Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

DESARROLLO DE APLICACIONES MEDIANTE LA PLATAFORMA GLITCH

Alumno: Francisco Javier Merelo Vacas

Tutor: Prof. D. Víctor Manuel Rivas Santos
Dpto: Departamento de informática

Junio, 2023

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Victor Manuel Rivas Santos, tutor del Trabajo Fin de Grado titulado: Desarrollo de aplicaciones mediante la plataforma Glitch, que presenta Francisco Javier Merelo Vacas, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, junio de 2023

El alumno: El tutor:

Francisco Javier Merelo Vacas Victor Manuel Rivas Santos

Agradecimientos

Con este trabajo pongo fin a mi experiencia en el Grado de Ingeniería Informática. Aprovecho este espacio para expresar mi agradecimiento a aquellos que me han ayudado a llegar hasta aquí.

En primer lugar, a mis padres, no sólo por su cariño y protección, también por haberme trasmitido los valores de la constancia y el esfuerzo.

A mis amigos, por haberme animado a no tirar la toalla en los momentos difíciles de estos años, sin ellos llegar hasta aquí no hubiese sido posible.

Por último, a mi tutor Víctor, por poner tanta ilusión en este proyecto como yo en mi aprendizaje.

Por todas esas cosas y las que mi corazón no es capaz de expresar, gracias, doblemente gracias.

Índice

Índice de ilustraciones

Índice de tablas

1. INTRODUCCIÓN

Desde los inicios de la *World Wide Web* en la década de los noventa el porcentaje de usuarios que utilizan internet ha ido siempre en aumento llegando a representar actualmente al 69% (Miniwatts Marketing Group , s.f.) de la población mundial. El número de webs, por lo tanto, ha ido creciendo de manera exponencial y existen cinco mil millones (Internet Live Stats, s.f.) de ellas elaboradas con diferentes tecnologías e ideadas con diversas finalidades.

Durante el proceso de evolución de la web, hemos pasado de las páginas estáticas que ofrecían información básica, a las aplicaciones web más dinámicas y complejas que permiten la interacción con los usuarios y ofrecen funcionalidades avanzadas. Esta evolución ha sido impulsada por varios factores, como el avance de las tecnologías web, el aumento de la demanda de interactividad por parte de los usuarios y la necesidad de brindar experiencias más enriquecedoras. Finalmente, las aplicaciones web se han convertido en poderosas herramientas para el comercio electrónico, la colaboración en línea, la gestión de contenidos y muchas otras áreas. En este contexto, JavaScript ha jugado un papel fundamental debido a su capacidad para ejecutarse en el lado del cliente, es decir, en los navegadores web de los usuarios, permitiendo así que gran parte del procesamiento se realice en el lado del cliente.

El presente Trabajo de Fin de Grado abordaremos el uso de una herramienta comercial denominada Glitch, utilizada para el desarrollo de aplicaciones web de varios tipos. Asimismo, evaluaremos el grado de dificultad de uso de la herramienta a través del desarrollo de una aplicación web para el etiquetado de imágenes.

1.1 Objetivos

1.1.1 Objetivo general

El objetivo del proyecto es estudiar y documentar las herramientas y recursos de la plataforma de desarrollo de aplicaciones Glitch. Se quiere estudiar hasta qué punto es viable el uso de la plataforma de desarrollo por parte de usuarios con conocimientos técnicos escasos. Además,

desarrollaremos una aplicación web para mostrar las posibilidades de la plataforma.

1.1.2 Objetivos específicos

Para lograr el objetivo general establecido, se deberán alcanzar los siguientes objetivos específicos, los cuales serán fundamentales para el correcto desarrollo del trabajo:

1. Conocer la herramienta y su interfaz, su historia y creadores.
2. Realizar un estudio sobre las tecnologías con las que trabaja la plataforma.
3. Buscar alternativas semejantes a la estudiada.
4. Desarrollar una aplicación web utilizando los servicios de Glitch.
5. Seleccionar y utilizar una metodología ágil para guiar el desarrollo de la aplicación.
6. Documentar el proceso de desarrollo de la aplicación y los resultados, para garantizar la transparencia y la continuidad del proyecto.

1.2 Motivación

El motivo que me llevó a escoger este Trabajo de Fin de Grado fue que, observando el actual panorama de desarrollo de aplicaciones, existen múltiples tecnologías que permiten realizar aplicaciones semejantes. Me sedujo la idea de investigar una plataforma que aunase diferentes tecnologías de desarrollo. Tras analizar brevemente la web de la plataforma fui consciente de que uno de los objetivos de ésta es lograr una gran accesibilidad y facilidad de uso; cualquier persona con un equipamiento básico y acceso a internet puede utilizarla para construir su web sin necesidad de invertir dinero en software ni en hardware. Esto me pareció interesante en un mundo globalizado donde la brecha digital es una realidad.

Por otro lado, coincidí con el profesor que tutoriza el trabajo en una de las asignaturas del grado, por lo que asumía que con la realización del mismo iba a

suponer una gran experiencia de aprendizaje en mi paso por la Universidad de Jaén.

2. GLITCH

2.1 Conceptos previos

El apartado de "Conceptos Previos" desempeña un papel fundamental en este trabajo, ya que nos permite establecer una base sólida de comprensión de los conceptos fundamentales necesarios para abordar el desarrollo de aplicaciones mediante la plataforma Glitch.

Al entender la naturaleza de una aplicación web, el rol de una plataforma PaaS y la dinámica de interacción entre el cliente y el servidor, estaremos en mejores condiciones para explotar las características y ventajas que Glitch ofrece en el proceso de desarrollo de aplicaciones web.

Aplicaciones web

Podemos definir las aplicaciones web como aplicaciones que no necesitan instalación por parte de los usuarios, ya que se ejecutan en un servidor remoto, al cual se accede a través de internet mediante a un navegador web.

De forma usual, las aplicaciones web se diseñan bajo un patrón que divide la aplicación en tres niveles, este recibe el nombre de arquitectura modelo-vista-controlador (MVC). El primer nivel es el modelo, que se encarga de la gestión y estructuración de los datos utilizados por la aplicación. Luego, está la vista, que se ocupa de la presentación visual de la aplicación, garantizando una interfaz atractiva y accesible para los usuarios. Por último, tenemos el controlador, que actúa como intermediario entre los usuarios y la aplicación, gestionando las solicitudes y proporcionando las vistas necesarias en respuesta a esas solicitudes. La arquitectura MVC busca una clara separación de responsabilidades y promueve una estructura modular y escalable para el desarrollo de aplicaciones web.

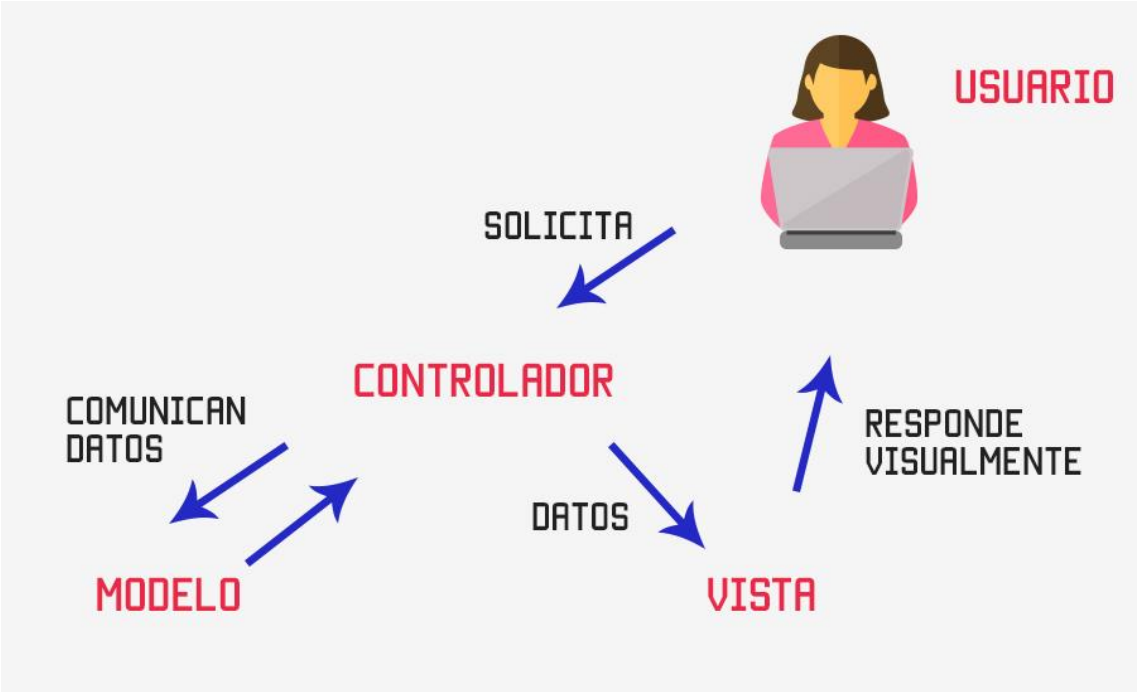

Ilustración 1. Esquema MVC. Fuente: codigofacilito.com

Arquitectura cliente-servidor

La arquitectura cliente-servidor es fundamental para el funcionamiento de las aplicaciones web. En esta arquitectura, el cliente, que es el navegador web utilizado por el usuario, solicita información o recursos al servidor remoto donde se encuentra alojada la aplicación web. El servidor procesa la solicitud del cliente y devuelve la respuesta correspondiente, que puede ser una página web, datos, imágenes u otros recursos.

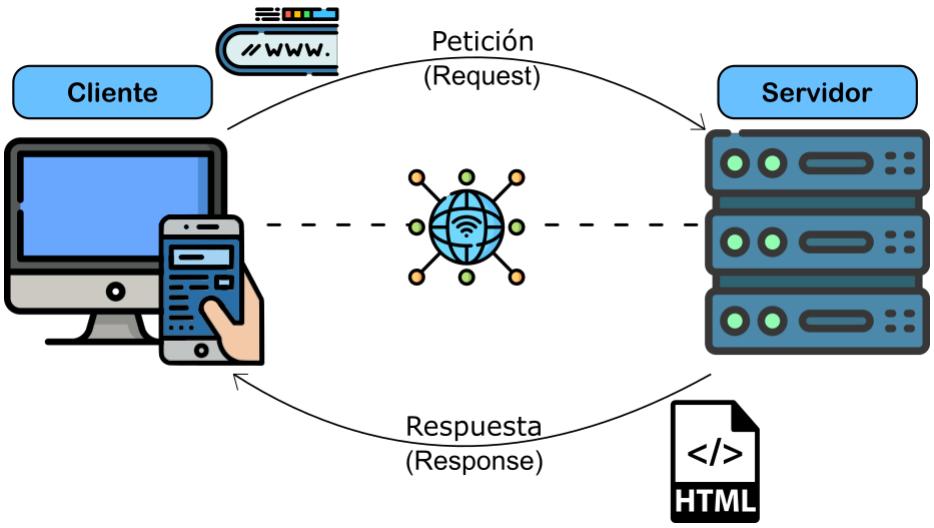

Ilustración 2. Esquema de la arquitectura cliente - servidor

Un sistema cliente-servidor es según (Luján-Mora, 2002):

> "*Una combinación de la parte cliente (también llamada front-end) que interactúa con el usuario (hace de interfaz entre el usuario y el resto de la aplicación) y la parte servidor (o back-end) que interactúa con los recursos compartidos*"

PaaS

Como hemos visto, una aplicación web requiere de un servidor para su ejecución, ya que necesita un entorno de alojamiento donde se puedan procesar las solicitudes de los usuarios y servir el contenido solicitado. Aquí es donde entran en juego los servicios de Plataforma como Servicio (PaaS, por sus siglas en inglés). Un PaaS es un entorno de desarrollo y despliegue de aplicaciones que brinda a los desarrolladores una infraestructura lista para usar, sin tener que preocuparse por la configuración y el mantenimiento del servidor. Dentro del Cloud Computing existen más modelos de servicio en la nube (ver *Ilustración 3*), sin embargo, no los detallaremos puesto que no son objeto de estudio de este trabajo.

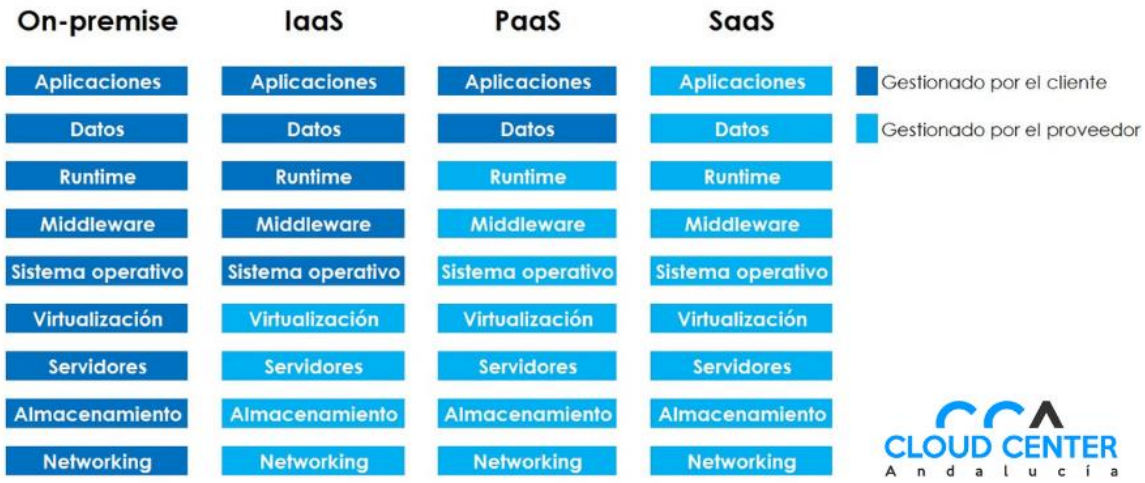

Ilustración 3. Modelos de servicio en la nube. Fuente: www.cloudcenterandalucia.es

2.2 ¿Qué es Glitch?

Glitch es una plataforma de desarrollo de *webapps* en línea, ofrece un entorno de desarrollo colaborativo y accesible que permite elaborar aplicaciones de manera rápida y sencilla. La plataforma ofrece gran cantidad

de recursos y herramientas incluyendo: un editor de código en línea, plantillas prediseñadas, componentes, paquetes y una consola de comandos en línea que permite la gestión de los proyectos. En cuanto a los lenguajes con los que permite trabajar, oficialmente soporta HTML, CSS, JavaScript, y Node.js (Glitch, s.f.), sin embargo, algunos usuarios afirman poder trabajar con otros como Java, .NET, Python, GO, Rust y lenguajes compilados a JavaScript como TypeScript y CoffeeScript (Berry, 2018). Los desarrolladores pueden además crear una aplicación del lado del servidor con *Node.js* o una web estática con una interfaz de usuario elaborada con *ReactJS*.

La variedad de herramientas que integran la plataforma es muy amplia. Así, permite la integración de aplicaciones de terceros como el conocido repositorio de código GitHub para la importación de proyectos. También permite el despliegue a DigitalOcean para una mayor escalabilidad o, en caso de ser web estática sin necesidad de procesamiento en servidor, a Fastly (Foro Red Hat, 2021). También permite la descarga del proyecto para su desarrollo de forma local.

La plataforma presenta una serie de funcionalidades que la caracterizan; en primer lugar, no es necesario tener conocimientos avanzados sobre IT, puesto que el despliegue de la aplicación se hace de forma automática, lo que implica que nuestra web estará disponible desde el inicio del proyecto. Según vayamos avanzando en el proyecto, podremos ir contemplando los cambios en tiempo real, tanto en la apariencia de la página o de la aplicación, como en los archivos que afectan el funcionamiento del servidor. Además, al ser una herramienta online, no es necesaria la instalación de ningún software en nuestro equipo, simplemente necesitaremos iniciar sesión en ella para empezar a desarrollar.

De forma predeterminada, la máquina virtual sobre la que funciona nuestra webapp tendrá una dirección web única generada automáticamente, además esta será accesible a través de HTTPS ya que la plataforma se encarga de cifrarla con un certificado SSL propio. La herramienta ofrece además la posibilidad de establecer nombre de dominios personalizados, lamentablemente no hemos podido estudiar esta funcionalidad debido a que

desde el día 1 de marzo de 2023, por problemas con el proveedor de certificados SSL, se encuentra desactivada de forma indefinida. (Glitch, s.f.)

Glitch se centra en la creencia de que el desarrollo de aplicaciones web debe ser accesible y colaborativo para todos, independientemente de su nivel de habilidad técnica. La colaboración en la plataforma es en tiempo real, permite a los usuarios trabajar juntos en un mismo proyecto facilitando la resolución de problemas en equipo; también fomenta el código abierto permitiendo que los usuarios puedan compartir libremente sus aplicaciones para ser imitadas o editadas por otros, lo que ha permitido crear una comunidad muy activa de desarrolladores. De esta filosofía nace la funcionalidad *REMIX (ver Ilustración 4),* esta se encuentra cuando se está explorando otros proyectos de la comunidad, dentro del apartado *discover*. Al seleccionar esta opción los usuarios pueden crear una copia del proyecto seleccionado en su propia cuenta, lo que les permite modificar y personalizar el código y probar nuevas ideas sin afectar el proyecto original.

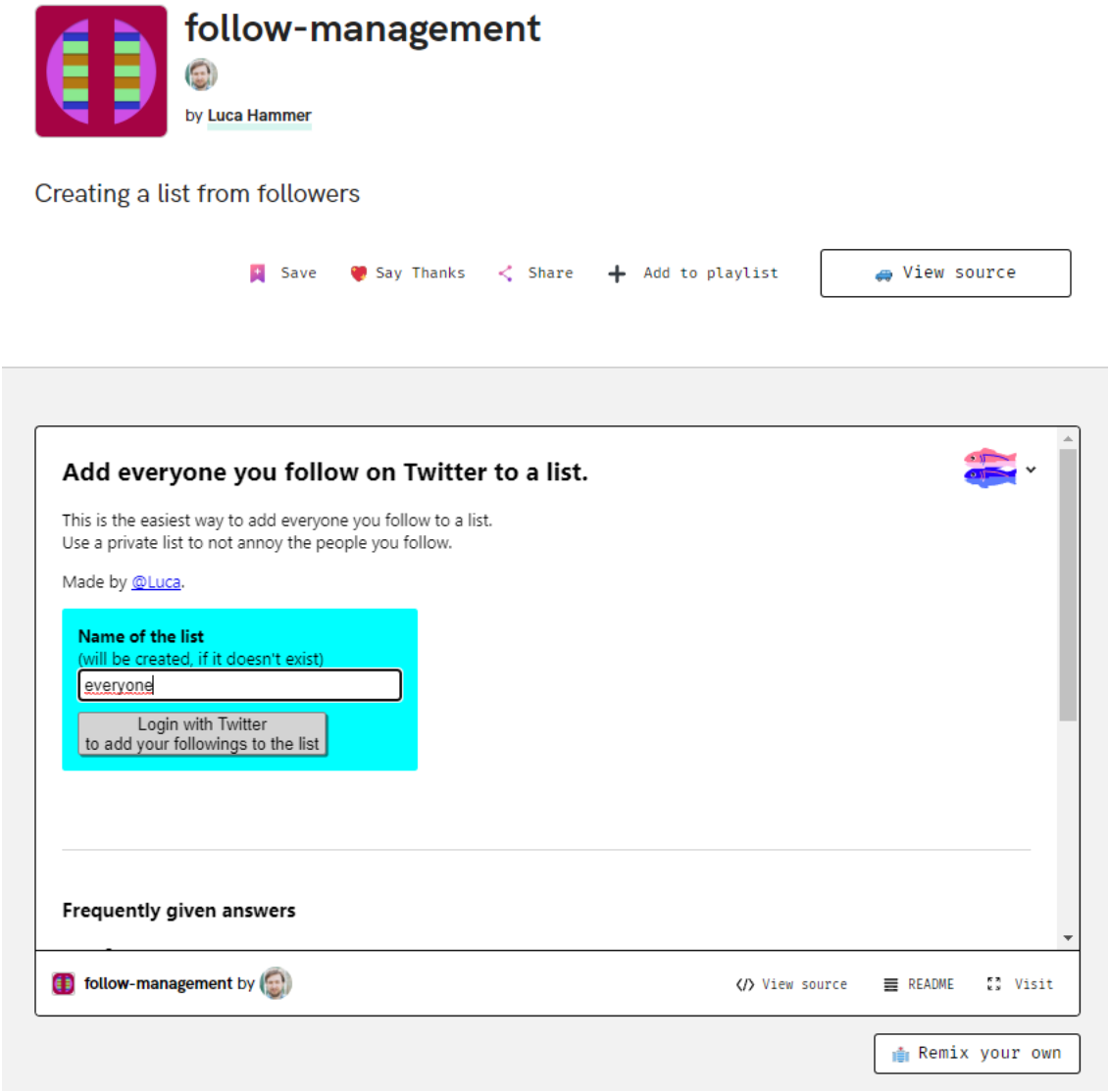

Ilustración 4. Vista de un proyecto de la comunidad y el botón *Remix*

Tras la plataforma están presentes grandes desarrolladores y empresarios de otras herramientas de éxito. El autor de la herramienta, el ingeniero de software Joel Spolsky, fue responsable también de la creación de otras famosas herramientas como *Trello* y *Stack Overflow* donde participó como CEO hasta 2019 (Spolsky, Joel On Software, s.f.). El actual CEO de la compañía es Anil Dash una figura destacada en la industria tecnológica y conocido por su activismo político y social; entre sus logros laborales se encuentra haber sido asesor de la Oficina de Estrategia Digital de la casa Blanca cuando Obama la presidia. (Dash, s.f.).

La aplicación sigue un modelo de negocio *Freemium*, es decir, se ofrecen a los potenciales clientes las funcionalidades más básicas de forma gratuita.

Para aquellas más avanzadas, se sigue un modelo de subscripción mensual o anual, de esta forma con una cuenta de pago podremos mejorar las prestaciones, o acelerar, como se denomina en la plataforma. Actualmente el precio de la suscripción a Glitch pro puede pagarse a través de dos formas: con pagos mensuales de 10 dólares o con un pago anual de 96 dólares estadounidenses. (Glitch, s.f.)

Con una cuenta *Pro* es posible acelerar hasta 5 aplicaciones y disfrutar de las características que podemos observar en la *Tabla 1*, que muestra las funcionalidades de pago.

	Funcionalidades gratuitas (plan starter)	**Funcionalidades de pago (plan Pro)**
Privacidad	Proyectos y códigos públicos	Proyectos privados
Tiempo de actividad	Aplicaciones que se duermen tras 5 minutos de inactividad	Aplicaciones siempre activas
Memoria RAM	512MB por cuenta	Hasta 2GB por cuenta
Espacio en Disco	200MB por proyecto	400MB por proyecto acelerado, 200 MB para el resto
Límite en la frecuencia de visita	4000 peticiones por hora	Ilimitado para todos los proyectos
Horas de proyecto	1000 horas mensuales	Ilimitadas para los proyectos acelerados, 4000 horas mensuales para el resto

Tabla 1. Comparativa entre los planes de la aplicación

La *Tabla* 1 nos muestra una comparativa entre los distintos tipos de cuenta que pueden obtenerse al registrarse en la aplicación, sin embargo, existen restricciones técnicas a tener en cuenta (Glitch, s.f.), las cuales se comentarán a continuación:

- Las horas de proyecto no sólo se contabilizan con las horas de edición de código durante el desarrollo; en caso de que se trate de una aplicación web dinámica, a este tiempo habrá que sumarle el tiempo que cada usuario esté utilizando nuestra aplicación.
- Del espacio en disco que ocupa cada proyecto no se tendrá en cuenta aquello que se encuentre en la carpeta */tmp* que se eliminará con cada reinicio de la aplicación, tampoco los módulos de *node.js* que cuentan con un extra de 1GB para ser instalados, ni los *assets* necesarios para la web que cuentan con un extra de 512MB de espacio, permitiendo como máximo archivos de 256MB.
- Una vez se alcancen el número máximo de peticiones, el servidor responderá con un código de estado HTTP 429 “Demasiadas solicitudes”.
- Todos los proyectos, estén acelerados o no, son reiniciados cada doce horas aproximadamente para recibir actualizaciones de software y los desarrolladores no pueden programar estos reinicios.

