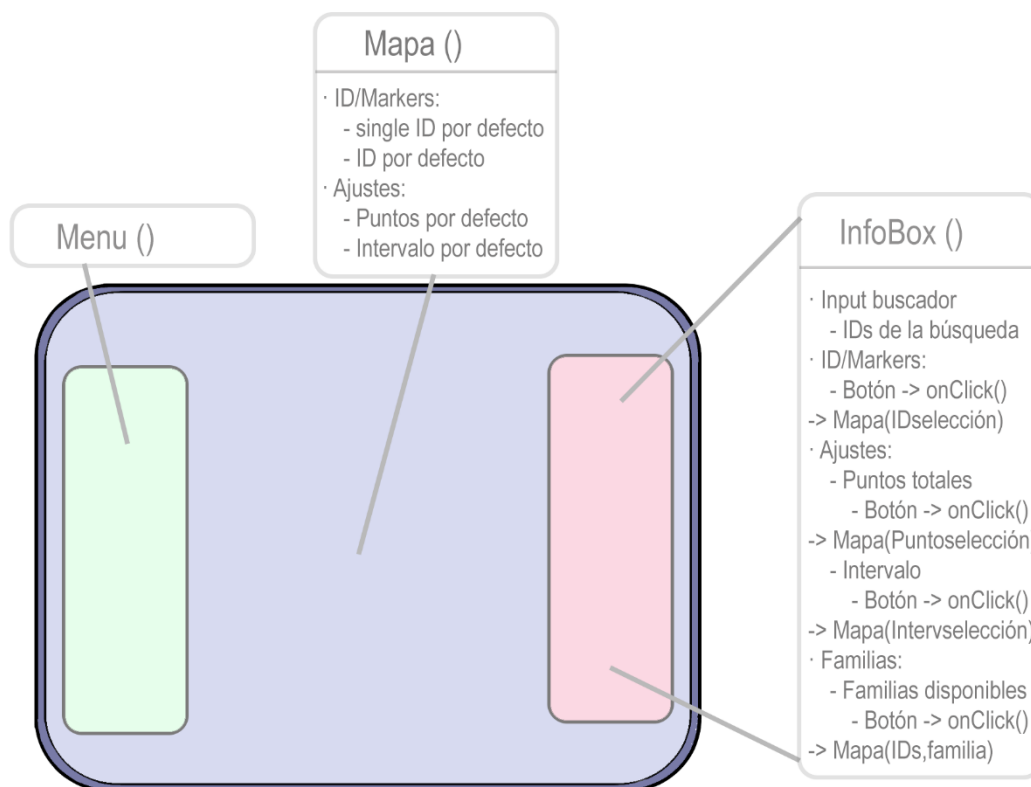


Fig. 9.1 Diagrama conceptual de la interacción InfoBox/Mapa.

La gestión de las opciones elegidas en esta pestaña será la misma para todas: el mapa en cuestión parte de una serie de variables estándar definidos en el primer renderizado, como el ID del satélite seleccionado, intervalos de frecuencia de la traza, número total de puntos de la traza, etc. El modo de funcionamiento estará basado en variables estado, funcionalidad clave del entorno ReactJS, las cuales están definidas en el subapartado 7.2.1.

9.1.2. Filtrado de información

Una barra de búsqueda es esencial en esta página web, habrá ocasiones en las que el usuario quiera hacer uso del nombre específico del satélite, veces en las que el NORAD ID (o número de catálogo, mencionado en el subapartado 5.4.2) sea el elemento de filtrado, o incluso ambas. Para ello, es necesario que exista una barra de búsqueda personalizada que sea capaz de hacer un filtrado (generalmente conocido en términos de programación como 'mapeo') personalizado.

Para ello, se ha de diseñar esta herramienta con la posibilidad de elegir el parámetro al cual se le hará dicho mapeo. Para facilitar la comprensión, se muestra en la figura 9.2 la tarea que se ha de desarrollar para esta búsqueda. En este caso a la función se le denomina *'input'* debido a que funciona dependiendo de los datos de entrada.