Todas estas limitaciones pueden parecer demasiado restrictivas y hacen pensar que la plataforma está diseñada principalmente para proyectos de pequeña escala. Sin embargo, Glitch también ofrece opciones para soluciones más profesionales, a través de un formulario de contacto los usuarios pueden comunicarse con el equipo de soporte de Glitch para solicitar ayuda con aquellos proyectos más grandes y complejos. El equipo de soporte también ofrece servicios de consultoría y soluciones personalizadas para empresas y organizaciones que necesitan herramientas de desarrollo en la nube más avanzadas y escalables.

2.3 Historia de la herramienta

Fue creada por la firma Fog Creek Software de Joel Spolsky y Michael Pryor, nació en 2016 como “un experimento de los desarrolladores para los desarrolladores, que eliminase el punto de fricción de empezar algo nuevo” (Glitch, s.f.).

Existió previamente una fase beta del proyecto bajo el nombre HyperDev (Spolsky, Joel on software , 2016) que meses después comenzó a llamarse Gomix (Dash, Medium, 2016) para finalmente ser reconocida bajo el nombre de Glitch, el motivo que impulsó el cambio fue que tras escuchar a la comunidad el anterior alias evocaba palabras mal sonantes para la comunidad rusa (Dash, Medium, 2017). En septiembre de 2018, este producto se convirtió en el buque insignia de la empresa que decidió renombrarse como Glitch Inc.

La conocida empresa Mozilla anunció en diciembre de 2018 que retiraría su producto Thimble un editor de código con filosofía educacional, por lo que ambas compañías colaboraron para permitir la migración de proyectos entre las plataformas lo que supuso un gran incremento de proyectos en la plataforma (Glitch, 2018). En abril de 2020 se introdujo en la herramienta un plan de pago aún hoy disponible que permite a los usuarios obtener mayor espacio en disco, una mejora en la memoria RAM y tráfico ilimitado a cambio de una suscripción mensual (McEwen, 2020).

Recientemente, en mayo de 2022, la empresa ha sido adquirida por Fastly (Sharma, 2022) una empresa encargada de una red de entrega de contenido (CDN) lo que ha permitido a la empresa renovar su plantilla (Kastrenakes, 2020).

3. ANÁLISIS DE LA HERRAMIENTA

3.1 Tipos de proyecto que permite

En el momento de empezar un proyecto utilizando Glitch, la propia plataforma ofrece distintos proyectos que pueden servirnos de base en función

de la tecnología que queramos utilizar, por lo que podremos elegir crear un proyecto en blanco desde cero, o partir de alguna plantilla.

En el momento de iniciar un nuevo proyecto, la herramienta nos ofrecerá trabajar con las siguientes tecnologías:

- *Basic Website*: Será un sitio web estático con las tecnologías web más básicas: un fichero principal HTML, un fichero de hojas de estilo (CSS) y un fichero Javascript que dé interactividad a la aplicación
- *Glitch in Bio*: Una web estática, que permite redireccionar a distintos enlaces personales configurables de interés.
- *Hello Node!:* Una aplicación *full-stack* con Node.js donde podremos configurar el comportamiento del lado del servidor, podremos crear una API o una aplicación web
- *React:* Permite generar una web estática utilizando el conocido *framework* de JavaScript: React.js.
- *Blog with Eleventy:* Genera una web estática que servirá como blog personal para postear artículos a través de diversos lenguajes de marcado. Se ofrecen más detalles de esta tecnología en el apartado *3.3.2.*
- *SQLite:* Permite crear una base de datos persistente SQLite a través de un servidor Node.js.
- *Import from Github:* Permite importar el código de nuestro proyecto desde el repositorio de código más famoso de internet.

3.2 Interfaz de la herramienta

Una vez aclarado el contexto en el que surge Glitch y conocer su propósito, procedemos a iniciar sesión con nuestra cuenta y analizar la herramienta.

La exploración por la web se realiza principalmente a través de la barra superior de navegación en la que se encuentran las pestañas que nos darán acceso a las diferentes vistas.

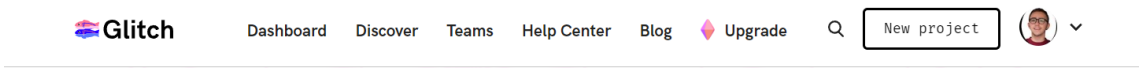

Ilustración 5. Barra de navegación de la aplicación Glitch

Además, desde aquí podremos crear un nuevo proyecto de entre las distintas opciones que ofrece la plataforma (ver *Ilustración 6*).

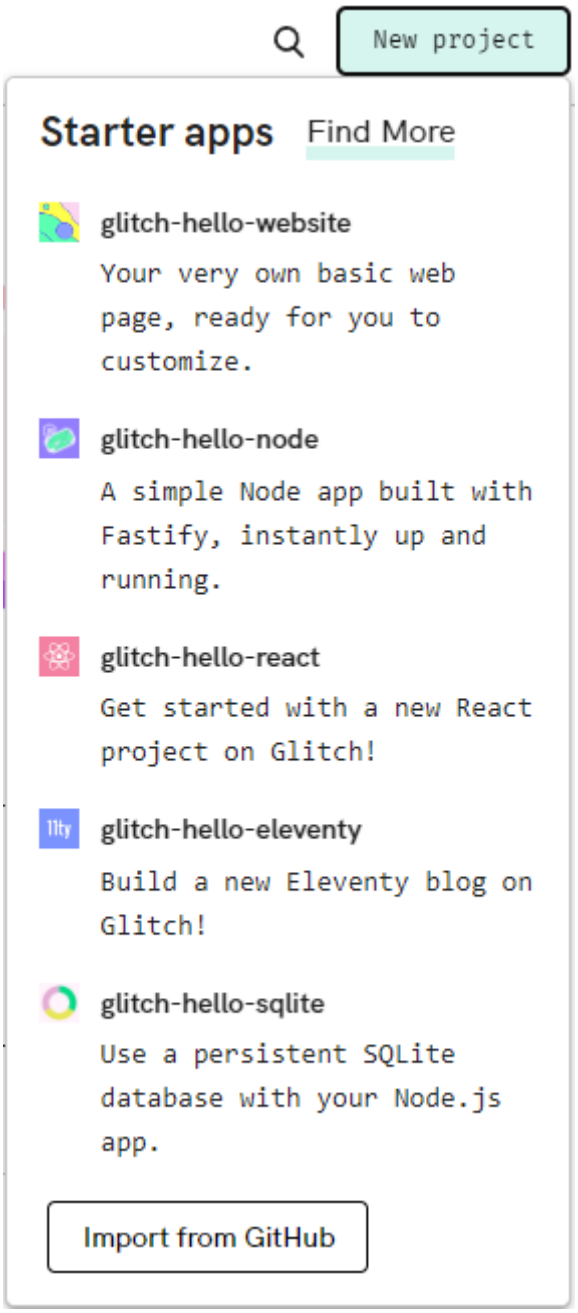

Ilustración 6. Posibles proyectos para crear

La pestaña principal es el dashboard (*Ilustración* 7). Esta vista es nuestro principal panel de control y en ella podremos obtener un listado de los proyectos activos que tenemos, así como controlar las horas restantes

disponibles para gestionarlos. Además, será desde esta vista desde donde accederemos a la edición de los proyectos y podremos controlar cuáles queremos acelerar en caso de que dispongamos de un plan de pago.

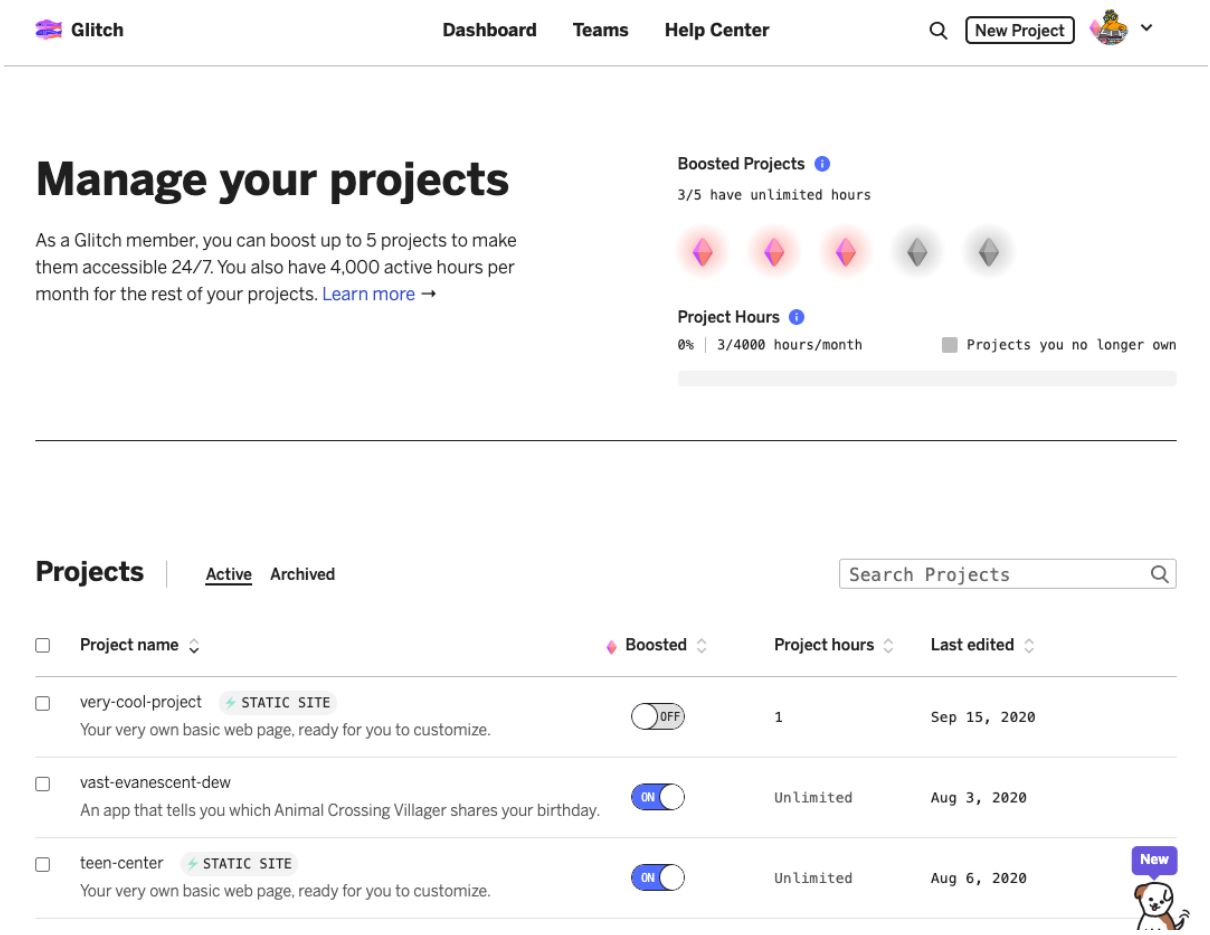

Ilustración 7. Vista *Dashboard* con Glitch Pro. Imagen tomada de: help.glitch.com

Una vez accedemos a uno de nuestros proyectos, se nos muestra el propio editor de código (*Ilustración 8*), así como las distintas herramientas y funcionalidades que ayudan a los desarrolladores en la creación de páginas web. A continuación, procedemos a analizar y explicar cada una de ellas.

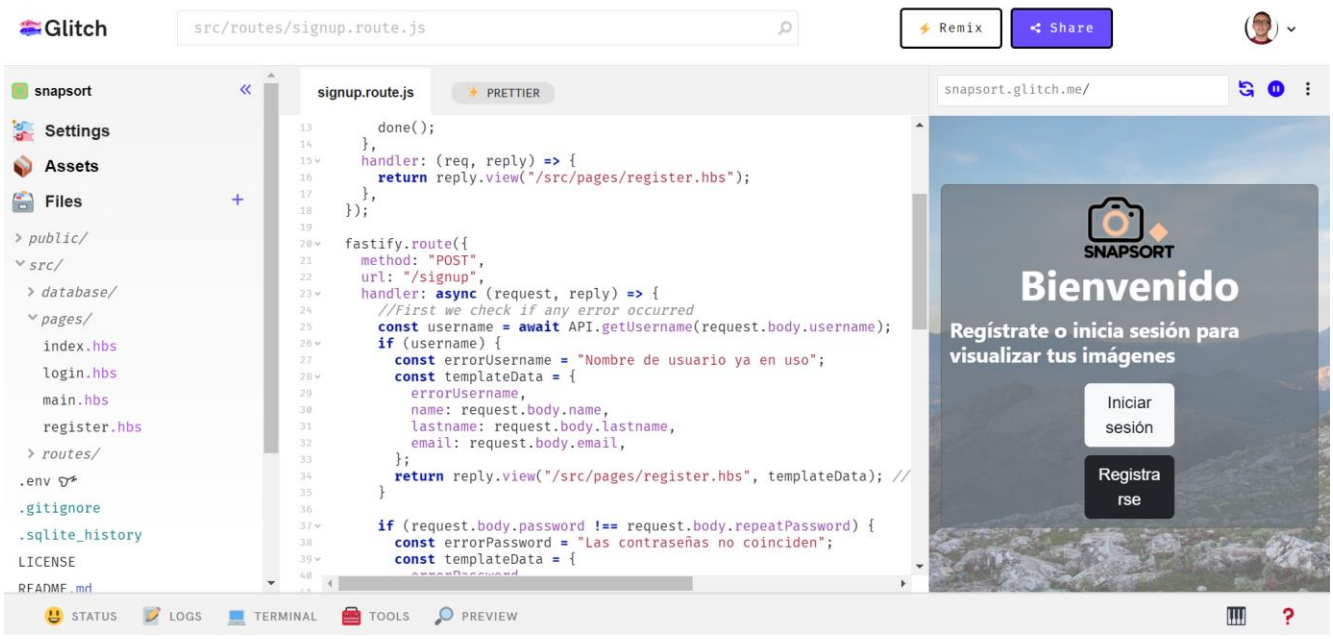

Ilustración 8. Vista general de la pantalla de edición

En la parte central de la pantalla encontramos el editor de código. En él se muestra el contenido del fichero que se encuentre seleccionado en el panel lateral. Cabe destacar que Glitch detecta de qué tipo de fichero se trata resaltando las palabras reservadas de cada lenguaje. Además, en la parte superior podemos acceder a la herramienta *PRETTIER* que de manera automática tabulará el código para que sea más legible.

En el lado izquierdo encontramos el menú lateral de proyecto, desde el cual podremos acceder a distintas funcionalidades:

- *Assets:* nos permitirá importar las imágenes que sean necesarias para nuestra aplicación. Además, genera un enlace CDN para acceder a estos recursos desde fuera de la herramienta
- *Settings:* en este menú también podremos gestionar la configuración del proyecto (ver *Ilustración 9*). A través de la ventana de configuración podremos realizar algunos ajustes básicos; como modificar el nombre y la descripción del proyecto, así como activar/desactivar el modo oscuro del editor. También podremos embeber la aplicación para insertarla en una web a través de HTML y crear una copia a través.

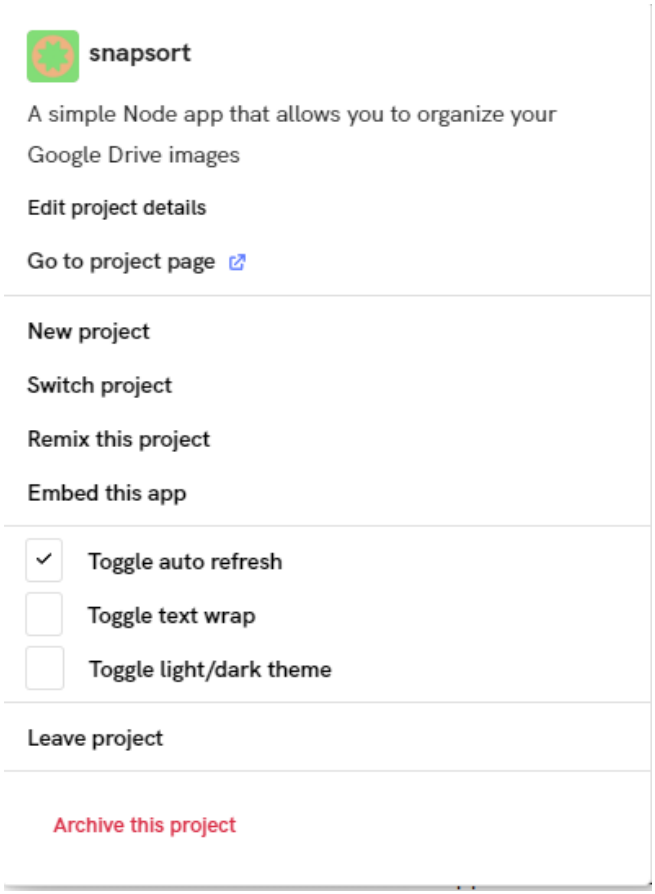

Ilustración 9. Menú de configuración del proyecto

- *Files:* a través de esta herramienta podremos crear ficheros y carpetas dentro del proyecto. Es importante tener en cuenta que debemos utilizar rutas absolutas para la definición de las mismas

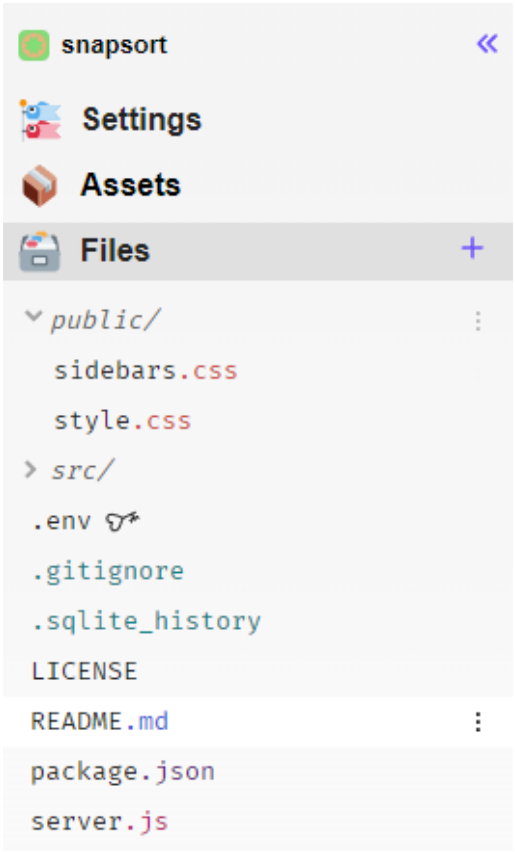

Ilustración 10. Panel lateral izquierdo en la vista de edición

A la derecha del editor se nos muestra una vista previa, como si de un navegador embebido se tratase, de manera que podemos observar la dirección web a la que estamos accediendo dentro del proyecto y la vista que la aplicación devuelve. En la parte inferior de este panel lateral derecho, como se muestra en la *Ilustración 11*, aparecen dos iconos que nos redirigen a una vista con los atajos de teclado y al foro de Glitch.

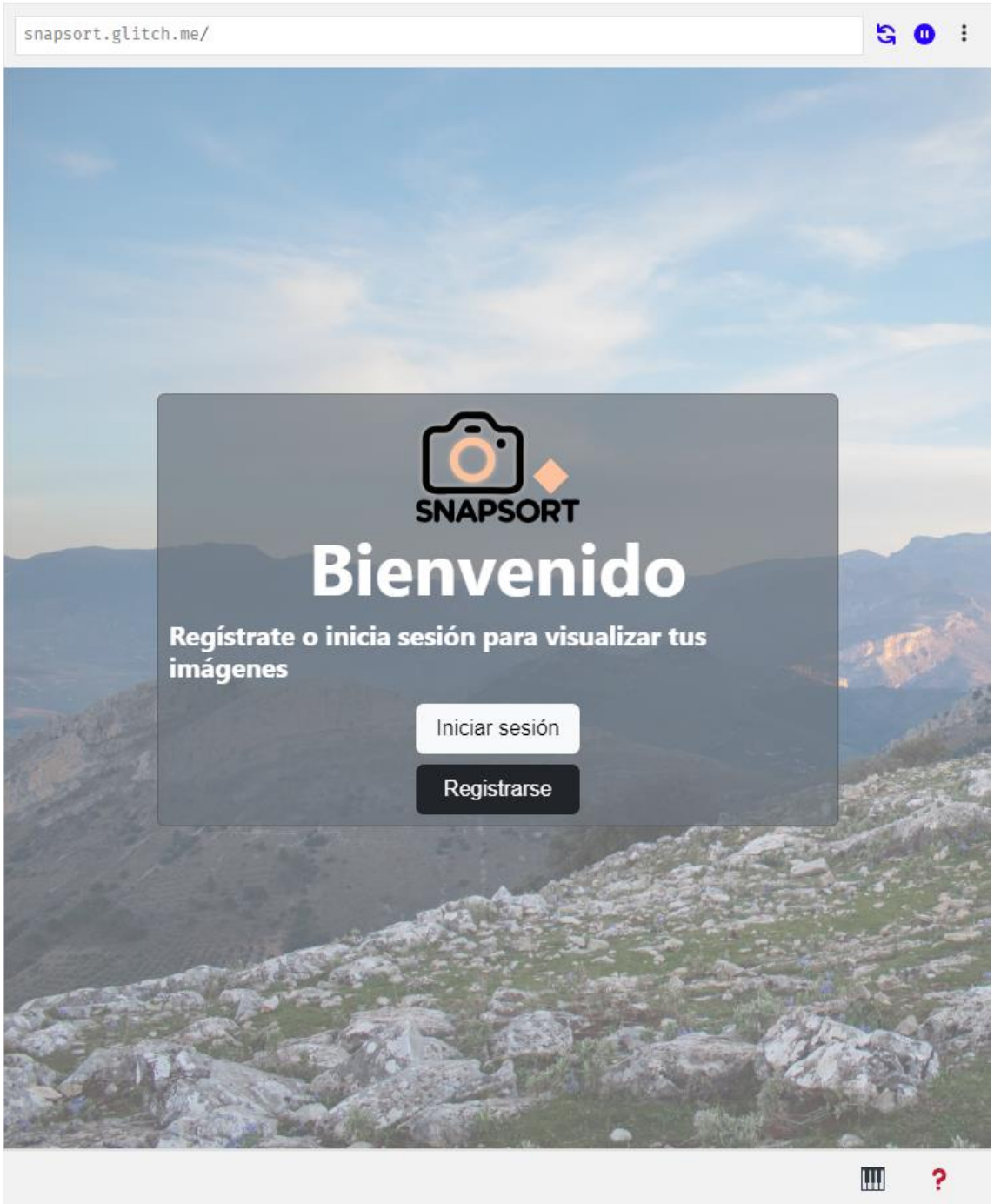

Ilustración 11. Panel lateral derecho con la vista previa. Los iconos que aparecen en la parte inferior derecha corresponden a los accesos directos a los atajos y al foro, respectivamente.

Finalmente, las herramientas posiblemente más interesantes se encuentren en la parte inferior de la página donde se nos da acceso a paneles enfocados principalmente, a la monitorización.

En primer lugar, encontramos el panel *STATUS* que, como muestra la *Ilustración 12*, presenta de manera visual el estado de los recursos que sustentan nuestro proyecto, tales como la memoria o la carga de CPU. Como curiosidad, el emoticono que muestra un rostro sonriendo evoluciona en función del estado de la aplicación, mostrando un emoticono pensativo cuando la aplicación está siendo desplegada y un emoticono con gesto enfadado cuando no ha podido desplegarse por algún problema.