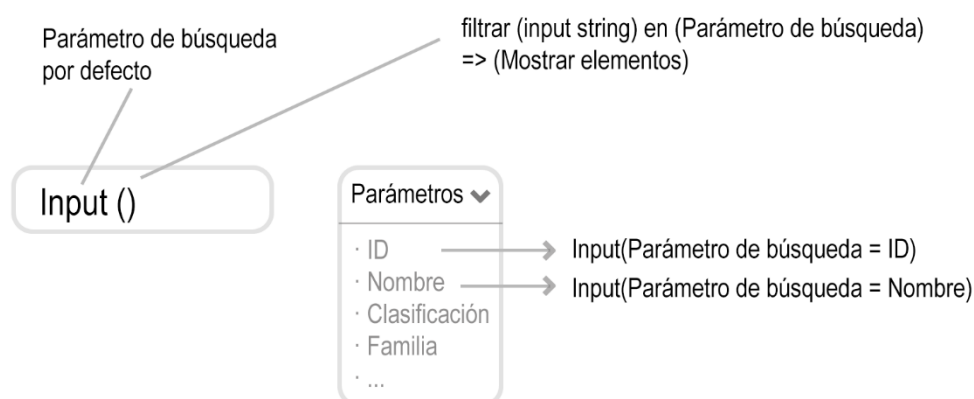


Fig. 9.2 Diagrama conceptual del funcionamiento de la búsqueda.

Cabe mencionar que, esta herramienta, una vez creada, puede servir para otro tipo de búsquedas. Por lo que puede ser realmente útil en las pestañas de documentación y satélites, así como fácilmente reutilizable.

9.1.3. Visualización de la traza de la órbita (2D y 3D)

En los subapartados 1.3.2, 5.3.3 y 5.4.3 se discute la fuente de donde se obtienen los datos satelitales, las librerías utilizadas en el proyecto para los cálculos oportunos y el significado de esta información recolectada. A través de la posesión de todos estos parámetros, se puede proceder al uso de los resultados deseados.

En los Anexos I y II se especifica concretamente el funcionamiento de las librerías escogidas para el procesamiento de la información referente a los satélites, por lo que aquí se hará simplemente referencia a las funciones utilizadas.

Mediante la función *'getLatLngObj'* es posible obtener en forma de objeto las propiedades *'lat'* y *'long'* (latitud y longitud) de un satélite (para un TLE y tiempo concreto dado). La razón por la que se requieren estos dos parámetros de entrada (cuya explicación se puede razonar a partir del subapartado 5.4.2) es que el TLE sufre una constante devaluación de exactitud a partir del instante en el que es creado. Por ello, se requiere tanto el TLE como el instante en el que se desea conocer estos parámetros. El resultado de introducir un TLE con mayor antigüedad para un mismo instante de tiempo supondría una diferencia de error debida a la deriva.

En la aplicación se desea conocer la posición de un satélite para el instante en el que se haga ‘click’ en él (posición instantánea), pero también es interesante saber la traza que éste describe con el tiempo, así como la que lleva describiendo en el pasado. La forma en la que se procederá en el proyecto será siempre partiendo de que el satélite se situará usando la fecha y tiempos exactos en los que se hace la petición, y, tras la asignación de un intervalo y un número total de puntos (que, si estos no son definidos, toman valores por defecto), se señalarán en el mapa marcadores que contengan el tiempo en el que dicho satélite ‘pasó’ o ‘pasará’ por la ubicación marcada. En la figura 9.3 se muestra el modo de proceder para esta tarea.

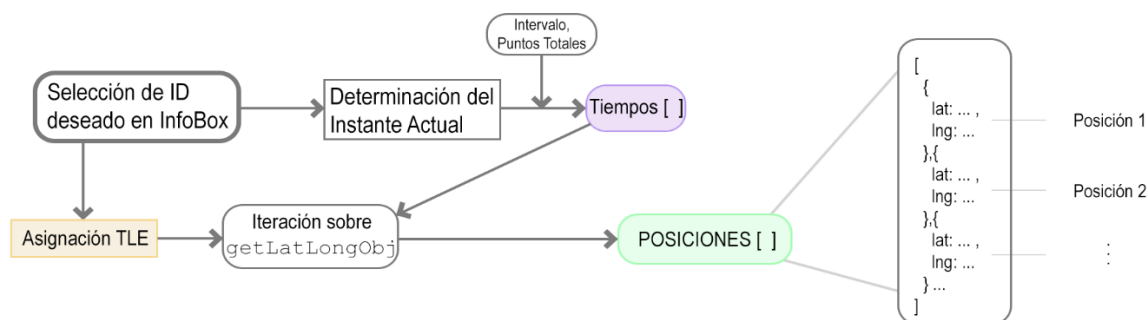


Fig. 9.3 Diagrama conceptual de la creación de las variables ‘tiempos’ y ‘posiciones’.

Tras la creación del objeto ‘posiciones’, es posible la puesta en marcha de la visualización de marcadores en ambos mapas. Para ello, se ha de hacer una mención a la forma se crear marcadores.

En el modo de dos dimensiones, la librería seleccionada (de lo cual se discute en el subapartado 5.3.1) es *Google Maps*. No obstante, para adaptarlo a la extensión *ReactJS*, las funciones y elementos del mapa serán calificados como componentes. Para la creación de los marcadores en esta librería se necesitan los parámetros de entrada:

- **Latitud** (‘lat’).
- **Longitud** (‘long’).
- **Imagen / texto.** Existe la posibilidad de asignar tanto una imagen, como un texto a la ubicación seleccionada.

Además, en la creación del componente es posible añadirle más características, por lo que se ha optado por añadirle una variable llamada ‘number’ y otra llamada ‘info’:

- **‘number’.** Guarda el número de marcador del que se trata, información útil a la hora de generar las variables y en la asignación de *Tooltips* [43].

- ‘info’. Cada marcador en la visualización de dos dimensiones estará dotado de un *Tooltip* (definido anteriormente) en el que se mostrará información importante sobre el marcador en concreto, lo cual hará que la experiencia del usuario y su interacción con el mapa sea mucho más visual y rápida. En este elemento se mostrarán: *ID* (NORAD ID), nombre del satélite, posición exacta y tiempo asignado a dicho marcador. La asignación de la posición y el tiempo a esta clase de elementos es extremadamente útil para hacer una comparativa temporal de la posición del satélite, así como para analizar visualmente los puntos de paso en tiempos pasados y futuros.

Para la visualización de la traza en el mapa de tres dimensiones se sigue un proceso casi idéntico, el cual no es necesario repetir, con la pequeña diferencia de que en la librería de este mapa (ArcGIS) se crea una línea que sigue los puntos que se determinen dentro de la traza (ya no será una sucesión de puntos, sino una ‘polilínea’ que una las ubicaciones calculadas previamente). Para ello, los parámetros de entrada para la definición de esta polilínea son:

- ‘geometry’. Variable que contiene los puntos por los que pasa la línea que se quiere definir, en ocasiones llamados *waypoints*. Este parámetro, a su vez, está dotado de las propiedades ‘type’ y ‘path’, que contienen el tipo de línea que se desea representar y los puntos en el formato deseado por ArcGIS.
- ‘symbol’. En esta variable se definen las propiedades relativas al elemento que se quiere representar, tales como: ancho de línea, color, tipo u opacidad, entre otros.

Finalmente, el conjunto completo de variables que se han de declarar para representar correctamente la traza en 3D supone mayor número de cálculos. A continuación, puede verse un ejemplo de este tipo de visualización, recogido en el cuadro de texto de la figura 9.4.

```
for (let j = 0; j < numeroDeWaypoints ; j++) {  
  markers[j]=[lat[j],lng[j]] }  
  
polylineProps = { type: "polyline",  
  paths: [markers] };  
  
lineSymbol = { type: "simple-line",  
  color: [156,152,247],  
  width: 1 };  
  
polyline1 = { geometry: polylineProps,  
  symbol: lineSymbol };
```

Fig. 9.4 Código ejemplo para la creación de una traza en librería ArcGIS.