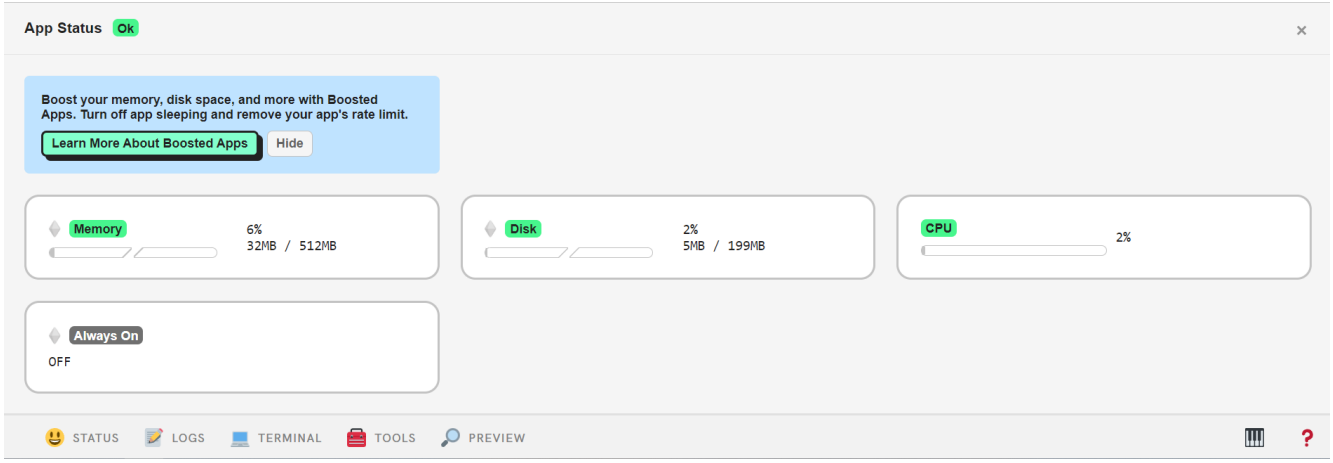

Ilustración 12. Panel STATUS

A la derecha de *STATUS encontramos la herramienta LOGS*, su funcionalidad es muy simple pero muy efectiva: permite que imprimamos la información que está gestionando el servidor a través de la orden de código `console.log()`. (ver *Ilustración 13*)

Ilustración 13. Panel LOGS

La tercera de estas herramientas es el *TERMINAL* (mostrado en la *Ilustración 14*), la cual es la única herramienta que nos permite la comunicación con la máquina que está ejecutando nuestra aplicación. Este terminal será útil a la hora de instalar módulos y de conocer el estado del servicio. Además, puede ser abierta como una pestaña en nuestro navegador.

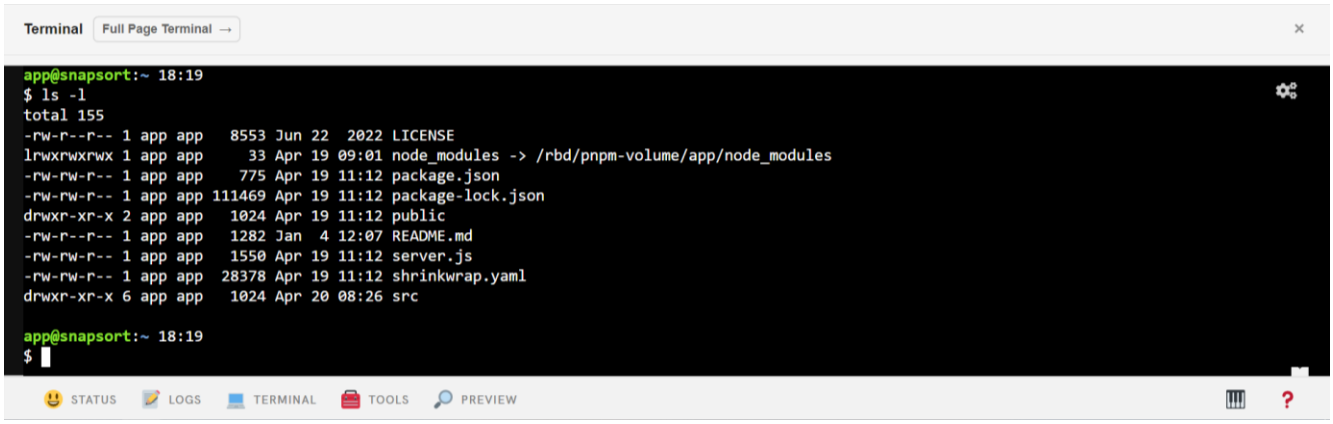

Ilustración 14. Terminal de un proyecto en Glitch

El panel *TOOLS* se representa con una caja de herramientas ya que esconde más de una funcionalidad: en primer lugar, podemos importar código del proyecto desde GitHub o por el contrario exportarlo como un fichero comprimido; además, nos da enlace de acceso al repositorio git del mismo; con la segunda opción de este panel podremos ajustar un nombre de dominio personalizado en caso de disponer de uno; y por último, el panel *TOOLS* permite acceder a la herramienta *REWIND;* esta herramienta nos muestra una línea temporal en la que podremos observar los diferentes *commits* que Glitch ha ido realizando. En caso de pinchar sobre alguno, de ellos observaremos el estado de los ficheros del proyecto en ese momento. La herramienta subraya

con un color rojo las líneas eliminadas y en verde las añadidas como se puede observar en la *Ilustración 15*.

Ilustración 15. Linea de tiempo y cambios en el código

La barra de herramientas finaliza con el panel *PREVIEW*, desde el cual podremos abrir la vista previa en el lateral o en una pestaña nueva en el navegador.

3.3 Lenguajes y tecnologías con las que trabaja

En esta sección examinaremos los lenguajes de programación que son compatibles con la plataforma y que se pueden utilizar para desarrollar aplicaciones en ella. Además, estudiaremos las tecnologías subyacentes que son necesarias para el correcto funcionamiento de la misma.

3.3.1 Lenguajes

Ilustración 16. Logos de los lenguajes con los que trabaja la plataforma

HTML

> *"HTML, siglas en inglés de HyperText Markup Language ('lenguaje de marcado de hipertexto'), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del World Wide Web Consortium (W3C)"* (Wikipedia, s.f.).

HTML5 es la última versión de esta tecnología que añade algunas funcionalidades nuevas como: nuevas etiquetas que mejoran la estructuración de la web, mayor soporte multimedia sin necesidad de software de terceros, validación de formularios y una mejora en la accesibilidad (Gauchat, 2013).

CSS3

> *"Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML […]. CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios."* (Mozilla, s.f.).

CSS3 es la última versión de esta tecnología que ofrece novedades como: nuevas propiedades para el diseño de páginas web, fuentes personalizadas y nuevas formas de seleccionar elementos. (Gauchat, 2013).

JavaScript

"JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente del lado del cliente" (WIkipedia, s.f.)

Aporta dinamismo y funcionalidad a las webs estáticas dotando a los elementos de la web de movimiento, diseños creativos y respuestas a la interacción del usuario o incluso permite la creación de elementos como juegos, animaciones y gráficos en dos y tres dimensiones (Mozilla, s.f.).

3.3.2 Tecnologías

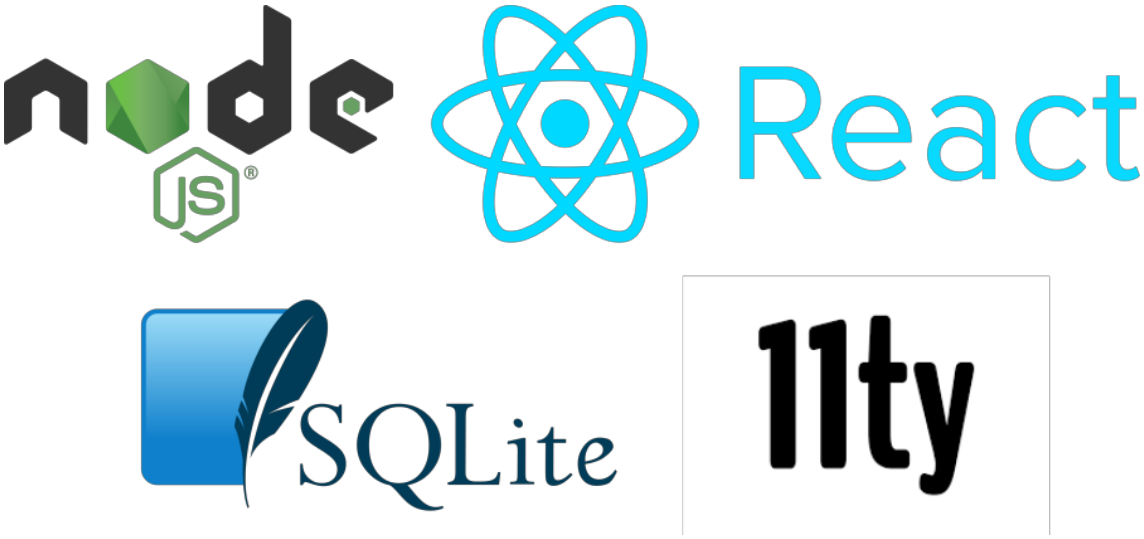

Ilustración 17. Logos de las tecnologías con las que trabaja la plataforma

Node.js

Node.js es un entorno de ejecución de JavaScript que permite correr código JavaScript en el servidor. En lugar de ser un lenguaje de programación en sí mismo, Node.js utiliza JavaScript como su lenguaje de programación y permite a los desarrolladores utilizarlo en el servidor, lo que hace posible el desarrollo de aplicaciones web de alta escalabilidad y rendimiento en tiempo real.

Node.js utiliza un modelo de E/S sin bloqueo y basado en eventos, lo que significa que es capaz de manejar una gran cantidad de solicitudes simultáneas, esta característica lo hace ideal para aplicaciones web de alta demanda. Además, Node.js también proporciona una gran cantidad de módulos y paquetes que los desarrolladores pueden utilizar para simplificar el desarrollo de aplicaciones web.

React

> *"Es una librería open source de JavaScript para desarrollar interfaces de usuario. Fue lanzada en el año 2013 y desarrollada por Facebook, quienes también la mantienen actualmente junto a una comunidad de desarrolladores independientes y compañías."* (Coalla, 2021)

Utiliza un modelo de programación declarativo, es decir, se centra en describir el resultado deseado, en lugar de especificar cómo llegar a ese resultado utiliza un lenguaje llamado JSX parecido XML. Además, se centra en la creación de componentes reutilizables los cuales manejan su propio estado (React, s.f.).

SQLite

SQLite es un sistema de gestión de bases de datos que no necesita de un servidor, lee y escribe directamente del sistema de archivos. Se trata de un SGBD transaccional y de uso libre para cualquier propósito que cuenta con una alta compatibilidad entre diferentes tecnologías.

Originalmente fue escrito en lenguaje de programación C, se trata de una herramienta muy compacta como se demuestra en el espacio en disco que ocupa, aproximadamente 750KiB (SQLite, s.f.). Por defecto, para poder usarla en nuestro proyecto, Glitch añade de forma automática el módulo para Node.js *sqlite3* que permite crear, acceder y gestionar bases de datos SQLite3 utilizando JavaScript. No obstante, existen otras bibliotecas de JavaScript que

podrían utilizarse con el mismo propósito, tales como *sql.js* y *PouchDB,* la cuales también pueden utilizarse dentro de la plataforma Glitch.

Eleventy

Eleventy es un generador de sitios web estáticos, es decir, convierte plantillas de texto que pueden estar en diversos formatos (tales como Markdown, Nunjucks, WebC, Liquid, HTML…) en páginas webs estáticas.

Según los autores (Eleventy, s.f.) no se trata de un *framework* de JavaScript pese a utilizar Node.js, puesto que una vez obtenido el producto final no es necesario mantenerlo dentro del proyecto. Este hecho supone grandes ventajas sobre otros *frameworks* de JavaScript; la principal es que se disminuye el tiempo de carga para los visitantes al tratarse de texto HTML. Está principalmente enfocado a la creación de blogs personales.

Contenedores y Máquinas virtuales

Para un correcto funcionamiento, Glitch utiliza una combinación de máquinas virtuales y contenedores, por lo que procedemos a explicar estas tecnologías.

Un contenedor es una unidad de software liviana y autónoma que encapsula una aplicación y todas sus dependencias necesarias para ejecutarse, incluyendo bibliotecas, herramientas y configuraciones. Además, utiliza la tecnología de virtualización a nivel de sistema operativo para proporcionar aislamiento y portabilidad entre diferentes entornos de ejecución. (Amazon Web Services, s.f.). Pese a que existen distintas implementaciones de esta tecnología, como Linux Containers o CoreOS rkt, la herramienta Glitch utiliza contenedores Docker. (Glitch, s.f.).

Por otro lado, según (Red Hat, 2023) "*Una máquina virtual (VM) es un entorno virtual que funciona como sistema informático virtual con su propia CPU, memoria, interfaz de red y almacenamiento, pero se crea en un sistema de hardware físico*". Esta tecnología permite la ejecución simultanea de diversos sistemas operativos sobre una misma máquina, Glitch utiliza

máquinas virtuales gestionadas por Amazon Web Services o en la Google Cloud Platform. Si acudimos a la consola de nuestro proyecto de Glitch podremos conocer la versión del *kernel* del sistema Linux utilizando la instrucción "`uname`" (ver *Ilustración 18*). Mediante esta comprobación podemos suponer que detrás de nuestro proyecto encontraremos una máquina virtual gestionada a través de *Amazon Web Services*.

Ilustración 18. Versión del kérnel del Contenedor

3.4 Tecnologías alternativas

Como ya se ha indicado, Glitch es una herramienta de desarrollo en la nube, que nos permite desarrollar aplicaciones web de manera colaborativa y en tiempo real. Sin embargo, es importante destacar que existen muchas otras herramientas semejantes que pueden ser útiles para desarrolladores de diferentes niveles y necesidades. En este sentido, explorar diferentes opciones nos permitirá elegir la herramienta que mejor se adapte a nuestras necesidades. Es por ello que, a lo largo de nuestro proceso de aprendizaje, vamos a explorar y analizar diversas herramientas de desarrollo de aplicaciones en la nube para tener una visión amplia y completa de las diferentes opciones disponibles.

3.4.1 AWS Cloud9

AWS (Amazon Web Services) es una plataforma de servicios en la nube que brinda una amplia variedad de recursos informáticos, como almacenamiento, bases de datos, redes y análisis entre otros. Es una de las soluciones que ofrecen la Infraestructura como servicio (IaaS) líderes en el mercado de servicios en la nube (flexera, 2023), y es utilizada por empresas y organizaciones de todos los tamaños para almacenar, procesar y analizar datos de manera escalable y eficiente, sin necesidad de tener infraestructura física propia.

AWS ofrece una gran cantidad de herramientas y servicios que permiten a los desarrolladores crear y desplegar aplicaciones de manera rápida y sencilla. Dentro de los servicios de AWS, encontramos Cloud9, una herramienta de desarrollo en la nube que ofrece un entorno de desarrollo integrado (IDE) completo y flexible, que permite a los desarrolladores trabajar en colaboración y en tiempo real. Con Cloud9, los desarrolladores pueden escribir, ejecutar y depurar código en diferentes lenguajes de programación, como C, C++, PHP, JavaScript, Python, Ruby, y Perl entre otros. Además, esta herramienta se integra con otros servicios de AWS, como EC2 el servicio de alquiler de máquinas virtuales, lo que permite escalar y desplegar aplicaciones de manera fácil y rápida (Amazon Web Services, s.f.).

La idea original surgió en 2010 por Rik Arends, Ruben Daniels y Dennis Hijink. En poco tiempo, Cloud9 se convirtió en una de las herramientas de desarrollo en la nube más populares hasta que en 2016, Amazon adquirió Cloud9 con la intención de integrarla en su plataforma de servicios en la nube, AWS (Wikipedia, s.f.).

En comparación con Glitch, AWS Cloud9 tiene algunas ventajas significativas, como una mayor escalabilidad y cantidad de lenguajes con los que trabajar, además de compatibilidad con las herramientas de AWS. Una de las mayores desventajas de Cloud9 es que requiere conocimientos técnicos para realizar el despliegue, lo que significa que puede no ser la mejor opción para principiantes o para proyectos más pequeños y simples.

3.4.2 Replit

Replit es una plataforma en línea que permite escribir, ejecutar y colaborar en código sin necesidad de instalar software adicional en el equipo local, todo en un mismo sitio web. Proporciona una interfaz de programación y una amplia variedad de lenguajes, como Python, Java, C++ y Ruby (Lardinois, 2018). Los usuarios pueden crear y compartir proyectos, así como colaborar en tiempo real con otros desarrolladores. Replit también proporciona integración con Github y otras herramientas de gestión de código.

Fue 2009 cuando a Amjad Masad le surgió la idea de intentar escribir todos los lenguajes en JavaScript, este reto lo llevó a crear en 2016, junto a su hermano Faris Masad y la diseñadora Haya Odeh, el actual web IDE. Al inicio se encontraba bajo el dominio repl.it que pasó a ser replit.com en marzo de 2021 debido a problemas de limitaciones con el dominio de nivel superior anterior. La herramienta ha ido experimentando varios cambios de editor de código, ha pasado por el uso de Ace y Mónaco, hasta que finalmente implementó CodeMirror (Wikipedia, s.f.).

En comparación con Glitch, Replit ofrece un servicio muy similar que posee una serie de ventajas sobre la herramienta que estamos analizando:

- Replit cuenta con más lenguajes de programación los que programar, pudiendo, por ejemplo, crear una web app en Python con Flask.
- Permiten crear *Unit Test* automatizando las pruebas permitiendo un verdadero flujo DevOps
- El editor de código cuenta con un chat dentro de las herramientas disponibles que facilitan la construcción de código de forma colaborativa
- La versión de pago ofrece más memoria RAM
- El control de versiones se realiza con git, de forma que podremos empezar un proyecto a través de un repositorio online y trabajar desde éste.
- Cuenta con un chat conversacional con una inteligencia artificial entrenada para ayudar a los desarrolladores a generar código

Sin embargo, Glitch ofrece una mayor capacidad de almacenamiento para la versión gratuita y una interfaz más simple (Replit, s.f.).

3.4.3 Codeanywhere

Codeanywhere es un *IDE* en la nube que permite a los programadores escribir, editar y colaborar en proyectos de desarrollo de aplicaciones web. Según la propia web de la plataforma, su editor “soporta todos los lenguajes que soporta Visual Studio Code” (Codeanywhere, s.f.).

Su precursor nació en 2009 bajo el nombre PHPanywhere debido a que solo soportaba dicho lenguaje y así se mantuvo hasta 2013 donde sus fundadores, Ivan Burazin y Vedran Juckić, lanzaron Codeanywhere que tuvo un temprano éxito embolsando $600.000 de World Wide Web Hosting (Wikipedia, s.f.).

Tanto Codeanywhere como Glitch son plataformas en línea que permiten a los desarrolladores trabajar en proyectos de software desde cualquier lugar y dispositivo. Ambas ofrecen una amplia variedad de herramientas y funcionalidades para facilitar la creación colaborativa de aplicaciones en línea. A continuación, se presentan algunas ventajas y desventajas comparativas de ambas plataformas:

- Mientras que Codeanywhere permite trabajar con múltiples lenguajes de programación, Glitch solo permite el uso de JavaScript y lenguajes de marcado
- Codeanywhere ofrece integración con servicios en la nube más populares como AWS, GCP, Azure y DigitalOcean. Además, Codeanywhere permite la conexión con tus propios servidores a través de SSH
- Esta herramienta aporta funcionalidades como seguimiento del cursor, compartir pantalla y un chat en tiempo real lo que hace el trabajo en equipo aún más eficiente.
- No presenta plantillas de proyectos que puedan servir de punto de partida para los usuarios
- Pese a ser un servicio Freemium, el tiempo de prueba gratuito es excesivamente corto, de apenas 7 días, y el plan de suscripción más básico de seis dólares mensuales solo nos ofrece la posibilidad de trabajar en un proyecto. (Codeanywhere, s.f.)

3.4.4 Dockside

Dockside es una herramienta de desarrollo en la nube de código abierto, enfocado a trabajar a través de la ejecución de contenedores que reciben el nombre de Devtainers. Su objetivo es permitir a los desarrolladores trabajar con

copias del entorno de producción de un proyecto principal, evitando así tener que configurar un entorno local.

Utiliza Eclipse Theia online IDE como editor. La herramienta permite trabajo colaborativo gestionando los permisos en función de un esquema de roles y brindando acceso a través de HTTPS.

Dockside.io ofrece la ventaja de permitir diferenciar claramente entre la rama de producción y la rama de desarrollo de un proyecto. Esto es beneficioso para mantener un entorno de prueba separado y asegurar que los cambios y mejoras realizados no afecten directamente a la versión en producción. Sin embargo, es importante tener en cuenta que Dockside.io es exclusivamente un entorno de prueba y no está diseñado para desplegar la aplicación al público general. Esto puede ser considerado como una desventaja, ya que se requiere de otro entorno ya funcionando para poner en marcha el proyecto en producción. Además, el uso de Dockside.io puede resultar más desafiante en comparación con Glitch, ya que implica un mayor nivel de conocimientos técnicos y configuración adicional.

3.4.5 JSFiddle

JSFiddle es una herramienta en línea que permite probar, compartir y colaborar en el desarrollo de código HTML, CSS y JavaScript. Es un entorno de desarrollo integrado basado en CodeMirror que proporciona un editor de código en tiempo real y una vista previa del resultado.

JSFiddle ofrece la posibilidad de incluir bibliotecas y frameworks populares, como jQuery, React y AngularJS, incluso incluye plantillas de proyectos basados en estas tecnologías. Además, cuenta con características útiles como: autocompletado de código, resaltado y verificación de sintaxis, sangrado automático y concordancia de corchetes; todo ello facilita el proceso de escritura y ayuda a evitar errores comunes.

Una característica destacada de JSFiddle es el modo colaboración que permite compartir fácilmente el código a través de enlaces generados por la plataforma. De esta forma, es posible invitar a otros usuarios a trabajar en el

proyecto de forma conjunta, viendo y editando el código en tiempo real. Este modo cuenta incluso con un chat de texto y voz.

Existen muchas compañías que utilizan esta herramienta para mostrar los ejemplos de sus librerías, entre ellas podemos encontrar Mozilla Developer Network, Google Docs, y StackOverflow (JSFiddle, s.f.). Según (Carbonnelle, 2023) es el IDE en línea más buscado.

A diferencia de Glitch, esta herramienta no ofrece servicio de alojamiento en la nube para las páginas web creadas, es decir, aunque podamos utilizarla para crear un prototipo de la web y se nos muestre la vista previa de la misma, no será posible acceder únicamente al producto final.

3.4.6 Codiva.io

Se trata de un entorno de desarrollo online que permite la escritura, compilación y ejecución de ficheros C, C++, Java y Python. El IDE puede iniciarse desde cualquier navegador y almacena el código escrito lo que permite una gran accesibilidad.

Su funcionalidad es muy simple si lo comparamos con sus competidores, no permite la importación de módulos o librerías, y pese a que permite compartir el código, no es posible crear sesiones colaborativas donde más de un usuario modifiquen el proyecto.

La principal desventaja es que no se trata de un servicio de máquinas virtuales ni contenedores que permita mantener la aplicación corriendo en la nube, por lo que no es una herramienta apta para hacer un despliegue de una aplicación web; además, la interfaz de interacción con el usuario está muy descuidada.

3.5 Conclusiones de la comparativa de herramientas

Como se puede observar, existen diferentes ventajas técnicas entre los principales competidores de la plataforma que estamos estudiando. Principalmente, destaca la capacidad de sus competidores para escalar los recursos del sistema en el que corren las aplicaciones. Sin embargo, Glitch es

una plataforma muy accesible para usuarios con pocos conocimientos técnicos. La interfaz es intuitiva y sencilla, lo que facilita que cualquier persona pueda tener una aplicación web funcionando en poco tiempo. Además, Glitch promueve la colaboración y el aprendizaje a través de la comunidad, ofreciendo una experiencia enriquecedora para aquellos que quieren iniciarse en el mundo de la programación y el desarrollo web.

Aunque Glitch soporta de manera oficial menos lenguajes de programación que algunos de sus competidores, es posible utilizar otros lenguajes en la plataforma gracias a su flexibilidad. Además, Glitch cuenta con una comunidad muy activa y colaborativa que comparte proyectos y soluciones utilizando una amplia variedad de lenguajes de programación y que son fácilmente imitables gracias a la funcionalidad *Remix* de la herramienta.

La interfaz de Glitch es altamente personalizable y fácil de usar. Aunque la plataforma ofrece paneles laterales útiles para ayudar en la creación y edición de proyectos, también es posible ocultarlos si se prefiere una interfaz más limpia y enfocada. Además, Glitch cuenta con la opción de modo nocturno, que se puede activar y desactivar con facilidad, de forma que la visibilidad se ve muy mejorada y ayuda reducir la fatiga visual al trabajar en ambientes con poca luz.

4. DESARROLLO

Para poder comprobar de forma exhaustiva la capacidad de desarrollo de la plataforma vamos a utilizarla creando con ella una aplicación web, lo que nos permitirá comprender cómo funciona en el contexto práctico.

Ilustración 19. Código QR que redirige a la aplicación (snapsort.glitch.me)

La *webapp* que vamos a desarrollar se denomina SnapSort (ver *Ilustración 19*) y se trata de una galería virtual que permite clasificar las imágenes que el usuario importe desde servicios en la nube, como Google Drive. La aplicación permitirá listar a través de diferentes filtros tales como temáticas, puntuación, ubicación… El código es accesible a través de la plataforma Glitch mediante la *Ilustración 20*.

Ilustración 20. Código QR que redirige al proyecto (glitch.com/~snapsort)

4.1 DevOps

Para el desarrollo de la aplicación hemos decidido elegir una metodología ágil que nos permita trabajar en un marco DevOps. La elección de una metodología ágil se basa en la necesidad de adaptarnos de manera efectiva a los cambios constantes en los requisitos del proyecto.

Es realmente difícil definir que es DevOps, incluso algunos autores (Freeman, 2019) se atreven a decir que no existe definición para este concepto. Podemos considerar DevOps como una filosofía en la que las personas están por encima de los procesos, y los procesos por encima de las herramientas.

DevOps tiene como objetivo principal la colaboración entre los equipos de desarrollo (Dev) y los equipos de operaciones de sistemas (Ops), convirtiendo así el proceso de desarrollo en un proceso integral, teniendo en cuenta a todos los implicados: desarrolladores, *testers*, personal de operaciones, seguridad e ingenieros de infraestructura. Existen una serie de prácticas asociadas a esta filosofía que ayudan a favorecer la agilidad y la productividad (Microsoft, s.f.).

- Integración continua y entrega Continua (CI/CD): CI es la práctica consistente en integrar cambios en el repositorio de código múltiples veces al día, este código debe ser testeado en busca de errores de forma automática. Por su parte CD es el proceso automatizado que permite que el código testeado sea trasladado a producción.
- Control de versiones: utilizar un sistema de control de versiones como Git facilita la asignación de tareas y la colaboración entre los miembros del equipo, además de la revisión y la recuperación del código.
- Desarrollo ágil de software: el uso de una metodología de desarrollo ágil permite que los equipos se adapten con mayor facilidad a los cambios de requisitos que surjan, además favorecen la colaboración y comunicación entre los miembros del proyecto.
- Infraestructura como código (IaC): permite la gestión de la infraestructura de un proyecto a través de código permitiendo la automatización de procesos y la estandarización de la configuración.
- Supervisión continua: es tan importante la monitorización de los recursos del sistema como la representación de la misma a través de un sistema visual que permita mostrar de forma gráfica los datos recogidos; esto permite que los equipos puedan lidiar con los problemas de funcionamiento en tiempo real y eliminarlos en próximas iteraciones.
- Test automáticos: Las pruebas automáticas también forman parte integral de la filosofía DevOps. Estas pruebas se realizan de forma automatizada y continua a lo largo del proceso de desarrollo y entrega. Permiten detectar rápidamente posibles errores o fallas en el código, asegurando la calidad del producto final. Al incorporarlas en todas las etapas del desarrollo, desde la integración continua hasta la entrega continua, se garantiza la detección temprana de problemas y se agiliza la resolución de los mismos. Esto contribuye a la reducción de errores en producción y a la mejora de la confiabilidad y estabilidad del software desarrollado

A modo resumen podemos decir que el ciclo de vida de una aplicación desarrollada bajo una filosofía DevOps debe pasar por las siguientes fases de forma cíclica, consiguiendo así la retroalimentación necesaria:

- Planificación: esa fase implica la definición de requisitos y la asignación de las tareas necesarias para desarrollar y entregar el software
- Codificación y compilación: en esta fase los programadores trabajan en la implementación del software bajo alguna metodología ágil y herramientas de colaboración.
- Pruebas: se prueba el software desarrollado de forma automatizada en busca de errores garantizando una alta calidad.
- Despliegue: el software se traslada a producción.
- Operaciones: el software se monitorea y se mantienen en producción, utilizando herramientas de monitoreo y análisis para detectar y solucionar problemas.

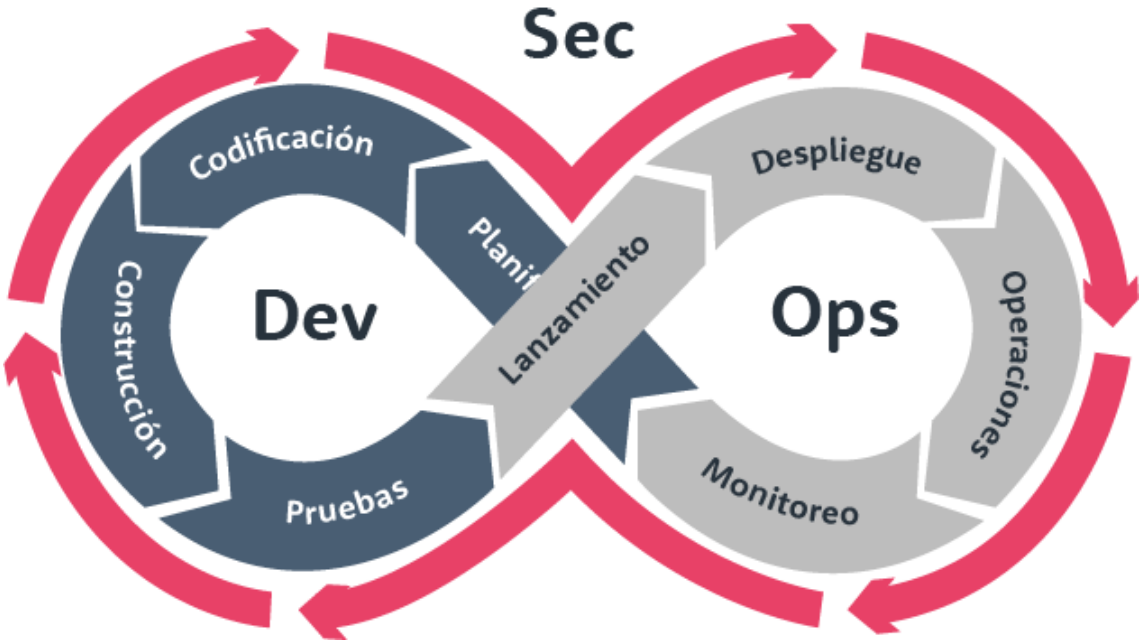

Ilustración 21. Ciclo de vida de una aplicación DevOps

4.2 Metodología Ágil

Nos acercaremos a la definición de metodología ágil a través de la siguiente explicación:

> *"La metodología ágil no hace referencia a una serie de indicaciones sobre qué hacer exactamente durante el desarrollo de software. Se trata más bien de una forma de pensar en la colaboración y los flujos de trabajo, y define un conjunto de valores que guían nuestras decisiones con respecto a lo que hacemos y a la manera en que lo hacemos."* (Red Hat, 2022)

Estas metodologías surgieron como una respuesta a las limitaciones de los enfoques tradicionales de gestión de proyectos, que se basan en planes detallados y rigurosos, y que no siempre son adecuados para proyectos complejos y cambiantes. Por ello surgió el Manifiesto Ágil en el cual se exponen los 4 valores clave "Individuos e interacciones sobre procesos y herramientas, software funcionando sobre documentación extensiva, colaboración con el cliente sobre negociación contractual, respuesta ante el cambio sobre seguir un plan" (Agile manifiesto, s.f.)

Específicamente, las metodologías ágiles de desarrollo de software tienen como objetivo entregar de manera rápida pequeñas funcionalidades en funcionamiento con el fin de incrementar la satisfacción del cliente.

El 16th State of Agile Report (State Of Agile, 2022) es un informe que recopila datos y estadísticas sobre el uso y la adopción de metodologías ágiles en todo el mundo. Este informe es elaborado por una comunidad global de profesionales de Agile, y se basa en una encuesta que se realiza cada año para recopilar información de miles de organizaciones y equipos ágiles en diferentes sectores y países.

En base a este informe, vamos a realizar un análisis detallado de las principales metodologías utilizadas en la actualidad. Este informe proporciona

información valiosa sobre cómo las organizaciones están implementando y beneficiándose de prácticas ágiles, lo que nos permitirá profundizar en las metodologías más populares y sus características clave. A través de este análisis, podremos obtener una mejor comprensión de cómo se están utilizando en diferentes entornos y cómo pueden aplicarse en una variedad de proyectos.

4.2.1 Metodología Scrum

Scrum recoge una serie de prácticas, principios y valores que ayudar a los equipos de desarrolladores a mantener el trabajo estructurado durante todo el proceso de entrega del prodcuto. Existen dentro de scrum tres tipos de agentes: El *scrum master* será el encargado de las diferentes reuniones y procurará que el equipo trabaje de manera eficiente, además lo aislará de las posibles interferencias que puedan surgir. El *product owner* conecta al equipo de desarrollo con el cliente y su principal misión es la de entregar a los clientes finales el mejor producto posible. Por último, el equipo de desarrollo que será el encargado de diseñar, crear, probar y entregar el producto.

Scrum utiliza un ciclo de desarrollo iterativo e incremental que se basa en diferentes iteraciones o *sprints.* Cada una de estas iteraciones conlleva una reunión que puede tener una duración y frecuencias variadas en función del objetivo de la misma, no existe un número fijo de reuniones, sino que depende del proyecto y las necesidades del equipo. Lo importante es que cada sprint tenga una duración coherente y sea lo suficientemente corto para permitir la entrega de valor en un plazo razonable.

4.2.2 Kanban

Kanban es un marco de trabajo visual utilizado en la gestión de proyectos y procesos para mejorar la eficiencia y la productividad. Su objetivo principal es optimizar el flujo de trabajo, minimizando el tiempo de espera y los cuellos de botella, y maximizando el valor entregado al cliente.

Se basa principalmente en la visualización del trabajo en un tablero Kanban, donde las tareas se representan en tarjetas o post-it que se mueven a través de diferentes etapas de un proceso y representan las tareas del proyecto. Existen 3 etapas distintas: tareas por hacer, tareas en progreso y

tareas realizadas de forma que los miembros del equipo pueden ver el estado actual de cada tarea y el progreso general del proyecto (Planview, s.f.).

Kanban implementa la filosofía Lean y por ello se centra en la mejora continua del proceso y en la eliminación de los desperdicios, como la sobreproducción, los tiempos de espera y los movimientos innecesarios. Además limita el *WIP* o *work in progress* estableciendo el número máximo de tareas en progreso que encontramos en el proyecto, esto ayuda a eliminar la sobrecarga del equipo y a mantener un flujo de trabajo constante (Boto, 2020).

4.2.3 Extreme Programming

Es un enfoque de desarrollo de software que se centra en la entrega del software que se necesita cuando se necesita. XP se enfoca en 5 valores muy comunes en el desarrollo de software ágil, como la comunicación frecuente, la retroalimentación temprana, la simplicidad, el respeto y la valentía (Extreme Programming: A gentle introduction, s.f.).

La metodología Extreme Programming fue creada por Kent Beck en la década de 1990 y se ha utilizado en una amplia variedad de proyectos de desarrollo de software en todo el mundo. La metodología ha sido descrita en detalle en varios libros y artículos, incluyendo el libro "Extreme Programming Explained: Embrace Change" de Kent Beck, que es considerado uno de los principales recursos para entender la metodología (Wikipedia, s.f.).

Algunas de las prácticas de desarrollo más comunes de XP son: desarrollo guiado por pruebas o TDD, programación en parejas, juego de planificación, integración continua, uso historias de usuario y diseño simple entre otras. Estas prácticas se combinan con valores y principios ágiles para formar un enfoque coherente y efectivo para el desarrollo de software (Agile Alliance, s.f.).

4.2.4 Metodología seleccionada

Una vez estudiadas las principales metodologías ágiles y sus prácticas, debemos elegir aquella que utilizaremos en la realización de nuestro proyecto.

No podemos adaptarnos a la metodología de Scrum puesto que no contamos con un equipo de desarrolladores en el que fomentar la comunicación y colaboración, tampoco contamos con un cliente con el que ponernos de acuerdo.

Sin embargo, una de las grandes ventajas de las metodologías ágiles es que comparten una serie de valores, principios y prácticas que las hacen altamente interoperables entre sí, esto significa que los equipos de desarrollo pueden adoptar prácticas de diferentes metodologías y combinarlas en un enfoque personalizado que se adapte a sus necesidades específicas y al contexto del proyecto. Por ello en nuestro proyecto utilizaremos prácticas variadas de las metodologías estudiadas.

4.3 Planificación

Para describir la funcionalidad de la aplicación que vamos a desarrollar, hemos decidido utilizar otra técnica propia de las metodologías ágiles: las historias de usuario. Esta técnica nos permitirá representar las necesidades del usuario final de una manera clara y concisa, utilizando un lenguaje fácil de entender. La relación de HUs consideradas se muestra en la *Tabla 2*.

Para evitar sobrecargar el flujo de trabajo, es fundamental establecer prioridades entre las diversas historias de usuario existentes. Una manera de lograrlo es mediante la aplicación del método MoSCoW.

MoSCoW es una técnica de priorización de tareas desarrollada por Dai Clegg en 1994 que clasifica las funcionalidades de una aplicación en cuatro categorías (Hermitte, 2021):

- **M de *Must have*:** es la categoría con mayor prioridad, representa a las tareas que son indispensables e innegociables. En caso de no terminar estas podemos considerar que nos encontramos ante un producto no funcional.
- **S de *Should have:*** agrupa a las categorías importantes que deben llevarse a cabo una vez finalizadas las imprescindibles. Aportan valor añadido al proyecto y permiten lograr objetivos.

- **C de *Could have*:** estas historias de usuario se llevarán a cabo siempre que sea posible, su ejecución no puede afectar de forma negativa al flujo de trabajo del proyecto.
- **W de Won't have:** representan a las tareas que por el momento son secundarias y no pondremos en marcha.

	Como	Quiero	Para	Categoría MoSCoW
1	Usuario no identificado	Identificarme dentro de la aplicación	Que mi información sea personal y privada	**M**
2	Usuario no identificado	Iniciar sesión con servicios de terceros	Evitar registrarme en más sitios web	**C**
3	Usuario	Añadir etiquetas de puntuación a las imágenes	Poder categorizarlas	**S**
4	Usuario	Etiquetar a mis contactos	Poder localizar fotos en los que aparezcan	**W**
5	Usuario	Añadir etiquetas de tema a las imágenes	Poder categorizarlas	**S**
6	Usuario	Filtrar las fotos por las diferentes categorías	Poder encontrarlas con mayor facilidad	**C**
7	Usuario	Importar imágenes y visualizarlas como un mosaico	Ver la mayor cantidad posible sin perder detalle	**M**
8	Usuario	Que las imágenes se auto cataloguen	Ahorrar tiempo	**W**
9	Usuario	Añadir etiquetas de localización a las imágenes	Poder categorizarlas	**S**
10	Usuario no identificado	Que se me muestre una pantalla de Inicio	Conocer las funcionalidades de la aplicación	**M**

Tabla 2. Historias de usuario y clasificación MoSCow

4.3.1 Planificación temporal

Se presenta a continuación la planificación a priori que estima la duración de cada uno de los eventos del presente Trabajo Fin de Grado (ver *Tabla 3*). Es importante tener en cuenta que solamente se contarán con los días lectivos y con una media de cinco horas diarias.

Tareas	Fechas		Duración	
	Fecha inicio	Fecha fin	Días	Horas (5h/día)
Introducción y objetivos	30-ene	06-feb	6	30
Tecnologías Alternativas	07-feb	22-feb	12	60
Tipos de proyecto que permite y lenguajes	23-feb	01-mar	4	20
Qué es Glitch e historia	02-mar	17-mar	12	60
Desarrollo	20-mar	19-abr	22	110
Interfaz de la herramienta	20-abr	25-abr	3	15
Conclusiones y conceptos previos	26-abr	03-may	5	25
Anexos y últimos retoques	04-may	16-may	8	40
		Total	**72**	**360**

Tabla 3. Planificación temporal

Estimación final de tiempo			
Días	Horas	Semanas	Meses
72	**360**	**14**	**3,6**

Tabla 4. Tabla final de la estimación temporal

Una vez completada la estimación del tiempo requerido para el desarrollo del proyecto, procederemos a utilizar un diagrama de Gantt con el fin de visualizar de manera clara y organizada la planificación de actividades. Como se observa en la *Ilustración 22*, el diagrama de Gantt nos permite identificar las tareas y su duración brindándonos una visión general del cronograma del proyecto.

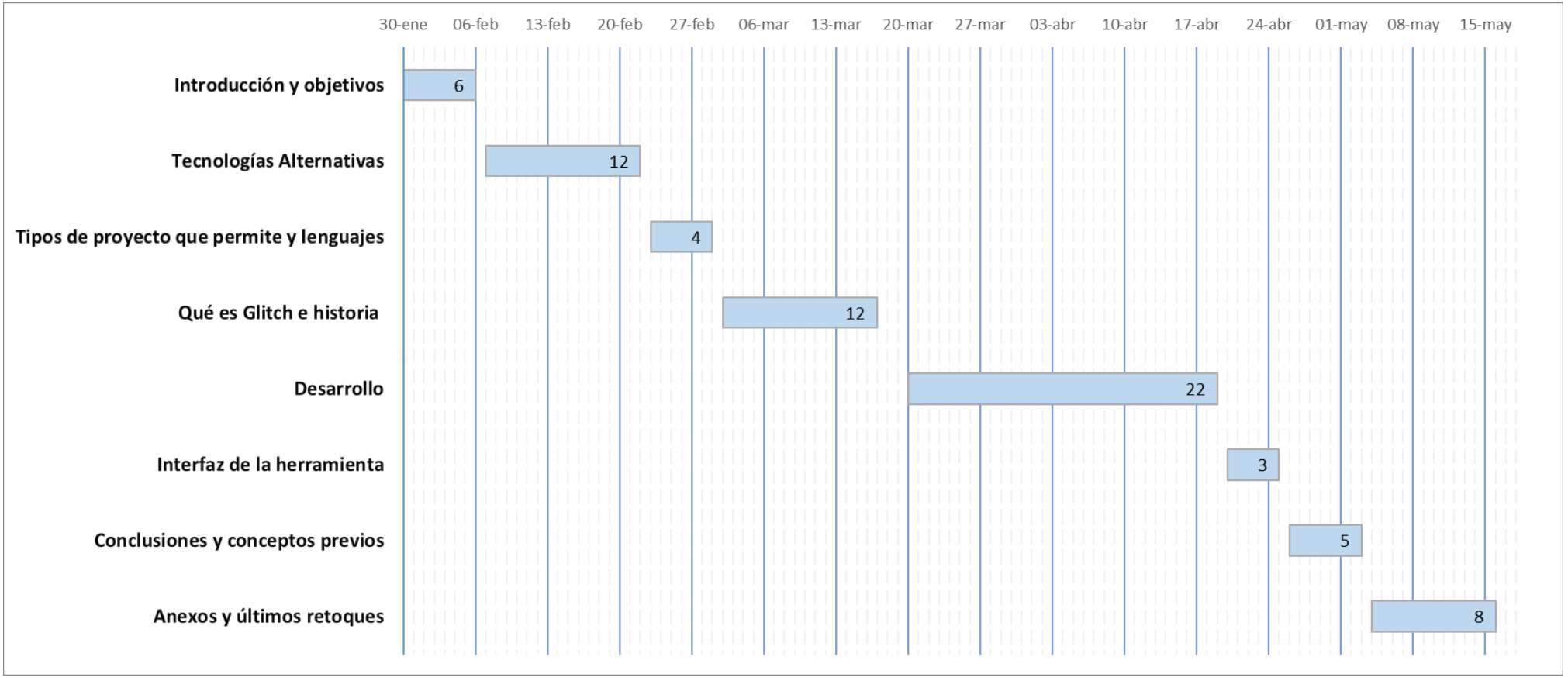

Ilustración 22. Diagrama de Gantt de la planificación del Trabajo Fin de Grado

4.3.2 Estudio de costes

Como hemos podido observar en la *Tabla 4*, la duración aproximada del proyecto es de 14 semanas. Estimaremos los gastos sabiendo que este proyecto contará únicamente con programador y que la vida útil del hardware será de 5 años.

Según (Jobted, s.f.) el salario neto medio de un programador es de 1610€/mes, suponiendo que un mes consta de 4 semanas y que de cada semana únicamente 5 días son laborables, obtenemos un sueldo de 80,5€/día, por lo que a la hora cobrará 16,1€/h.

En la *Tabla 5* podremos observar el coste estimado del proyecto, desglosado por categorías.

Recurso		Cálculo	Coste
Gasto personal	Programador	12,88€/hora x 360 horas	4.636,80 €
Gasto Hardware	MSI PS42 8RA-280XES	1198€ x $\frac{4\ meses}{60\ meses\ útiles}$	79,87 €
	Monitor AOC 24B1XH 23.8" LED IPS	99,99€ x $\frac{4\ meses}{60\ meses\ útiles}$	6,67 €
	Logitech G305 Ratón Inalámbrico	39,98€ x $\frac{4\ meses}{60\ meses\ útiles}$	2,67 €
	Teclado Inalambrico Mecanico Dierya DK63	49,00€ x $\frac{4\ meses}{60\ meses\ útiles}$	3,27 €
Gasto software	Glitch Pro	7,42 €/mes x 4 meses	29,68 €
	Jira	0,00 €	0,00 €
	Visual Paradigm 17.0	Licencia estudiante	0,00 €
	Microsoft Office 2019 Professional Plus	Licencia estudiante	0,00 €
Gasto Inmaterial	Internet Fibra óptica Orange 1GB	40,95 €/mes x 4 meses	199,80 €
Total			**4.958,74 €**

Tabla 5. Tabla de costes del proyecto

4.4 Herramientas y librerías adicionales

Es necesario destacar, que para el desarrollo de la aplicación nos hemos apoyado en diferentes bibliotecas y *frameworks* que facilitan algunas de las tareas de codificación, para mantenernos fieles a los objetivos de este proyecto hemos utilizado aquellas que se proponen en los ejemplos de la plataforma estudiar. De la misma forma ha sido necesario apoyarnos en otras herramientas software para llevar a cabo las prácticas propias de las metodologías ágiles utilizadas

Procedemos a realizar una breve explicación de cada una de ellas.

4.4.1 Librerías

Bootstrap

Se trata de un marco de trabajo *front-end* que permite desarrollar interfaces de usuario combinando estilos de CSS, Javascript y elementos HTML en las aplicaciones web.

Permite además crear diseños responsivos que se adapten a diferentes tamaños de pantalla. Su principal ventaja es que cuenta con una amplia biblioteca de componentes; además, no requiere instalación, se importa fácilmente al proyecto a través de la cabecera HTML y gracias a los enlaces CDN.

Fastify

Fastify es un *framework* web de Node.js diseñado específicamente para construir aplicaciones web rápidas y eficientes. Creado por Matteo Collina y Tomas Della Vedova (Fastify, s.f.), fue desarrollado para ser un marco de trabajo web que destacase por su velocidad en el manejo de solicitudes HTTP, lo que lo hace ideal para proyectos de cualquier tamaño y complejidad.

Una de las principales características de Fastify es su enfoque modular y extensible para el desarrollo de aplicaciones web. Cuenta con cincuenta y tres módulos propios y doscientos diez *plugins* de la comunidad (Fastify, s.f.) lo que

permite a los desarrolladores elegir los módulos que se adapten a las necesidades de su proyecto en particular.

Fastify/view y Handlebars

Fastify/view es un complemento para Fastify que permite el uso de motores de plantillas para la generación de vistas en aplicaciones web. Es decir, en lugar de devolver directamente una respuesta HTML desde el controlador de ruta, se utiliza un motor de plantillas para procesar la respuesta y generar una vista dinámica que se entregará al cliente.

El motor de plantillas que usaremos para generar vistas dinámicas es Handlebars ya que es de código abierto. Permite la creación de plantillas reutilizables y la incorporación de variables y lógica de programación en el proceso de generación de la vista. Por ejemplo, permite el uso de expresiones condicionales y bucles, así como la inclusión de plantillas parciales para simplificar la reutilización de código.

Fastify/session y Fastify/cookie

Una de las principales necesidades de una aplicación web es la autenticación de los usuarios en el servidor, con estos dos complementos de Fastify logramos obtener soporte para sesiones y cookies logrando así la identificación de los usuarios.

Con Fastify/session podremos almacenar en el servidor de forma temporal la información de la sesión del usuario y sus preferencias, además es altamente configurable permitiendo definir desde el tiempo de vida de la sesión como la ubicación.