Donde `'numeroDeWaypoints'` es el número total de puntos que se tendrán en cuenta para la traza, el cual es comentado en el subapartado 9.1.1 cuando se explican las funcionalidades de *InfoBox*.

9.1.4. Visualización de la órbita en 3D

Tras la explicación de cómo es posible la visualización de la traza que un satélite va marcando sobre la tierra (especialmente en el modelo 3D, el único en el que tiene sentido ver la órbita, puesto que en 2D no es nada intuitivo), es sencilla la explicación de la visualización en tres dimensiones: una vez se han obtenido los puntos de latitud y longitud, sólo se tendrá que añadir el valor de altitud y podrá verse la órbita.

Sin embargo, por muy sencillo que pueda parecer, es importante parar a pensar en los distintos tipos de representación explicados en el subapartado 5.4.2, en el que se habla de dos sistemas de referencia que son clave en este asunto: sistema de referencia inercial y sistema de referencia no inercial (ECI y ECEF, respectivamente). En este caso, en lugar de hablar de coordenadas X, Y y Z, se habla de latitud, longitud y altitud, un cambio muy sencillo que, además, es explicado en las ecuaciones 5.1, 5.2 y 5.3. La diferencia es que en este caso puede ser interesante, además, la conversión inversa (son necesarios los cambios de coordenadas cartesianas a esféricas y a la inversa). Para el caso de la altitud, parámetro de especial necesidad en esta tarea de vista 3D, se obtendría de sencilla forma que aparece en las ecuaciones 9.1 y 9.2.

$$r = \sqrt{x^2 + y^2 + z^2} \quad (9.1)$$

$$h = r - R_T \quad (9.2)$$

Donde h es la altitud, r es el radio desde el centro de la tierra hasta la posición del satélite y R_T es el propio radio de la tierra (generalmente, se hace uso del modelo terrestre WGS72, pero existen adaptaciones al modelo general WGS84 [44]).

La representación en ejes ECEF, aunque pueda parecer más compleja, y los resultados sean menos intuitivos, es prácticamente inmediata: se tienen los datos de latitud y longitud, y, además, mediante la función `'getSatelliteInfo'` (ver Anexo V) es posible la obtención del parámetro altitud, que utiliza las ecuaciones 9.1 y 9.2 (además de geometría más compleja). El resultado sería una sucesión de puntos con 3 coordenadas en las que la tierra es fija y se puede ver la evolución espacial del satélite.

En la representación ECI, sin embargo, se parte de la idea de que la tierra está en movimiento (ejes inerciales), por lo que la órbita del satélite se mantiene 'constante'. En la figura 9.5 se muestra un ejemplo en que se pretende mostrar la magnitud de la diferencia en ambas representaciones para un caso de órbita de alta excentricidad (caso en el que este fenómeno es más apreciable, puesto

que para órbitas cuya excentricidad sea similar a la de una circunferencia, no se vería diferencia alguna en la trayectoria).

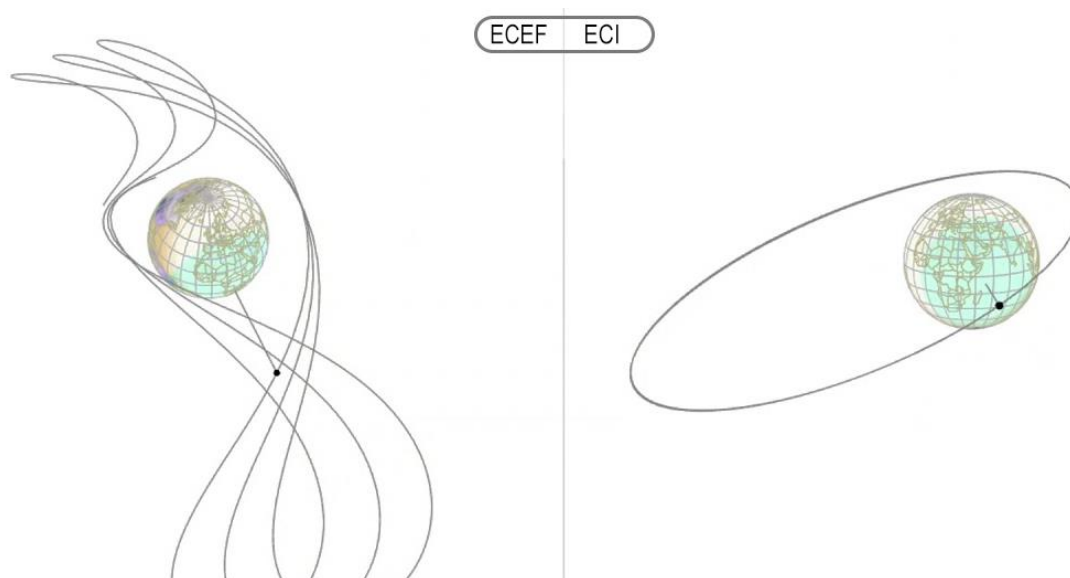


Fig. 9.5 Comparativa ejes inerciales vs ejes no inerciales de referencia [45].

Esta diferencia es causada por la naturaleza de los ejes: mientras que el ECI 'permite' el giro de la tierra, obtiene una trayectoria fija del satélite, es la tierra quien cambia su posición frente a esta elipse 'constante' en el tiempo; en la representación con los ejes fijos, puede apreciarse como la órbita que parecía constante en la representación inercial es, en realidad, una órbita cambiante en la visión desde la tierra, una visualización extremadamente útil para analizar la cobertura del satélite, su evolución temporal y las zonas que abarca (por ejemplo, en la creación de una constelación en la que se requiera cubrir unas zonas concretas).

Como ha sido comentado, la representación ECEF queda cubierta en el momento en el que se añaden los parámetros de altitud a los puntos calculados de la traza, por lo que es la representación ECI la que debe ser debidamente calculada y cumplimentada. Para ello, se han de pasar las posiciones de la elipse en 3D a coordenadas geodésicas y añadir la variable altitud. La conversión de coordenadas puede hacerse de varias formas: podría representarse una elipse directamente con los parámetros orbitales (lo cual es un cálculo matemático tedioso, que no imposible) o puede tenerse en cuenta la rotación de la tierra para transformar las posiciones a un sistema inercial. La forma en la que se implementa en la plataforma *Satellite Viewer* es la segunda. Se fuerza la órbita para describir sólo una vuelta por motivos de exactitud de cálculos: la velocidad de rotación de la tierra, pese a introducirse de la forma más 'exacta' que se conoce (con un período de 23 horas 56 minutos y 4 segundos) puede conllevar a errores en la representación. Siguiendo la ecuación 9.3 [46], se determina completamente esta transformación.

$$\theta_{ECI}(t) = \theta_{ECEF}(t) + \omega_T \cdot \Delta t \quad (9.3)$$

Donde $\theta_{ECI}(t)$ supone la longitud del satélite en el instante t en el sistema ECI, $\theta_{ECEF}(t)$ para el sistema ECEF, Δt es el incremento de tiempo desde el parámetro que se desea representar y ω_T es la velocidad angular de la tierra, calculada como se muestra en la ecuación 9.4.

$$\omega_T = \frac{360^\circ}{23h\ 56min\ 4s} = 4,17807 \cdot 10^{-6} \text{ }^\circ/ms \quad (9.4)$$

La razón por la que la velocidad es expresada en grados por milisegundo es sencilla: en la plataforma siempre se opera con grados y no radianes, así como el tiempo es operado en milisegundos, para hacer un convenio general en toda la página (aunque posteriormente sea reconvertido a formato DD/MM/AAAA, hh:mm:ss en UTC).