A través del uso de Fastify/cookie podremos generar una cookie para el usuario. Las cookies son archivos de tamaño reducido que se guardan en el navegador del cliente y recogen la información de este.

Utilizando de forma combinada estas dos bibliotecas logramos identificar qué usuario realiza las peticiones a nuestro servidor.

Fastify/formbody

Con este módulo de Fastify podremos transformar los datos de los formularios que el usuario nos envíe a través de una solicitud HTTP, es decir, esta biblioteca se encarga de manejar la solicitud HTTP, extraer los datos, analizarlos y convertirlos en objetos propios de Javascript y añadirlos al cuerpo de la petición. Este proceso lo realiza de forma transparente para el servidor, independientemente de si son enviados por los métodos GET o POST.

Sqlite3

Sqlite3 es una biblioteca de software que implementa un motor de bases de datos SQLite completo. Destaca por su facilidad de uso, ya que se integra con facilidad con proyectos Node.js, permitiendo incluso creación de la base de datos a través de código Javascript.

Será necesario para dotar de información persistente a nuestra aplicación, además de permitir los procesos básicos para identificación de usuarios.

Bcrypt

BCrypt es un módulo que nos permitirá cifrar y descifrar la información sensible de nuestra aplicación con un algoritmo de *hashing* conocido como Blowfish (Sourceforge, 2002). Utiliza una técnica llamada *salting* que agrega una capa adicional de seguridad añadiendo una cadena de caracteres aleatoria a una contraseña antes de cifrarla lo complica la tarea de descifrado a través de un ataque de fuerza bruta.

Jasmine y Supertest

Jasmine es una biblioteca de pruebas unitarias para JavaScript que se utiliza para escribir pruebas automatizadas de comportamiento en proyectos de desarrollo web. Esta biblioteca permite a los desarrolladores definir y ejecutar casos de prueba para sus aplicaciones, lo que les ayuda a identificar errores y fallos en su código.

La librería Supertest permite simular solicitudes HTTP a un servidor y verificar las respuestas obtenidas a través de: los códigos de estado, los encabezados y el contenido de la respuesta. En nuestro caso la integraremos con Jasmine lo que permitirá comprobar los puntos finales de la aplicación web.

4.4.2 Herramientas

Jira

Jira es una herramienta de gestión de proyectos y seguimiento de incidencias, que permite a los equipos de desarrollo trabajar de manera ágil y colaborativa. En particular, Jira es útil para aplicar metodologías ágiles como Scrum o Kanban, ya que permite crear y organizar tareas, establecer prioridades, asignar responsabilidades y hacer seguimiento del progreso.

En nuestro caso la hemos utilizado para crear un tablero Kanban, donde se pueden crear listas de tareas, columnas de estados y etiquetas que permiten clasificar las tareas de acuerdo con las necesidades del proyecto, garantizando así la gestión y control del proyecto de manera efectiva.

Visual Paradigm

Visual Paradigm es una herramienta de modelado y diseño de software que permite crear una gran variedad de diagramas y modelos, incluyendo los diagramas de secuencia. Con esta herramienta, es posible crear modelos visuales que representan el comportamiento del software en diferentes situaciones, tales como cuando el software recibe un evento o una entrada de usuario.

En nuestro proyecto, hemos utilizado Visual Paradigm para crear los diagramas de secuencia que nos han ayudado a diseñar la interacción entre los diferentes componentes de nuestra aplicación.

4.5 Desarrollo por incrementos

En el presente capítulo procedemos con el desarrollo de una aplicación mediante el uso de la herramienta Glitch, este proceso estará divido en tres fases diferentes y dentro de cada una de ellas se decidirá qué historias de usuario a abordar, se diseñarán los elementos necesarios para su funcionamiento y se llevará a cabo su implementación.

4.5.1 Primer sprint

Historias de usuario

Para esta primera versión de la aplicación escogeremos dos funcionalidades de la categoría *must*. Empezaremos por definir la pantalla de inicio para los usuarios que no se encuentren aún identificados, y las pantallas y servicios que permiten a los visitantes iniciar sesión y registrarse en la plataforma.

Como se ha mencionado anteriormente, trabajaremos con un tablero Kanban que nos ayude a limitar el flujo de trabajo. En la *Ilustración 23* se pude observar el tablero y las dos historias de usuario seleccionadas para la primera versión: Pantalla de inicio para los usuarios no identificados y el sistema de identificación de usuarios.

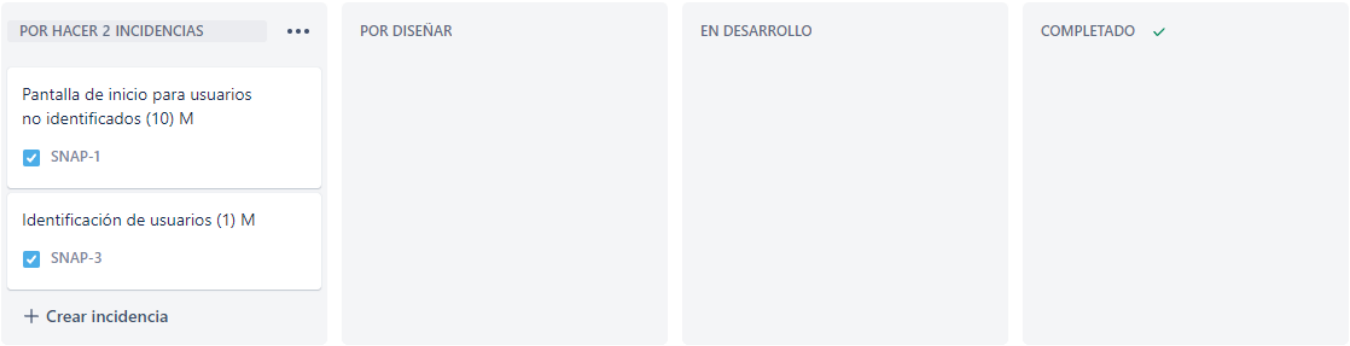

Ilustración 23. Tablero Kanban al inicio del primer sprint

Diseño de la base de datos

Previamente a diseñar las diferentes vistas de la aplicación, debemos definir el modelo de datos necesario para el funcionamiento de la misma. Hemos optado por una base de datos relacional ya que nos permitirá mantener estructurada la información y garantizará la escalabilidad; por lo que diseñaremos un modelo entidad-relación (ver *Ilustración 24*) que nos permitirá visualizar de manera clara las entidades que van a estar presentes en nuestra aplicación, así como sus relaciones y atributos. Este proceso de modelado es fundamental para poder diseñar una aplicación escalable, ya que es muy probable que en próximas versiones debamos modificarla para añadir nuevas funcionalidades y este diagrama nos ayudará a mantener la información estructurada.

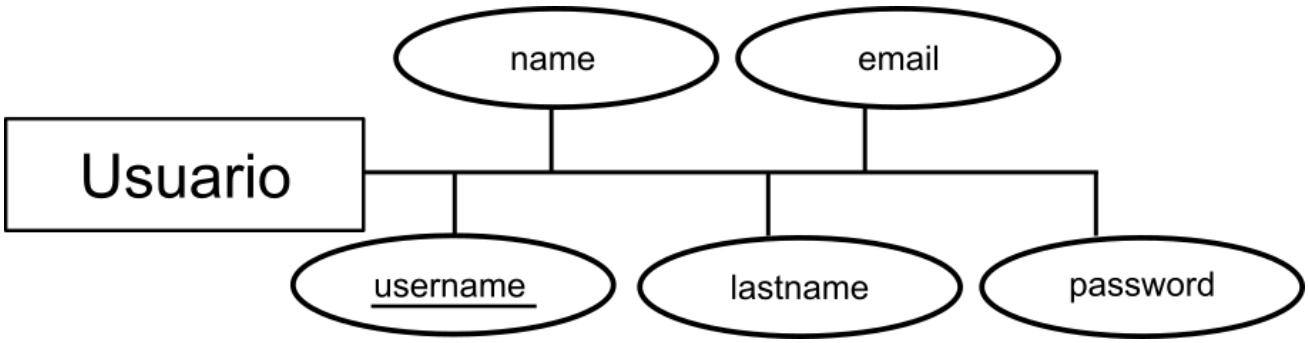

Ilustración 24. Esquema Entidad-Relación de la base de datos en el primer sprint

En esta primera versión contaremos únicamente con una entidad usuario que represente al cliente que accede a nuestra aplicación y cuenta con los siguientes atributos:

- Username: será la clave primaria de la entidad y representa el identificador único para cada usuario registrado en el sistema.
- Name: esta columna almacenará el nombre del usuario.
- Lastname: en esta columna se guardarán los apellidos del usuario.
- Email: esta columna almacenará el correo electrónico del usuario.
- Password: esta columna contendrá la contraseña de acceso al sistema, la cual será cifrada para garantizar la seguridad de la información.

Diseño de las vistas

Diseñamos por el momento tres vistas distintas para esta versión: una pantalla de inicio, una pantalla de inicio de sesión y una pantalla de registro. Empezaremos definiéndolas a través de *sketchs.*

Empezaremos por definir la vista de la página principal, como se puede observar en la *Ilustración 25*, esta vista contiene la información básica acerca de la herramienta: el nombre y una pequeña descripción, además de dos botones que redirigen a las pantallas de inicio de sesión y registro. Esta vista tendrá una imagen de fondo con una opacidad reducida que aportará atractivo a posibles visitantes.

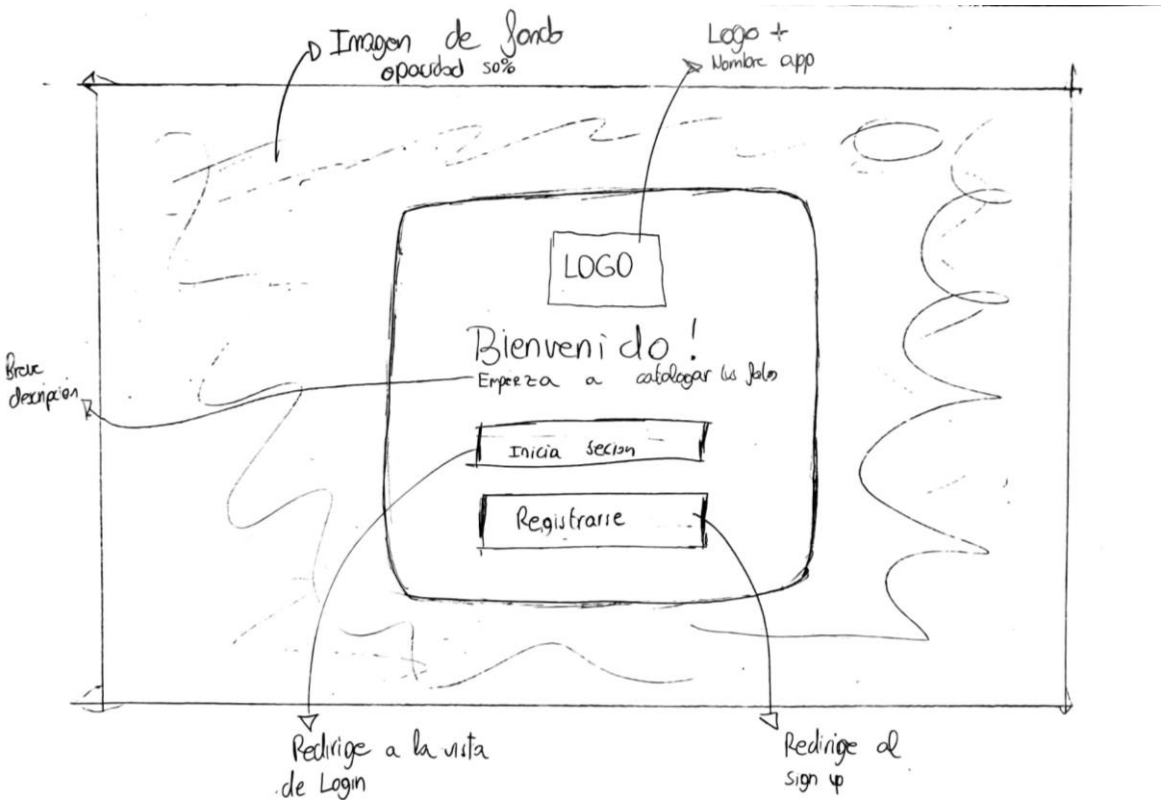

Ilustración 25. Sketch de la vista de inicio

Tras eso definiremos las vistas de las pantallas de inicio de sesión y de registro, *Ilustración 26* e *Ilustración 27*; estas dos vistas tendrán un contenido semejante, el elemento principal de las mismas serán los campos rellenables y mantendremos el mismo fondo que la página principal para aportar cohesión. Además, al igual que con la página principal, dentro de cada una de las vistas tendremos la posibilidad de navegar hacia otras secciones del sistema.

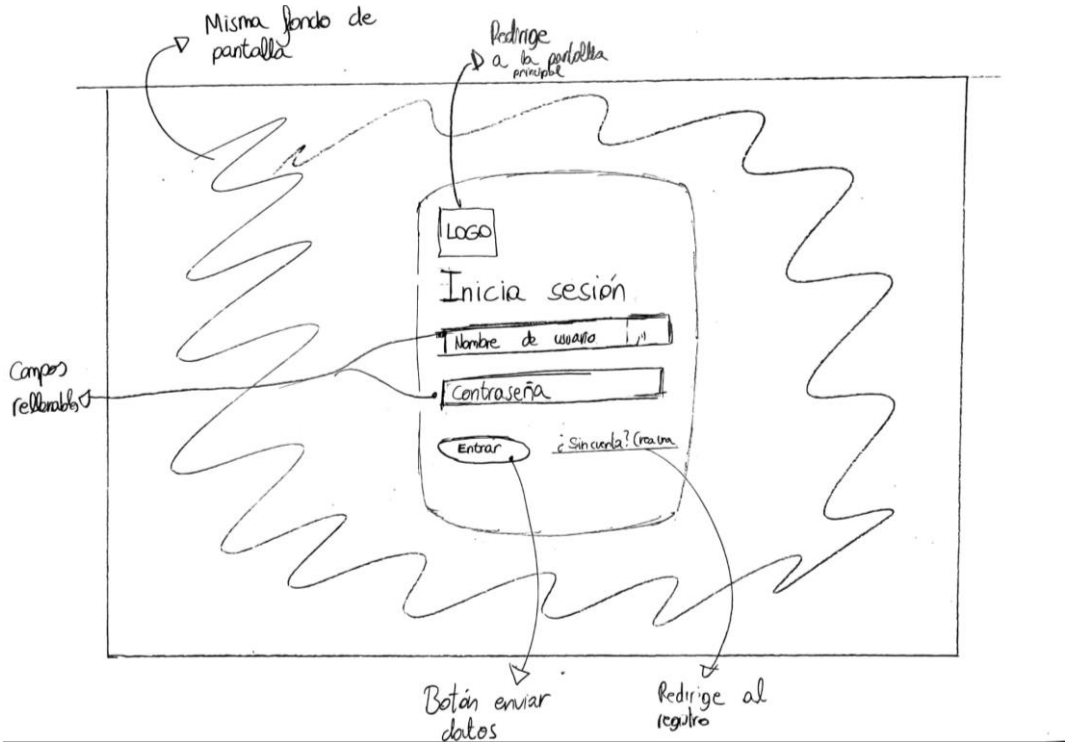

Ilustración 26. Sketch de la vista login

Como suele ser habitual, la vista de registro, *Ilustración 27*, contiene un mayor número de campos rellenables con la información del usuario; por este motivo cuando comencemos con la codificación deberemos adaptar el ancho el alto de los cuadros de contenido.

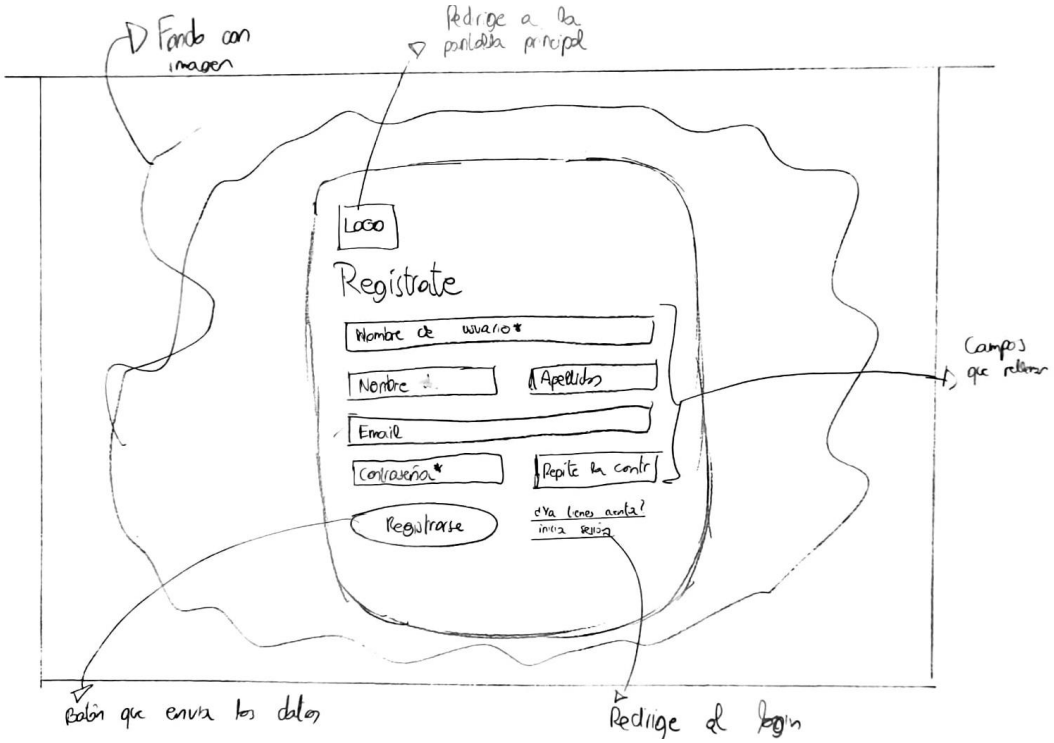

Ilustración 27. Sketch de la vista de registro

Diagrama de Secuencia

Para ilustrar la interacción entre los distintos objetos y componentes que forman parte del sistema procedemos a modelar los principales casos de uso de la aplicación con dos diagramas de secuencia. Como se puede comprobar en la *Ilustración 28* y la *Ilustración 29*, las entidades principales son el servidor y la base de datos que se encargan del tratamiento y la validación de los datos que el usuario ofrece.

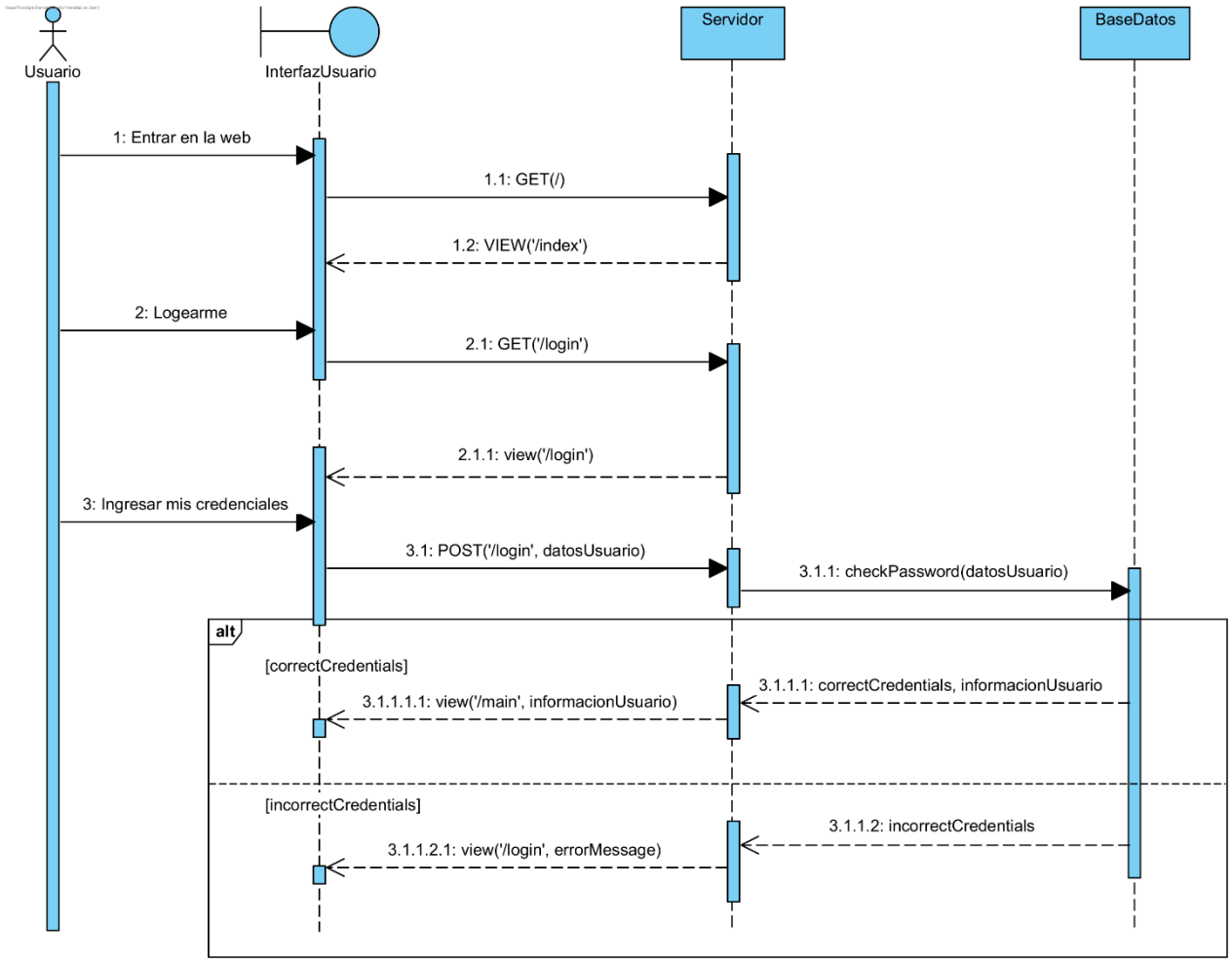

Ilustración 28. Diagrama de secuencia del proceso de inicio de sesión

Durante el proceso de creación de una nueva cuenta deben de validarse los datos del usuario, lo cual explica el doble uso del operado ALT en la *Ilustración 29*. En primer lugar, comprobaremos que el nombre de usuario que debe de ser un valor único en la aplicación que no esté siendo utilizado por otro usuario; en segundo lugar, comprobaremos que la contraseña cumple con un

mínimo de caracteres. Solo en caso de cumplir esta doble condición un usuario que esté creando cuenta en la plataforma podrá acceder a la vista principal.