Tras la descripción de todas las herramientas que se deben realizar en forma de tareas, es momento de la implementación dentro de la página, que será visto en el capítulo 10 de contenidos.

9.2. Requisitos no funcionales

La página web, desde el comienzo de esta idea, es pensada como una plataforma en la que usuarios de todo tipo se sientan cómodos, por lo que el diseño es un factor esencial en la creación de la misma. No sólo se pretende que la interacción usuario-página sea lo más cómoda posible, sino que, además, debe ser útil para la proyección de datos técnicos y complejos. Para mantener ambos requisitos, los valores estéticos, funcionales y prácticos deben prevalecer en la totalidad de la producción.

9.2.1. Estética

Si bien no se tienen conocimientos de diseño avanzados para la idónea elección de los componentes de la aplicación, se parte de algunas herramientas de ayuda para que esto pueda realizarse de forma correcta.

Gracias al uso de la plataforma web Adobe Color, es posible tomar colores deseados y una familia de colores que se encuentran en sintonía con éste. Mediante los filtros de analogía, monocroma, triada, complementarios o sombras, entre otros, se puede hacer una elección de tonos de la página web que hace que el usuario vea más vistoso y destacable el contenido. Ya sea un proyecto austero de una escala de grises o un proyecto infantil en el que los colores vistosos son requeridos, Adobe Color se trata de la herramienta adecuada para hacer estas elecciones.

Las fuentes de texto elegidas vienen marcadas por Google Fonts, otra herramienta online que pone a disposición del desarrollador la fuente que desee,

de todo tipo de estilos, tamaños y complementos: existen opciones de relleno, deformación, estilizado, color, etc. Por lo que se hará uso de esta para la implementación de texto en la página web.

Las páginas web de hoy en día vienen siempre dotadas de un apartado de 'Tema', en el que, como mínimo, el usuario puede elegir entre tonalidades oscuras (ampliamente conocido como tema oscuro o tema nocturno) y tonalidades claras (tema claro o tema 'de día'). Por esta razón, *Satellite Viewer* vendrá con esta funcionalidad, donde el usuario pueda elegir entre, al menos, estos dos temas diferentes. Se introducirá también, en la medida de lo posible, esta funcionalidad para la vista de los mapas, para que estén acorde con el tema elegido.

Si bien los mapas por defecto de *Google Maps* y de *ArcGIS* tienen una estética agradable adaptada a toda clase de situaciones, en este caso se han elegido ambas opciones teniendo en cuenta que son casi enteramente personalizables. Se han hecho las siguientes modificaciones:

Modo Oscuro

- Luz invertida: conversión de tonalidades claras en oscuras, y viceversa.
- Color azul oscuro, con tono grisáceo (acorde con el resto de la página en este modo) para el terreno.
- Color azul levemente más oscuro para el agua, acorde con el colorido de la página en modo oscuro.
- Etiquetas de ubicación eliminadas, para facilitar la lectura de la información satelital y destacar el trazado de los objetos frente a las localizaciones por las que pasa (de las trazas se pueden ver fácilmente los parámetros de latitud y longitud).

Modo Claro

- Filtro de luz blanca para todo el contenido, acorde con el contenido del menú en este modo.
- Colores claros para el terreno.
- Tonos beige para agua, así como se hizo con los botones en este modo claro.
- Etiquetas de ubicación igualmente eliminadas, por la misma razón que en el modo noche.

El inconveniente principal del visualizado en tres dimensiones es que el fondo no es fácilmente personalizable, ni es estéticamente agradable que se vea

totalmente blanco, por lo que este visualizado permanecerá del mismo color en ambos temas.

Por último, cabe destacar la función generada para la atribución de colores a las órbitas: con el fin de hacer el visualizado más vistoso, se eligen varios tonos previamente y se implementa el código visto en la figura 9.6 para asociar un tono cualquiera al satélite deseado.

```
colores = [color_1, color_2, color_3, color_4, ..., color-n];

marker = {
  width: ...,
  text: ...,
  color: Math.floor(Math.rnd()*colores.length)
};
```

Fig. 9.6 Código para la generación de colores ‘aleatorios’ en marcadores.

9.2.2. Accesibilidad

La composición de la página es un tema que tiene un gran peso en la plataforma, puesto que será lo primero que vea el usuario al entrar en esta. Por ello se busca un diseño que no cree ningún tipo de confusión al interesado: debe ser capaz de ver la información relevante o las herramientas disponibles en la primera pasada.

Para conseguir esto, se diseña un menú en el que no existirá una gran cantidad de elementos: la pantalla de inicio debe estar dotada de botones de considerable tamaño donde se expongan claramente las diferentes pestañas de *Satellite Viewer*: documentación, satélites, mapa 2D y mapa 3D. Un esquema conceptual de la pestaña deseada se muestra en la figura 9.7.

A su vez, las pestañas de Mapa 2D y Mapa 3D tendrán la composición expuesta en la figura 9.1, donde la mayor parte de la pestaña estará destinada a los propios mapas. Se desea que así el usuario sea capaz de experimentar una mayor inmersión en el software, a la vez que se facilita la visión de los múltiples elementos que se han de visualizar. Crear mapas de mayor tamaño puede ayudar a que pequeños elementos sean más rápidamente detectables y la impresión de la página sea mejor.

Por último, las pestañas de satélites y documentación tienen como finalidad la de informar al usuario de una serie de datos de interés, por lo que la estructura deseada deberá ser bien clara a la vista.