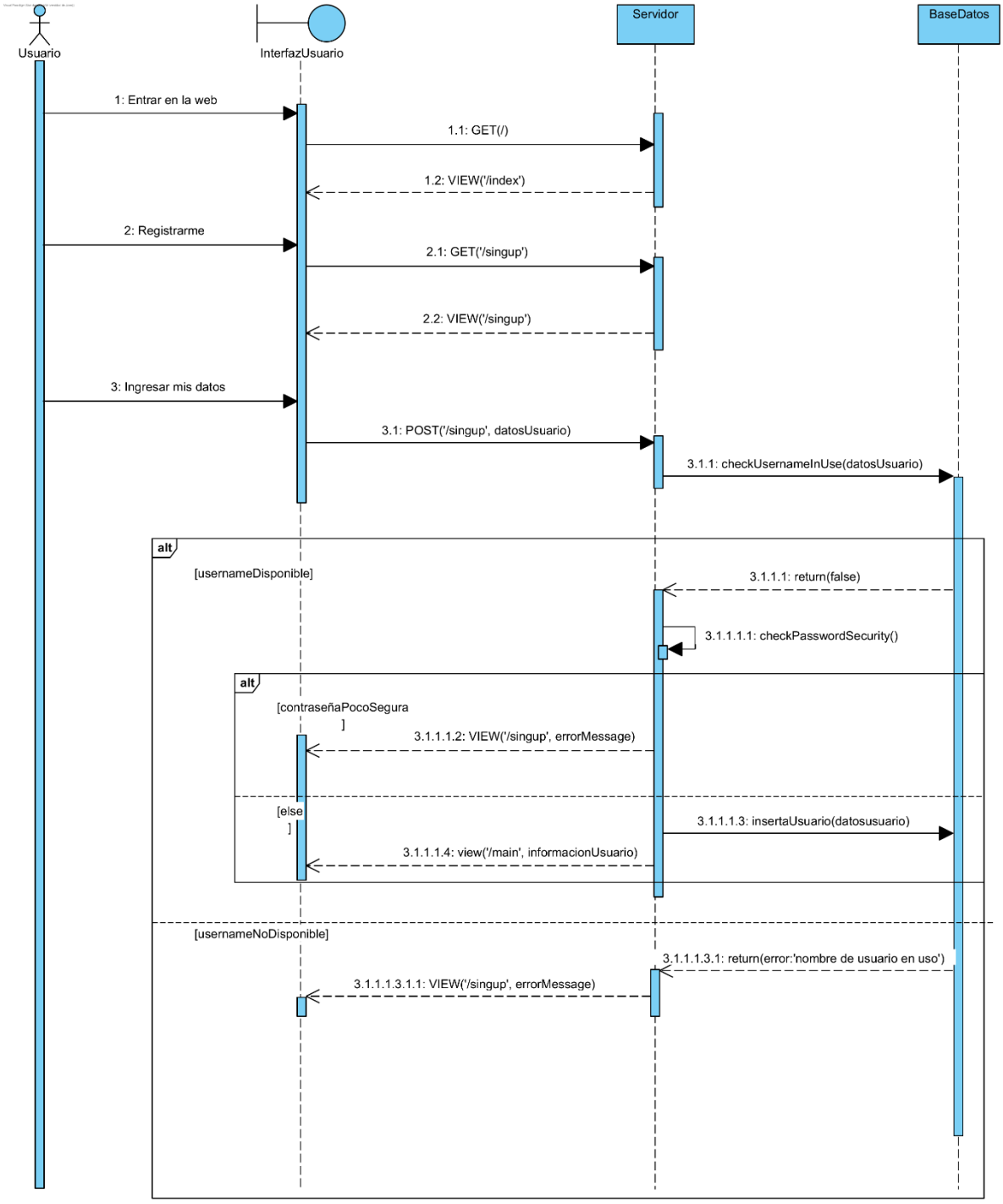

Ilustración 29. Diagrama de secuencia del proceso de registro

Implementación

Finalmente, procedemos con la codificación tanto de la parte *front-end* donde nos ocuparemos de las vistas, como de la parte *back-end* que se encarga de la lógica del servidor y la gestión de la base de datos. Partiremos de uno de los proyectos base de Glitch llamado *Glitch-hello-node*, y a raíz de ahí comenzaremos a cambiar el código.

Tras esto de manera automática Glitch realiza el despliegue de la aplicación y le asigna una dirección URL cuyo subdominio será el nombre de nuestro proyecto, dentro del dominio de *glitch.me*.

Para la parte *front-end* crearemos 3 archivos Handlebars, uno para cada una de las vistas a implementar, en estos se proporcionarán las instrucciones para generar la interfaz gráfica deseada, incluyendo la disposición de elementos, estilos, formatos y otros detalles visuales.

Para el desarrollo del back-end, se contará con el archivo `server.js`, el cual será responsable de levantar el servidor y configurar los elementos necesarios. Además, se utilizará un archivo de enrutamiento para dirigir adecuadamente las peticiones entrantes hacia las funciones y controladores correspondientes. En este caso, se implementarán tres enrutadores: `index.route.js, login.route.js` y `signup.route.js`.

Contaremos además con tres ficheros para el modelado de datos: en `API.js` definiremos las operaciones que los controladores pueden llamar para modificar la base de datos; en `db.config.js` se creará y configurará la base de datos, así como las diferentes tablas y, por último, en `SNAPSORT.sqlite` se recogerá la información, este será el fichero que nos sirva como base de datos.

Ilustración 30. Estructura de ficheros del proyecto al finalizar el primer sprint

Cómo se puede apreciar en la *Ilustración 30*, existen dos ficheros anteriormente no mencionados: `.env` y `package.json`. El archivo `.env` es un archivo de configuración que almacena variables de entorno, estas variables son accesibles en el código del proyecto y se utilizan para almacenar información sensible o configuraciones específicas del entorno de desarrollo, además es el único fichero que Glitch no comparte de forma pública con la comunidad. El archivo `package.json` también es un archivo de configuración propio de un proyecto Node.js, sin embargo, este contiene información sobre el proyecto, sus dependencias y scripts de ejecución.

La imagen de fondo que se muestra en las tres pantallas ha sido cargada gracias a la funcionalidad ASSETS de Glitch. Los principales estilos aplicados a esta vista se encuentran en el fichero `style.ccs`.

Tras finalizar el primer sprint la apariencia visual de la aplicación es la que se puede observar en las ilustraciones de las páginas 63 y 64.

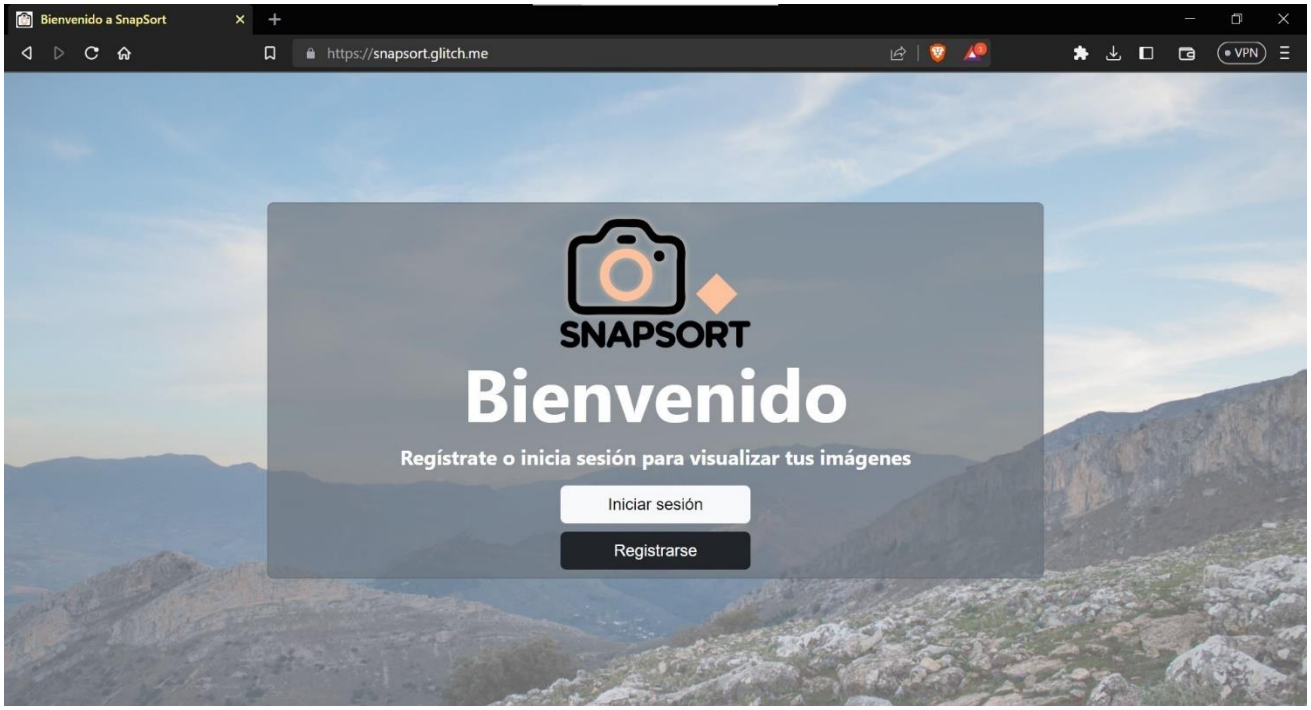

Ilustración 31. Vista Index

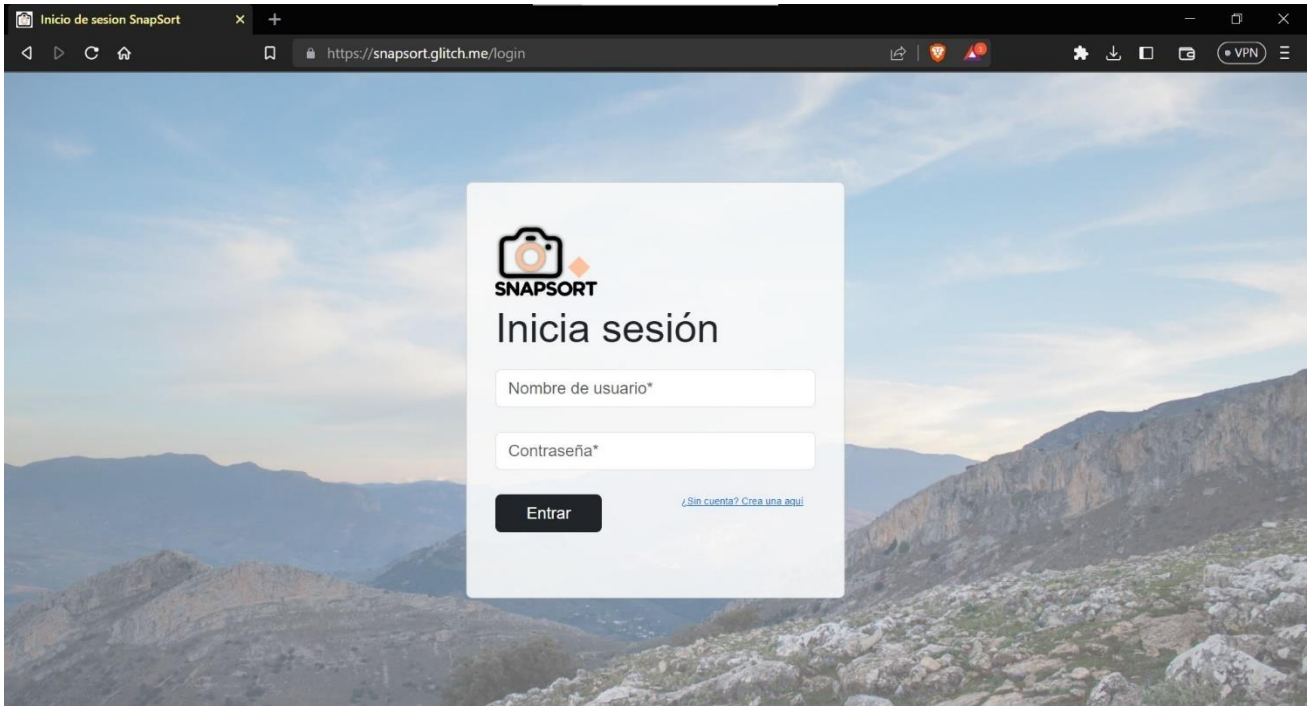

Ilustración 32. Vista inicio de sesión

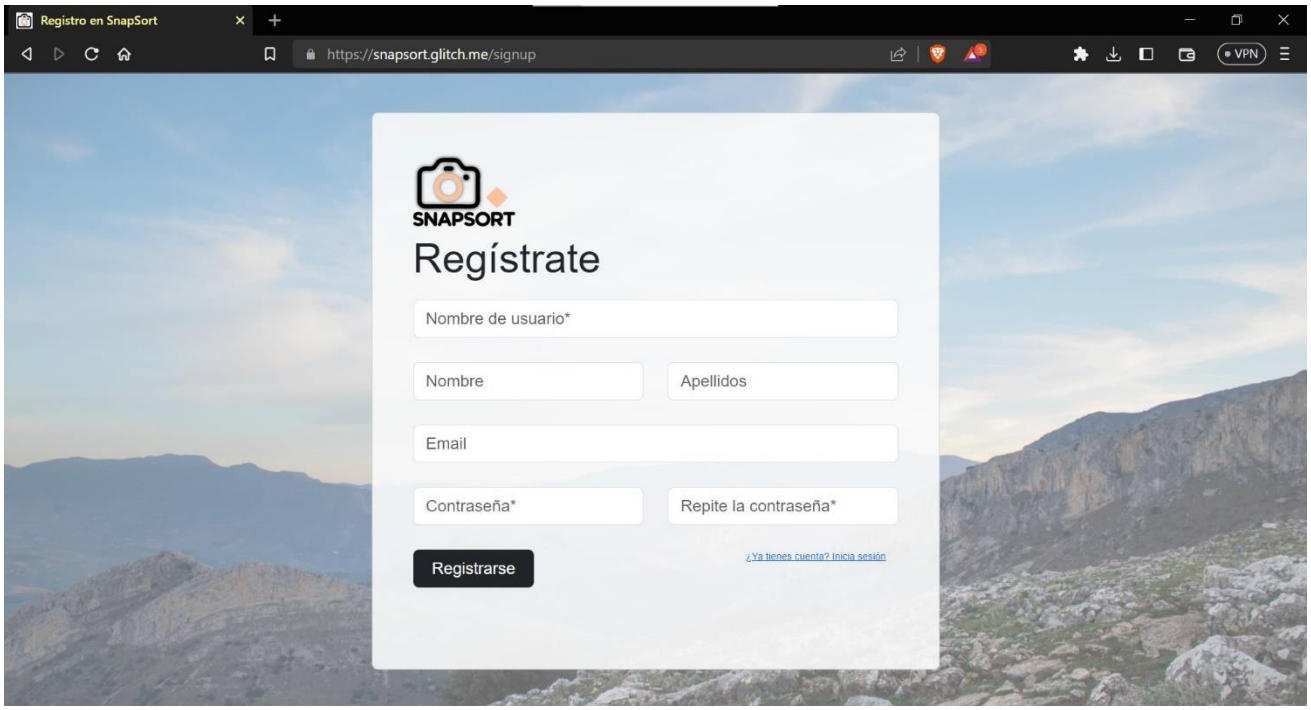

Ilustración 33. Vista de registro

4.5.2 Segundo sprint

Historias de usuario

Una vez contamos con la versión preliminar de nuestra aplicación procedemos a seguir incorporando historias de usuario a nuestro flujo de trabajo, aportando así funcionalidad a nuestra aplicación. Para esta segunda versión escogemos una única historia de la clase M: Importar imágenes y visualizarlas como un mosáico, como puede observarse en la *Ilustración 34*.

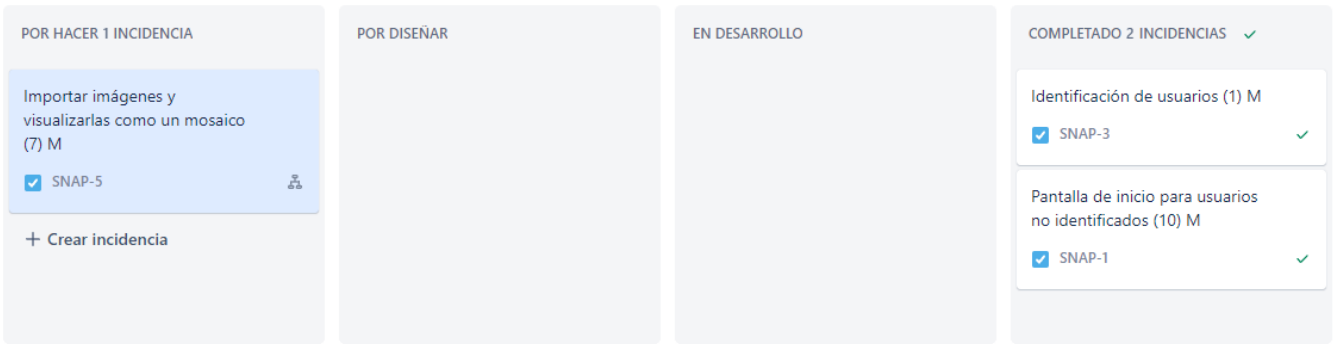

Ilustración 34. Estado del Kanban al inicio del segundo sprint

Si observamos la *Ilustración 34* junto a la historia de usuario recién añadida puede observarse un logo semejante a un árbol invertido esto indica que esa HU se divide en tareas más sencillas dentro del tablero Kanban. En este caso 4 subtareas como se muestra en la *Ilustración 35*.

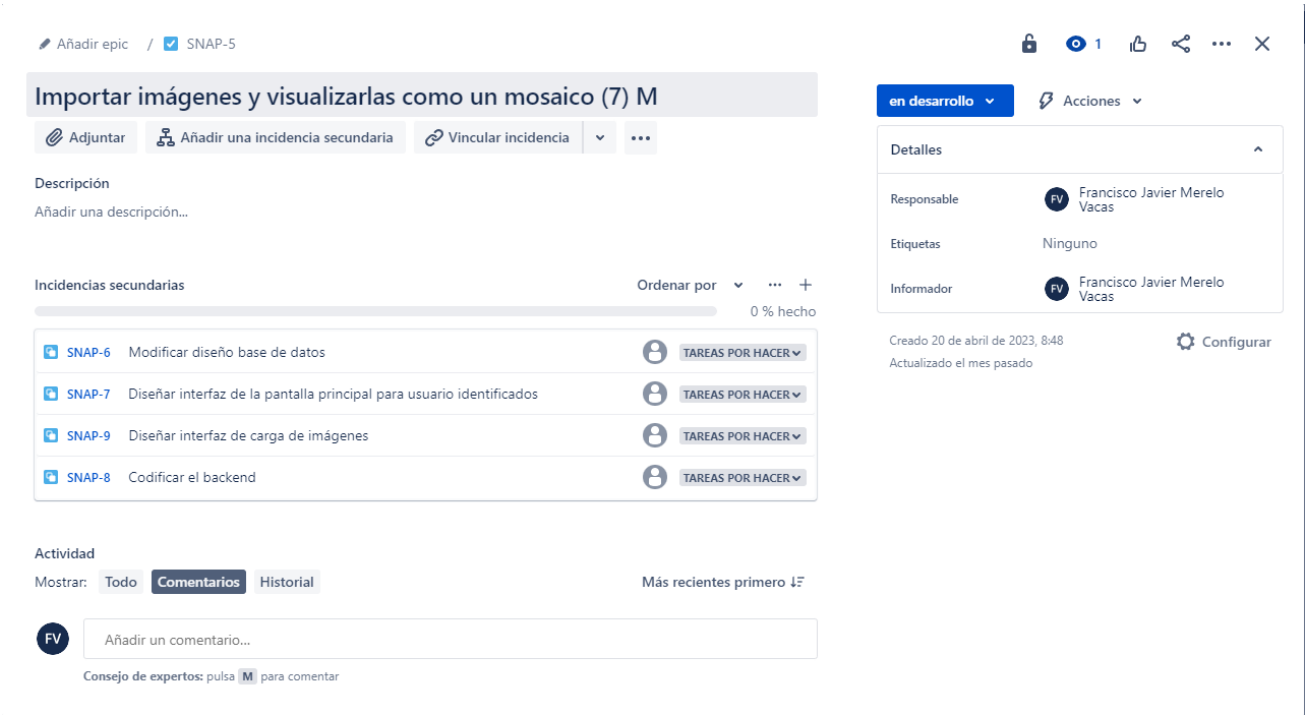

Ilustración 35. Subtareas dentro de la historia de usuario

Diseño de la base de datos

Para poder añadir esta nueva funcionalidad es necesario que nuestra base de datos permita el almacenamiento de las imágenes. No obstante, debido a las limitaciones de espacio en disco que posee la plataforma hemos optado por la siguiente opción en el diseño: no almacenar las imágenes, sino el enlace a esa imagen en un servicio de alojamiento de terceros, como Google Drive, y para que este proceso de carga de imágenes sea menos tedioso para el usuario se importará toda una carpeta, que se guardará en la base de datos. Por lo tanto, son necesarias varias entidades en nuestra base de datos como se muestra en la *Ilustración 36*.

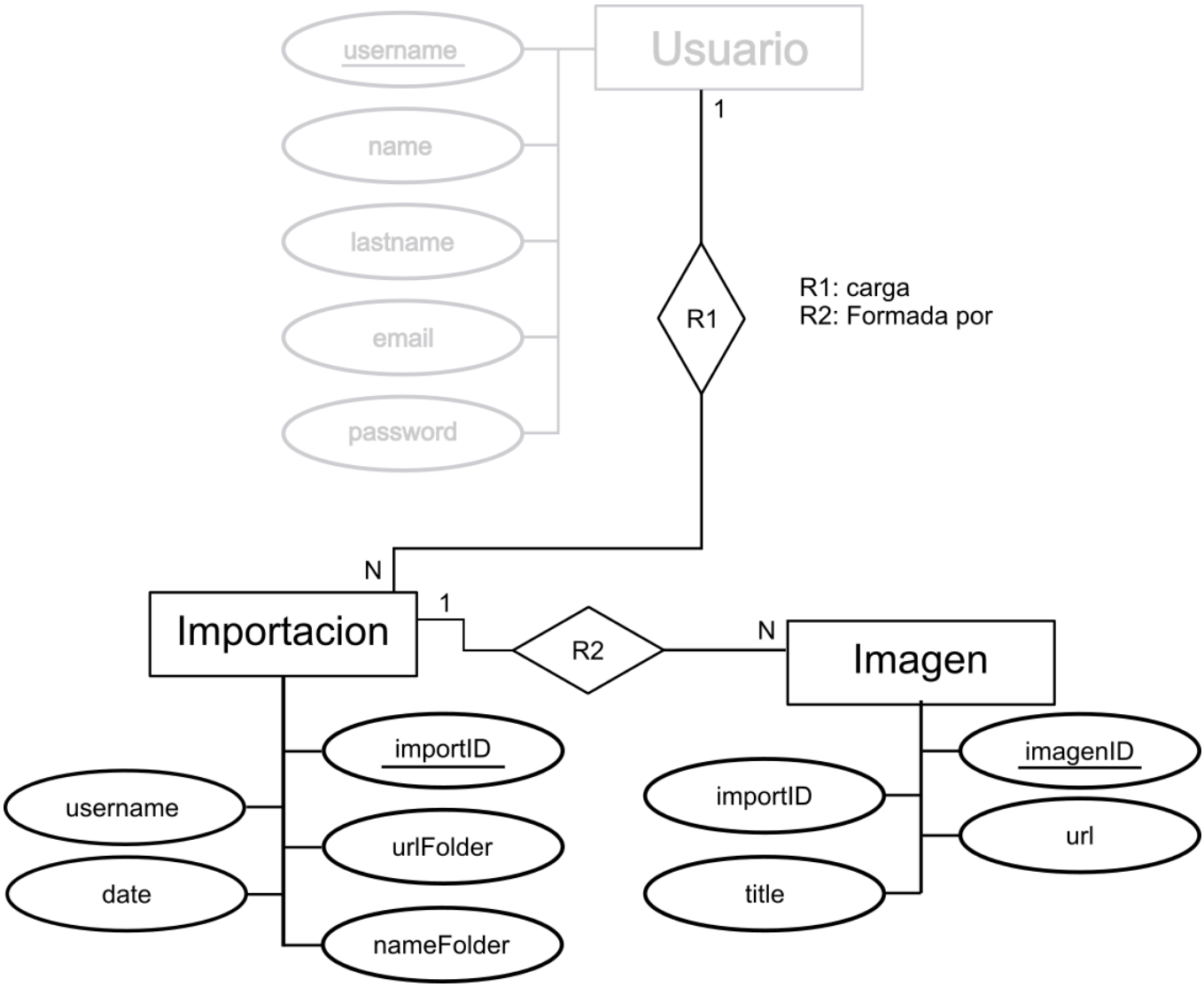

Ilustración 36. Esquema entidad-relación de la BD al inicio del segundo sprint. En gris los elementos ya existentes en la base de datos

La entidad importación representa al momento en el que se cargaron las fotos en la aplicación, por ello presenta los siguientes atributos:

- ImportID: guarda el identificador único de cada una de las tuplas, es generado de forma automática por la base de datos.
- Username: almacenará el nombre de usuario que realizó esa importación, es una clave foránea a la tabla usuario.
- urlFolder: Se utilizará esta columna para almacenar el valor de la dirección URL a la carpeta de Google Drive.
- Date: registra la fecha en la que se llevó a cabo el proceso de carga de imágenes.

- NameFolder: registra el nombre con el que se representa la importación, permitirá al usuario identificar la carpeta correspondiente en Google Drive que cargó.

La entidad imagen representa a cada una de las imágenes de las carpetas y cuenta con los siguientes atributos:

- ImagenID: almacena el identificador único que la base de datos genera de forma automática para cada imagen.
- ImportID: almacena el identificador de la importación en la que se cargó la imagen, es una clave foránea de la tabla importación.
- url: recoge la dirección de acceso a la imagen a través de Google Drive.
- title: guarda el título que recibe la imagen, por defecto será el mismo que el que posee en Google Drive.

Diseño de las vistas

Procedemos a realizar los *sketches* de las dos nuevas vistas que vamos a añadir a nuestra aplicación. La primera será la vista principal de la aplicación, en ella deben de poder contemplarse las diferentes importaciones que el usuario realice, así como las imágenes en forma de cuadrícula, y la segunda vista será la que nos permite añadir una nueva carpeta como importación.

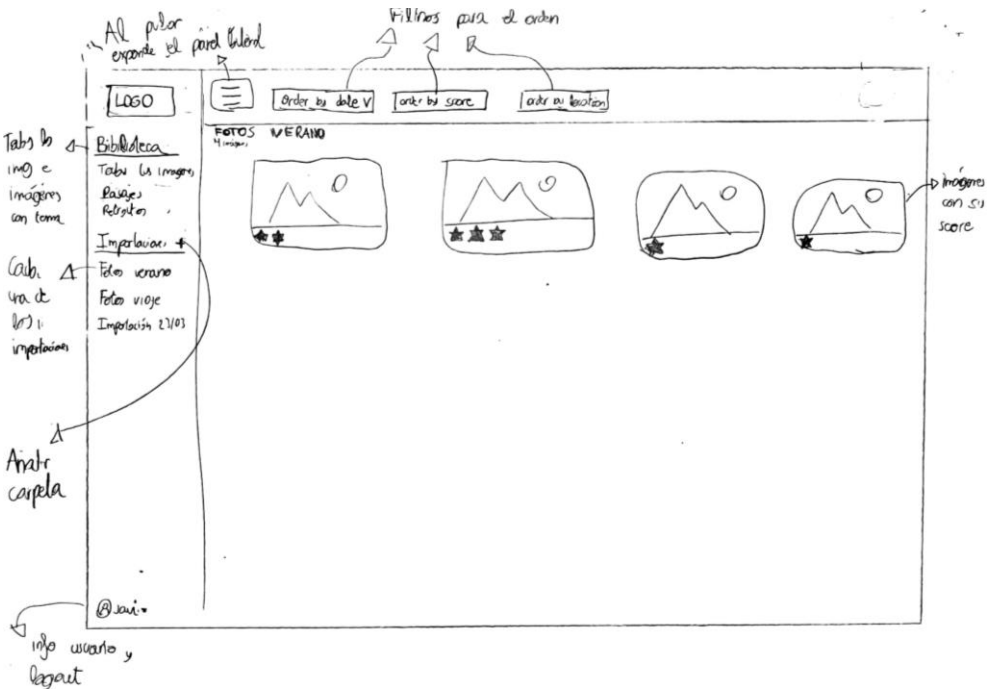

Ilustración 37. Sketch de la vista home

La segunda nueva vista a desarrollar es una ventana en la que el usuario pueda añadir una nueva carpeta; esta vista será una ventana *modal*, es decir, una pequeña ventana que se despliega sobre la vista *Home* difuminando el fondo de forma que este proceso se realice de forma más eficaz sin la necesidad de cargar una nueva vista.

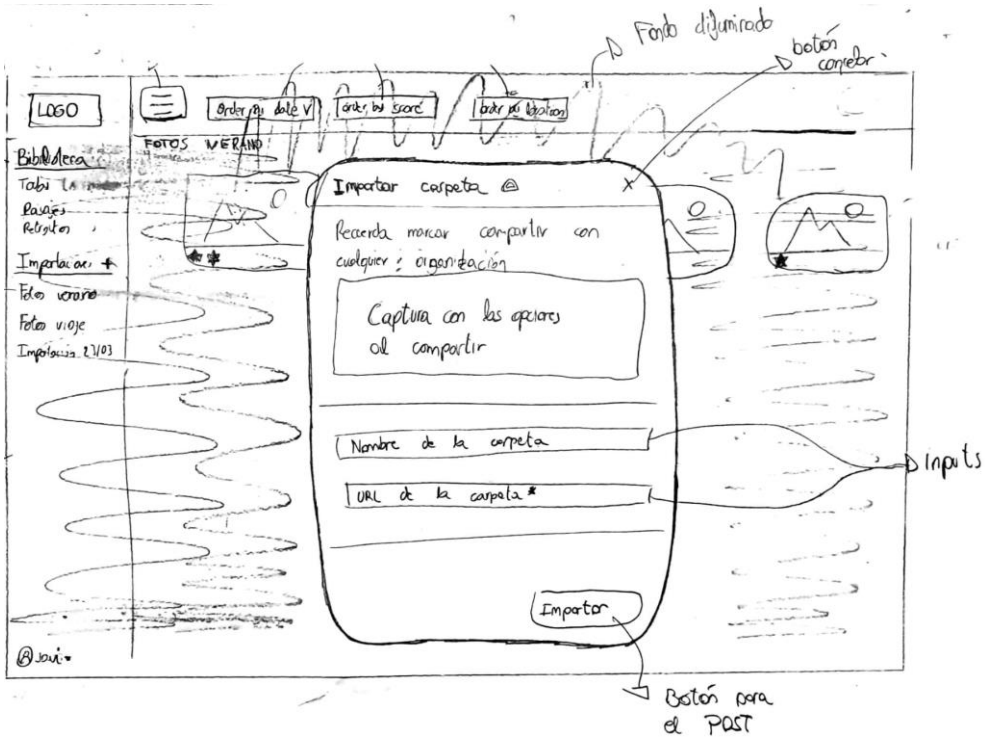

Ilustración 38. Sketch del modal para importar una nueva carpeta

Diagrama de Secuencia

Presentamos los diagramas de secuencia que ilustran el funcionamiento de estas nuevas funcionalidades.

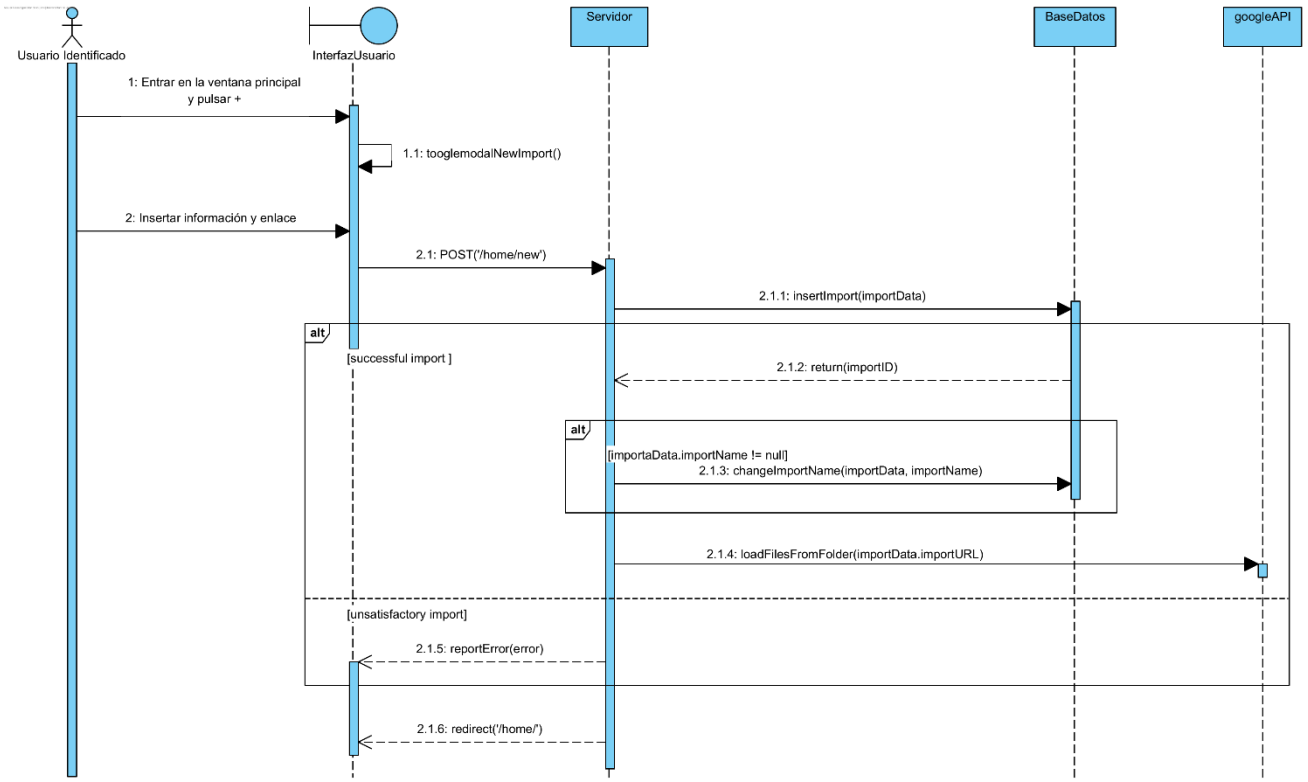

Ilustración 39. Diagrama de secuencia del proceso de carga de carpetas

La *Ilustración 39* muestra cómo las carpetas que importemos desde Google Drive pueden llevar un nombre asociado o asignársele uno por defecto. Una vez que se realice el proceso de inserción en la base de datos, se procederá a obtener la información de la carpeta a través de la API de Drive.

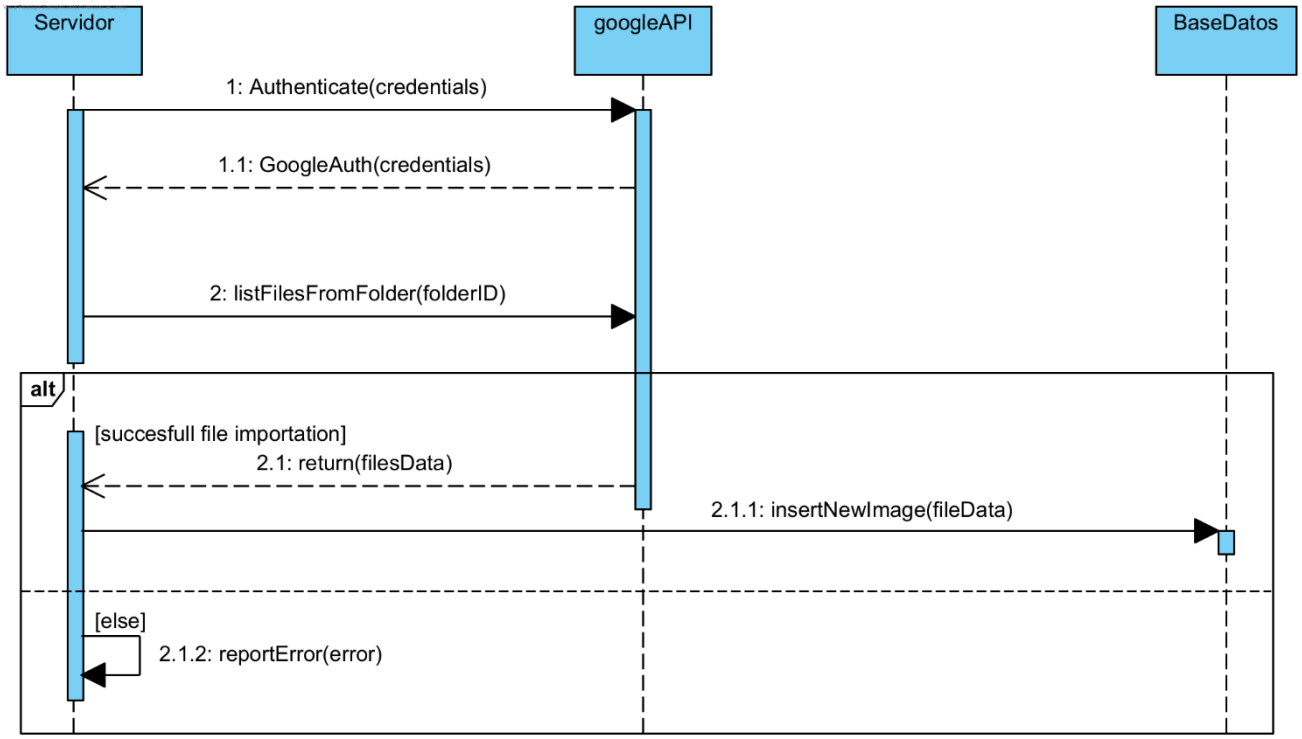

Ilustración 40. Diagrama de secuencia de la carga de imágenes

La *Ilustración 40* presenta el proceso de obtener la información de las imágenes, en nuestro caso identificador, título y enlace. Como se puede comprobar, una vez obtenida esta información se almacena en la base de datos, donde se crearán tantas tuplas como imágenes existan en la carpeta original.

Implementación

En esta versión se ha creado un nuevo *router* y una nueva vista: `home.route.js` y `home.hbs`, respectivamente. Además de un archivo para controlar la conexión a la API de Google Drive, se puede conocer más sobre el funcionamiento de este en el apartado 7.1.

La vista home posee una barra lateral que puede plegarse y desplegarse a través del botón de la barra superior. El funcionamiento de este componente está recogido en el fichero `sidebar.css` y está basado en uno de los ejemplos que ofrece Bootstrap.

Tras finalizar el segundo sprint la apariencia visual de las nuevas vistas añadidas a la aplicación es la siguiente:

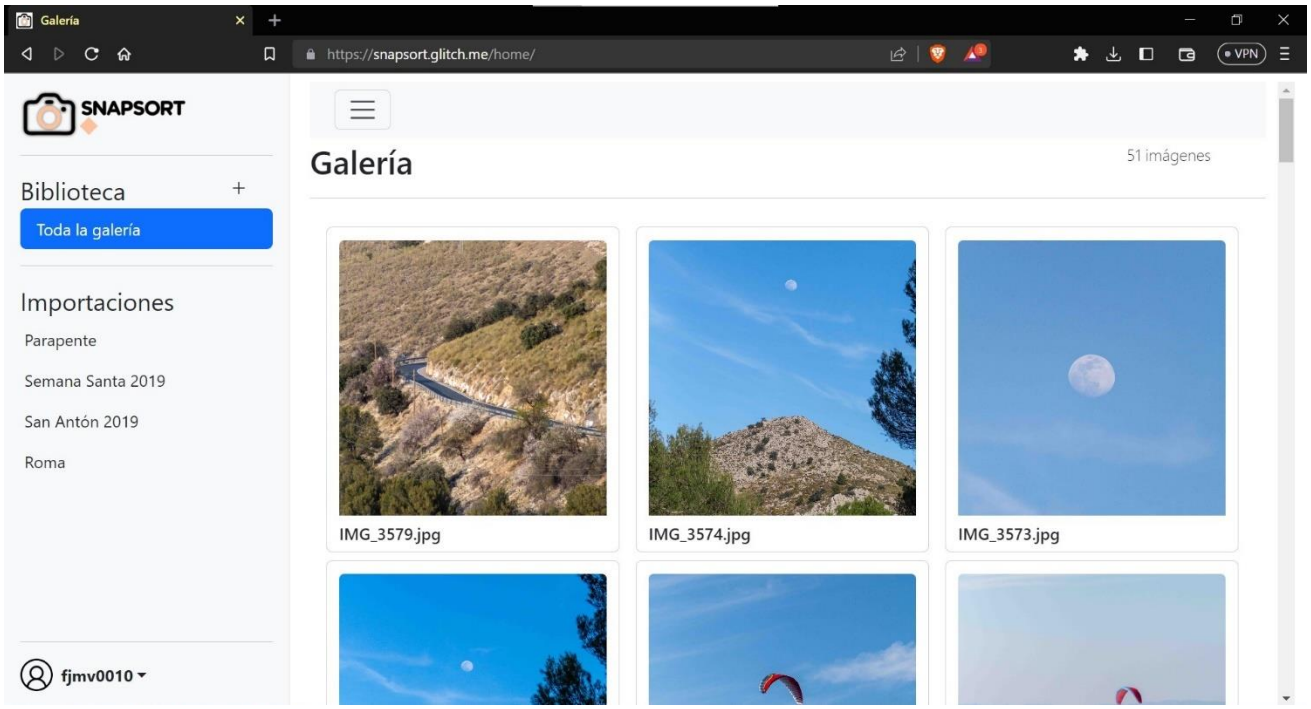

Ilustración 41. Vista Home al final del segundo sprint

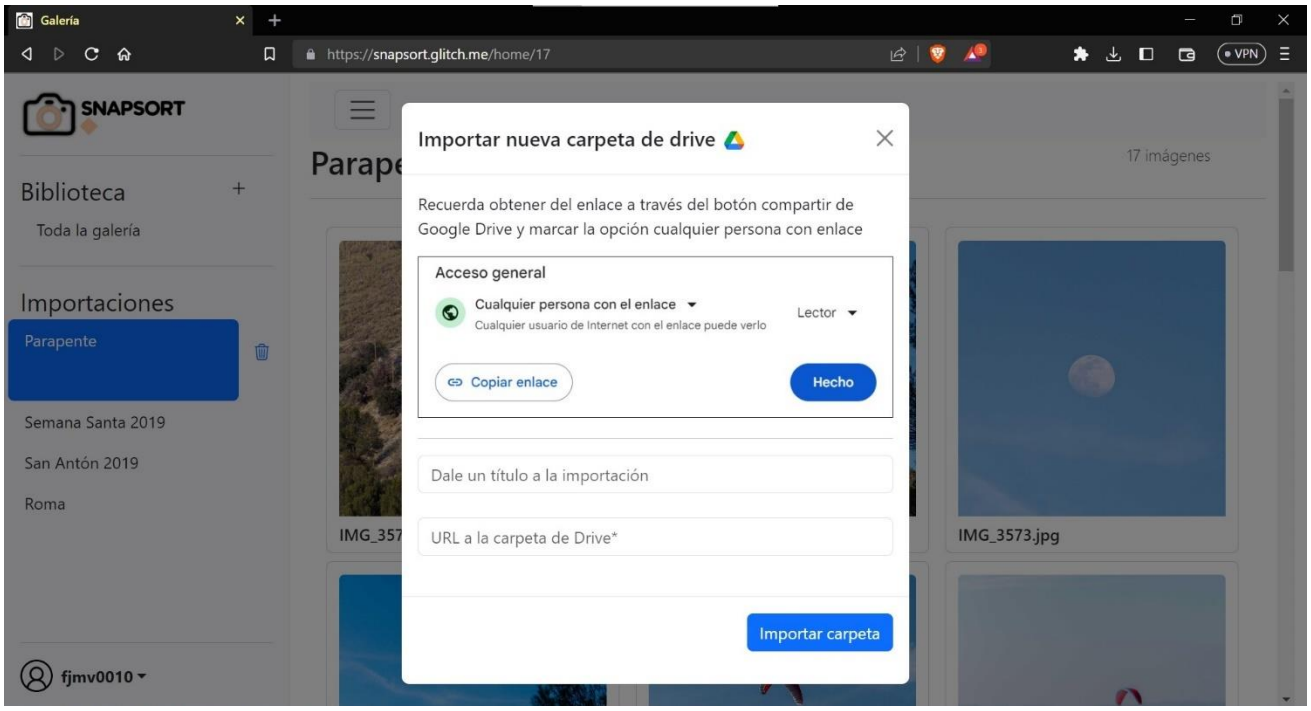

Ilustración 42. Menú para importar carpeta al final del segundo sprint

Durante el desarrollo de este sprint nos hemos encontrado con dos problemas relacionados con la carga de imágenes. En primer lugar, cuando las imágenes tenían una gran calidad, y por lo tanto un gran peso, la carga de estas resultaba muy lenta, afectando negativamente a la experiencia del usuario en la aplicación. En segundo lugar, tras un uso continuado de la aplicación, al realizar varias peticiones de carga a Google, nos encontramos

con un error HTTP *403 forbidden* que impedía la carga de las imágenes. Este código de error indica que el servidor ha entendido la petición, pero se niega a cumplirla. En este caso, Google nos estaba indicando que se habían superado los límites de la API.

Para solucionar ambos problemas se siguieron las siguientes estrategias; en primer lugar, se implementó la carga perezosa de CSS, de forma que las imágenes se cargan en el cliente solamente cuando estén cercanas al campo de visión. Tras eso, se decidió crear un conjunto de imágenes de ejemplo más ligeras, que permitieran una carga más rápida y eficiente en la aplicación. Esta decisión permitió mejorar la experiencia del usuario y retrasar los límites de carga, manteniendo la calidad visual de la aplicación. No obstante, el límite de peticiones por usuario de la API sigue siendo bajo debido a que contamos con una cuenta gratuita. Por todo ello, la solución final ha sido importar una imagen por defecto, que se visualizará en caso de que no pueda cargarse la imagen original.

4.5.3 Tercer sprint

Historias de usuario

Nuestra aplicación cuenta ya con la funcionalidad más básica, por lo que esta segunda versión vamos a añadir historias de usuario de las categorías *should* y *could* que aportarán valor añadido como herramienta de categorización de imágenes.

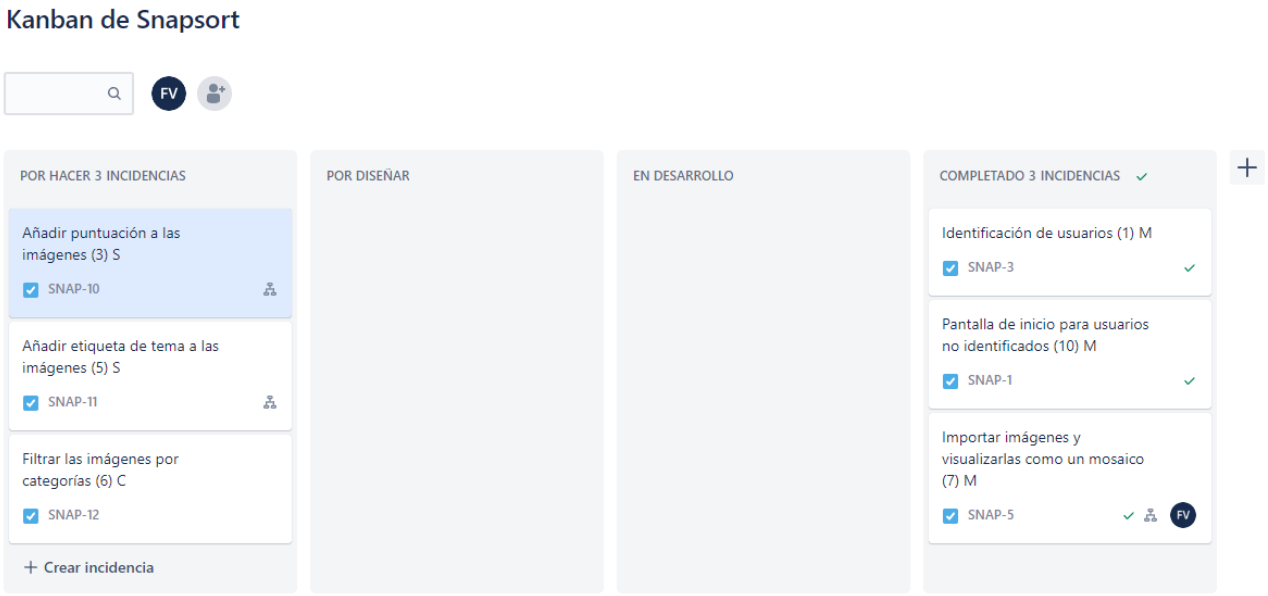

Ilustración 43. Estado del Kanban al inicio del tercer sprint

Como se puede comprobar en la *Ilustración 43* hemos seleccionado tres historias de usuario con las que trabajaremos a lo largo de esta versión: Añadir puntuación a las imágenes, añadir etiqueta de tema a las imágenes y filtrar las imágenes por categoría. Pasaremos a incorporar todo el sistema de categorización de imágenes, para eso debemos de diseñar una pantalla en la que podamos clasificar las fotografías que se incorporen a la aplicación y modificar la vista *home* para que nos permita mostrar las imágenes según un orden establecido.

Diseño de la base de datos

De nuevo tendremos que modificar el esquema de la base de datos para adecuarse a las nuevas características, como se puede ver en la *Ilustración 44* se han añadido dos nuevos atributos a la tabla imagen. Un campo *score* que recogerá la puntuación que se le da a esa imagen y un campo topic que representa la temática de la foto.