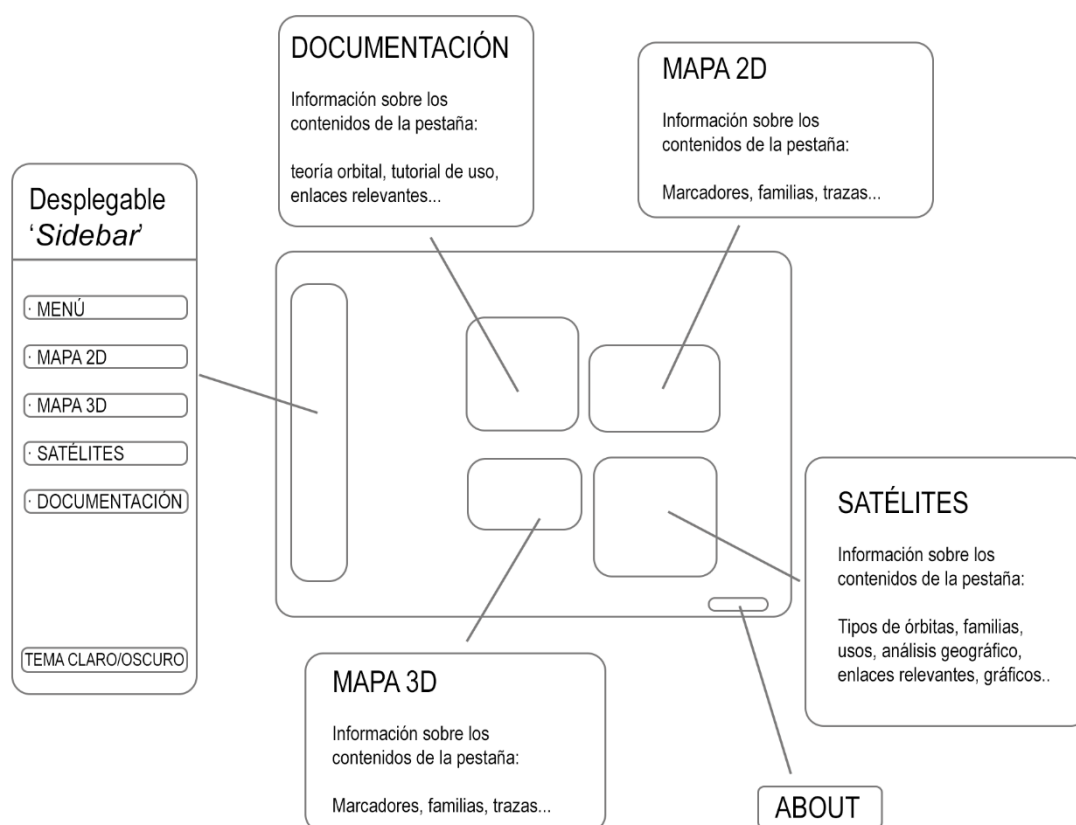


Fig. 9.7 Esquema visual de la pestaña inicio.

El menú desplegable en la parte lateral izquierda de la página estará dotado de un botón que dé animación a la funcionalidad que le da nombre: al pulsar sobre dicho botón, el menú se desplegará y será visto tal y como aparece en la figura 9.6. Por otro lado, cuando el contenido del menú no esté siendo en uso, éste será 'recogido' en un lateral y sólo algunos iconos referentes a las diferentes pestañas serán visibles.

9.2.3. Sencillez

Pese a la intención y la necesidad de mostrar gran cantidad de datos, gráficas e información, sobrecargar *Satellite Viewer* de elementos puede suponer un efecto contraproducente en la experiencia de uso. Por esta razón, se ha de ser consistente en la elección de espacios.

Existen multitud de funcionalidades que podrían ser vistas desde el elemento de InfoBox, no obstante, hacerlas visibles en todas las situaciones genera que el usuario desvíe la atención de los objetivos iniciales, y, finalmente, acaba saturándolo de estímulos y de información. Debido a esto, se ha decidido desarrollar toda la página con elementos desplegables con botón, para que el usuario sea quien elija en todo momento qué contenido quiere ver y cuándo.

En el caso de InfoBox, por ejemplo, donde existen numerosas divisiones (Búsqueda, Selección, Ajustes, Satélites y Familias), es necesario el uso de estos botones para desplegar contenido, puesto que en el momento en el que se esté interactuando con algunos satélites en concreto (por ejemplo, tras hacer una búsqueda) puede resultar molesta la visualización de todo el contenido: contraer las divisiones no deseadas efectuará una limpieza visual evidente.

Algunas herramientas de visualizado evolucionan integrando utilidades hasta el punto en el que resulta incómodo e intuitivo interactuar con la aplicación si no se tiene experiencia con ella, por lo que el apartado de sencillez es tenido en cuenta seriamente durante todo el proceso de producción de *Satellite Viewer*, siendo esta un pilar fundamental para el desarrollador y los usos para los que está destinada.

CAPÍTULO 10. CONTENIDO

10.1. Menú / *Sidebar*

10.2. Mapa 2D

10.2.1. Markers

10.2.2. Tooltips

10.2.3. Infobox

10.3. Mapa 3D

10.3.1. Traza

10.3.2. Órbita

10.3.3. Satélite

10.3.4. Infobox

10.4. Satélites

10.5. Documentación

10.6. About

CAPÍTULO 11. PRUEBAS Y EJEMPLOS

11.1. Tests

11.2. Ejemplos

11.2.1. Ejemplo de satélite en 2D

11.2.2. Ejemplo de familia de satélites 2D

11.2.3. Ejemplo de satélite en 3D

11.2.4. Ejemplo de familia de satélites 3D

CAPÍTULO 12. CONCLUSIONES

12.1. Conclusiones

12.2. Dificultades

12.3. Trabajos futuros

CAPÍTULO 13. BIBLIOGRAFÍA

ANEXO I

Columna 1	Columna 2	Columna 3	Columna 4	Columna 5
1	A	101	Z	a
2	B	102	Y	b
3	C	103	X	c
4	D	104	W	d

ANEXO II

Columna 1	Columna 2	Columna 3	Columna 4	Columna 5
1	A	101	Z	a
2	B	102	Y	b
3	C	103	X	c
4	D	104	W	d