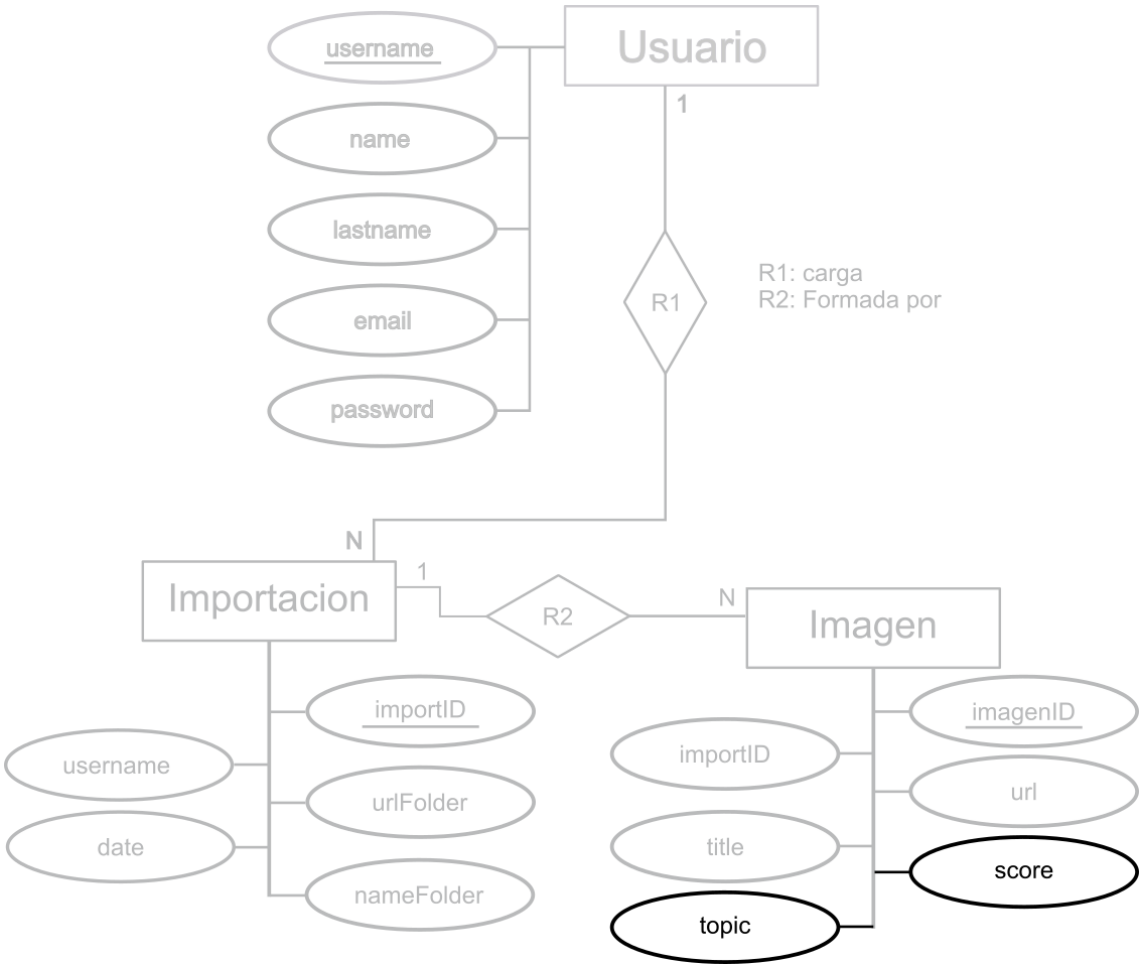

Ilustración 44. Esquema ER al inicio del tercer sprint. En gris los elementos ya existentes en la base de datos

En este sprint vamos a mostrar las posibilidades que brinda la consola en tareas de administración. En lugar de eliminar la base de datos y volver a construirla a través de código Javascript, vamos a utilizar los comandos propios de SQLite para modificar las tablas ya existentes.

```
app@snapsort:~ 18:30
$ sqlite3 src/database/SNAPSORT.sqlite
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
sqlite> .databases
seq  name             file
---  ---------------  ----------------------------------------------------------
0    main             /app/src/database/SNAPSORT.sqlite
sqlite> .schema imagen
CREATE TABLE imagen(
          imagenID INTEGER PRIMARY KEY AUTOINCREMENT,
          importID INTEGER NOT NULL,
          url TEXT NOT NULL,
          title TEXT,
          FOREIGN KEY(importID ) REFERENCES importacion(importID) ON DELETE CASCADE
          );
sqlite> ALTER TABLE imagen ADD COLUMN score INTEGER CHECK(score >= 0 AND score <= 5);
sqlite> ALTER TABLE imagen ADD COLUMN topic TEXT CHECK(length(topic) <= 25);
sqlite> .schema imagen
CREATE TABLE imagen(
          imagenID INTEGER PRIMARY KEY AUTOINCREMENT,
          importID INTEGER NOT NULL,
          url TEXT NOT NULL,
          title TEXT, score INTEGER CHECK(score >= 0 AND score <= 5), topic TEXT CHECK(length(topic) <= 25),
          FOREIGN KEY(importID ) REFERENCES importacion(importID) ON DELETE CASCADE
          );
sqlite>
```

Ilustración 45. Cambios en la base de datos a través de la terminal de Glitch

Como es observable en la *Ilustración 45*, con dos simples ordenes hemos podido modificar la tabla `imagen` y, al instante, apreciar los cambios en el esquema favoreciendo la persistencia de datos. Para representar las imágenes no clasificadas, tendrán valor cero en el campo *score,* por lo que de nuevo a través de la terminal hemos podido modificar los valores de las tuplas.

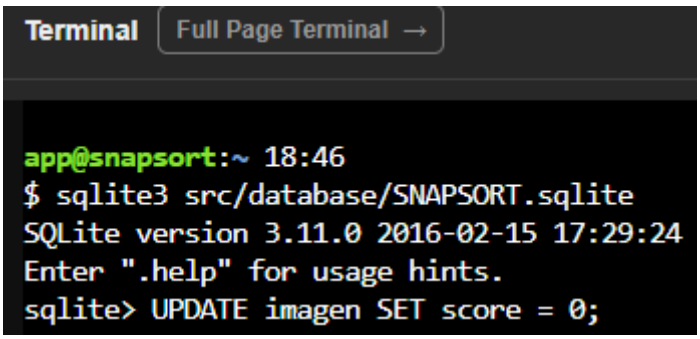

Ilustración 46. Modificando los valores de las tuplas a través de consola.

Diagrama de Secuencia

Una vez que el repositorio de información está listo para almacenar las imágenes clasificadas, diseñaremos la interacción entre los elementos de la aplicación a través de un diagrama de secuencia.

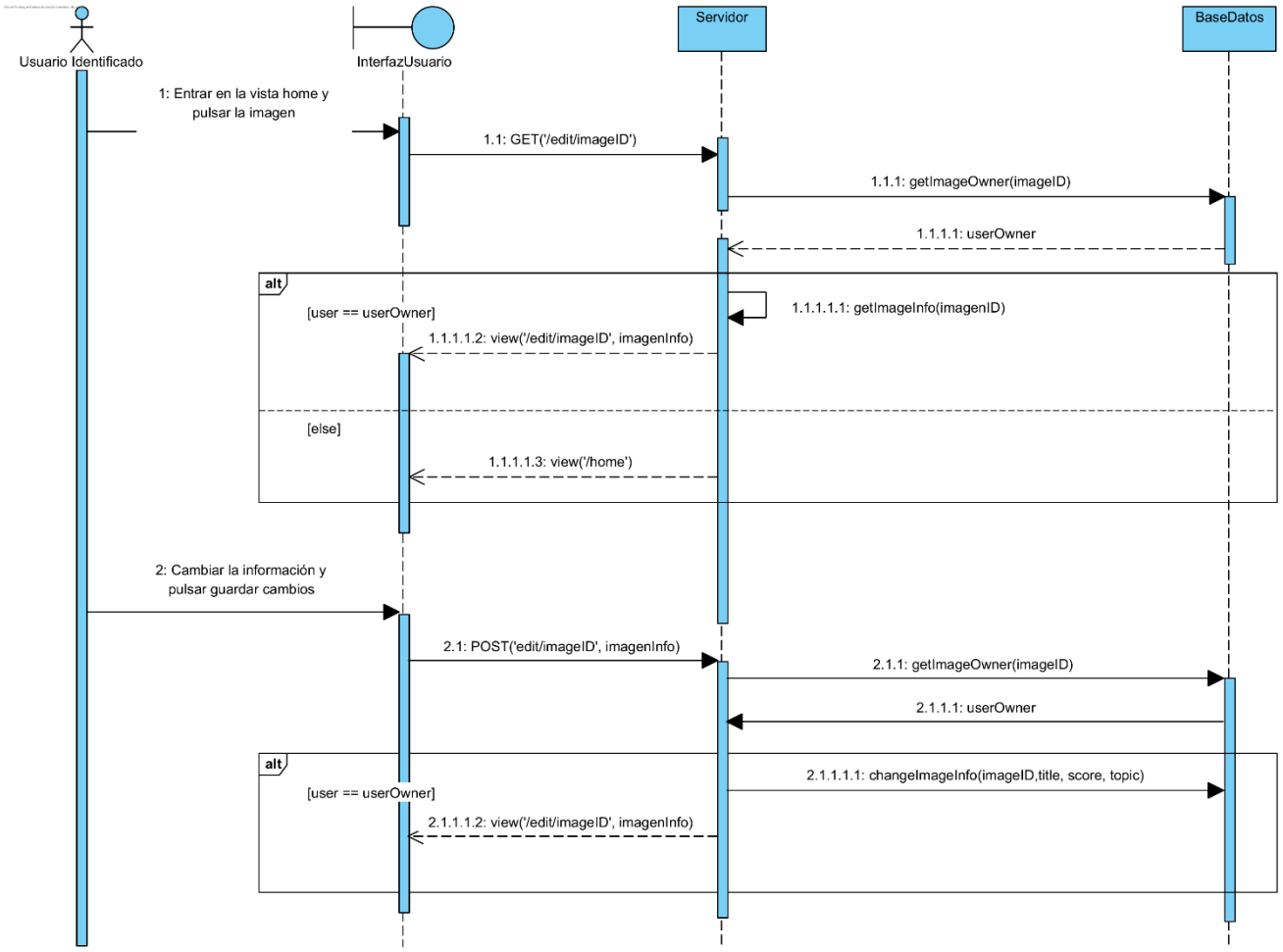

Ilustración 47. Diagrama de secuencia del proceso de categorización de imágenes

Para el diseño de la funcionalidad de filtrado de imágenes debemos tener en cuenta la eficiencia, ya que la aplicación está diseñada para almacenar gran cantidad de imágenes. En el caso del filtrado de imágenes para que se visualicen en orden de puntuación, debemos reordenar la estructura de datos en la que se encuentra recogida la información de cada una de las imágenes. El proceso de ordenar una colección puede ser altamente costoso computacionalmente, por lo que deberá ser nuestro servidor el que envíe al cliente los datos ya ordenados bajo los diferentes criterios. Sin embargo, el proceso de clasificar las imágenes por categorías es menos costoso, y solo

habrá que eliminar de la vista aquellas imágenes que no coincidan con el filtro establecido a través de la temática.

Diseñaremos la comunicación entre los distintos objetos del sistema para atender la petición de ordenación en función de un parámetro enviado desde el cliente.

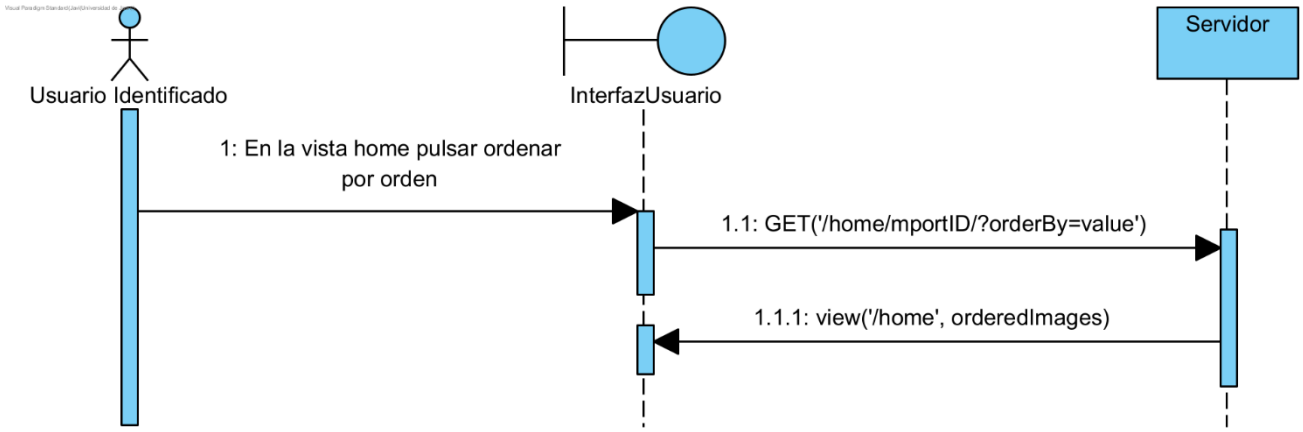

Ilustración 48. Diagrama de secuencia del proceso de ordenación de imágenes

Podría parecer observando la *Ilustración 48* que se trata de un proceso simple, realmente este diagrama se ha simplificado el flujo de comunicación necesario para obtención de las imágenes, eliminando las comprobaciones de seguridad anteriormente explicadas.

Diseño de las vistas

Finalmente diseñaremos las nuevas interfaces para este sprint, en este caso diseñaremos una nueva vista para la clasificación de imágenes (*Ilustración 50*) y modificaremos el sketch de la vista *home* para que permita clasificar las imágenes (*Ilustración 49*).

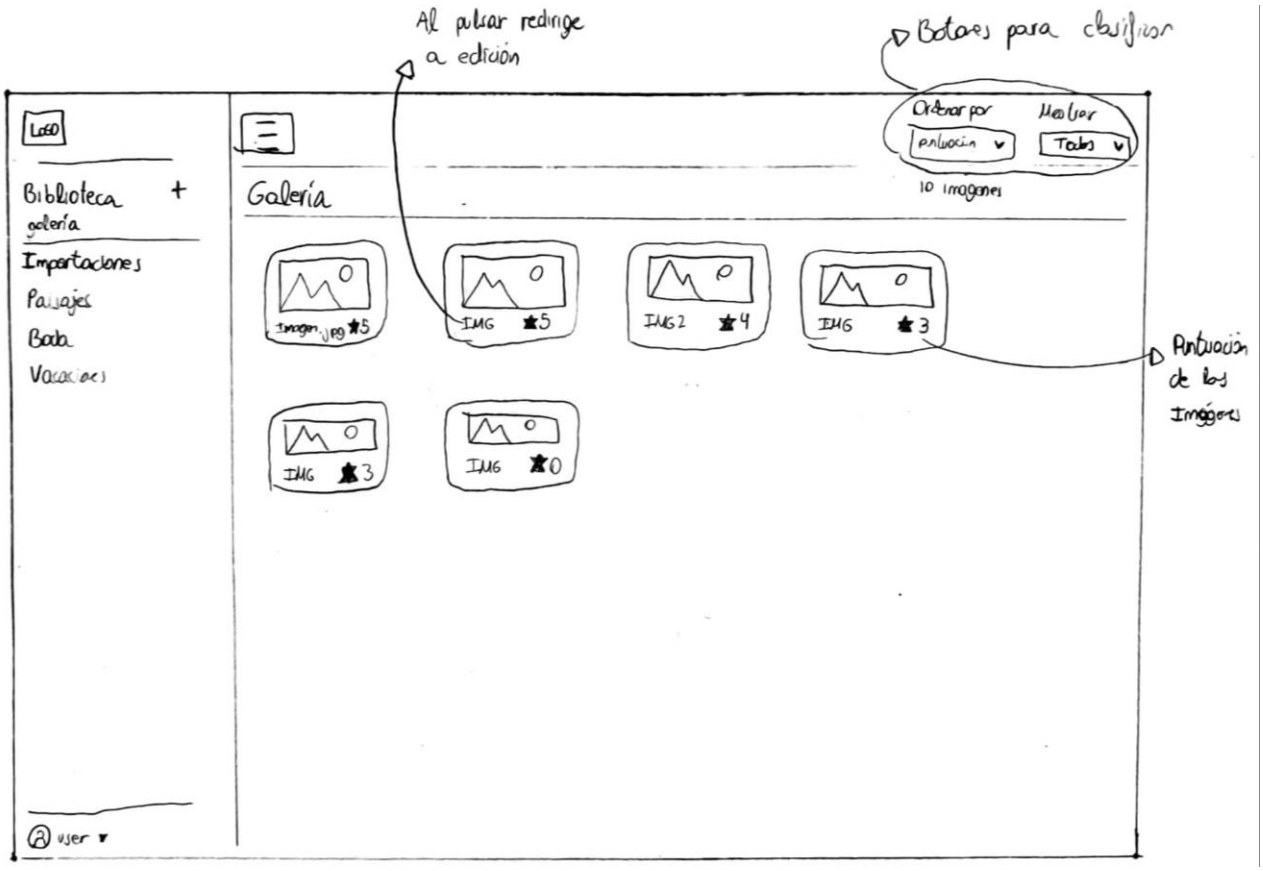

Ilustración 49. Sketch de la nueva vista home que permite clasificar imágenes

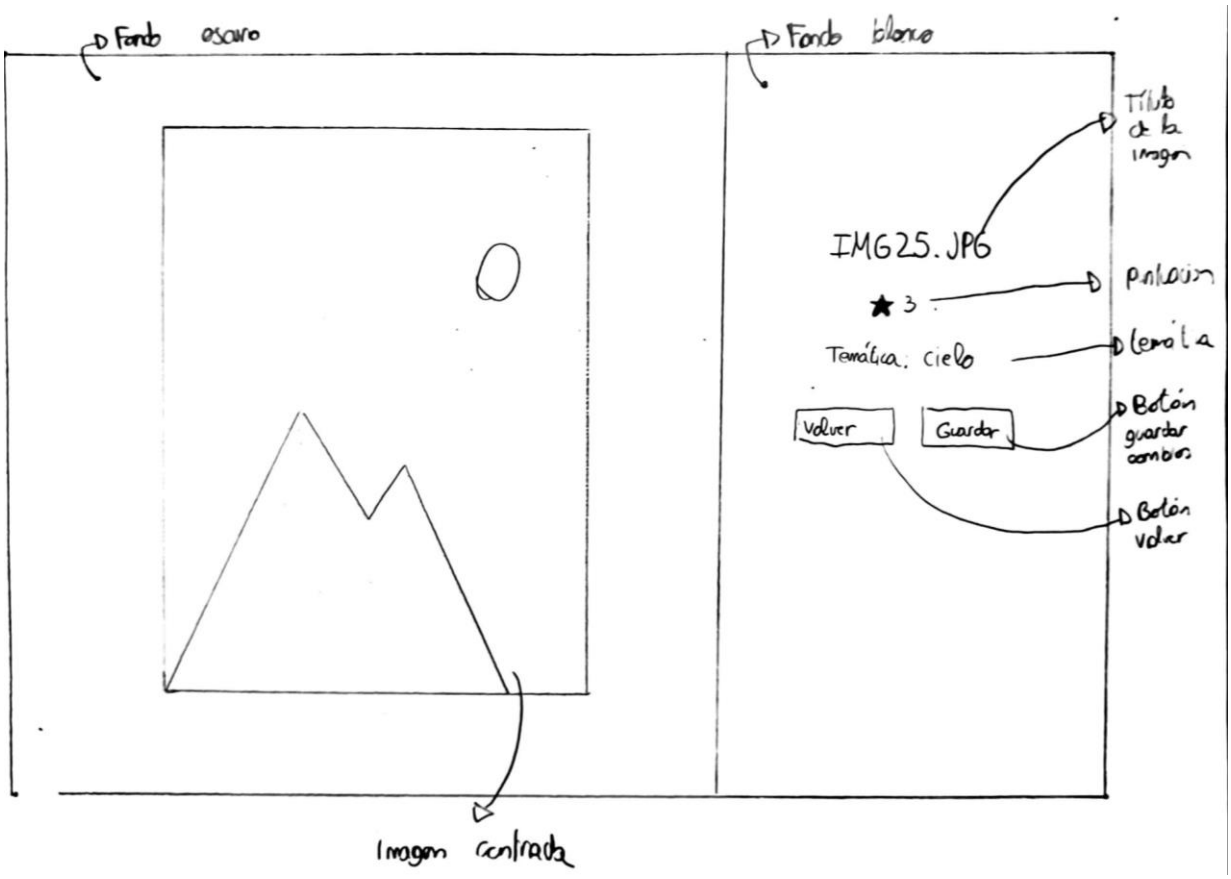

Ilustración 50. Sketch de la vista de clasificación de imágenes

Implementación

En esta versión se ha creado un nuevo *router* y una nueva vista: `edit.route.js` y `edit.hbs`, respectivamente. De forma que la estructura de los archivos del proyecto quedaría como se puede comprobar en la Ilustración *51*.

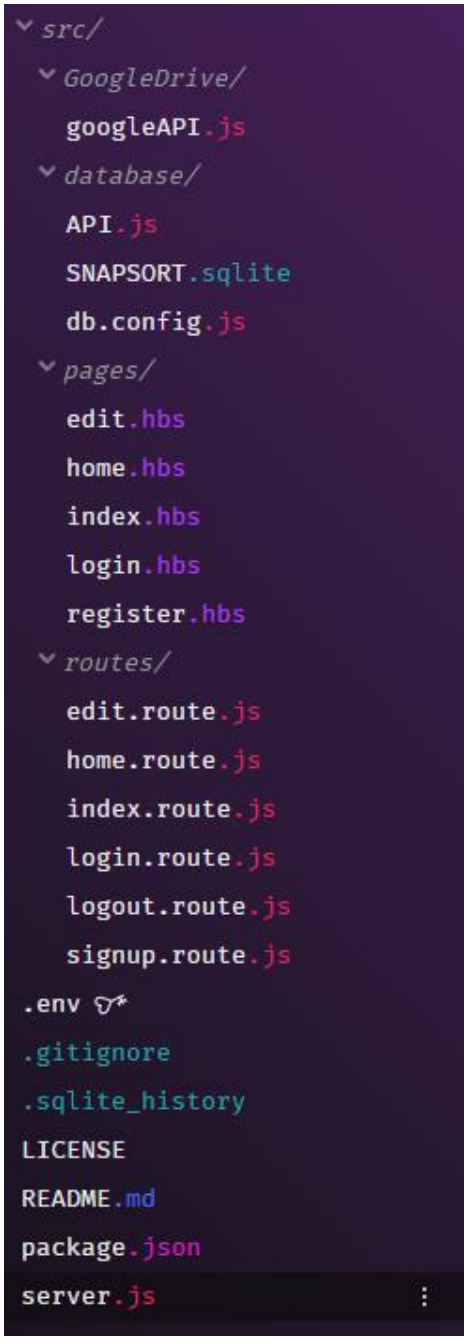

Ilustración 51. Estructura de ficheros del proyecto al final del tercer sprint

Durante este *sprint* hemos intentado poner a prueba cómo de compatible es Glitch con una de las prácticas propia de las metodologías ágiles: *Test-*

Driven Development (TDD) es una técnica que consiste en comenzar a desarrollar los tests unitarios y, a raíz de los errores en estos, ir completando el código que solvente los errores. Lamentablemente, nos hemos encontrado con que no ha sido posible ejecutar tests unitarios. No hemos encontrado documentación al respecto y suponemos que la raíz de este problema se encuentra en que el proceso de despliegue en Glitch es completamente automático. Esperamos que en futuras actualizaciones de la plataforma se considere la importancia de las pruebas unitarias en el desarrollo y se incorpore soporte para su ejecución en Glitch.

El servidor utiliza para ordenar la colección el método de Javascript `sort()`, la complejidad de este método depende de la implementación del mismo, de manera que no será la misma si se ejecuta en el lado del servidor, que en los distintos navegadores que pueden utilizarse como cliente (Mozilla Development Network , s.f.). Es sabido que Node.js está construido con el motor de JavaScript V8 (OpenJS Foundation, s.f.) y es en la propia documentación del motor donde especifican que utilizan un algoritmo llamado Timsort cuya complejidad es de $O(n \log n)$ (V8, 2018).

El aspecto final de la vista de edición y etiquetado de imágenes en nuestra aplicación puede consultarse en la *Ilustración 52*.

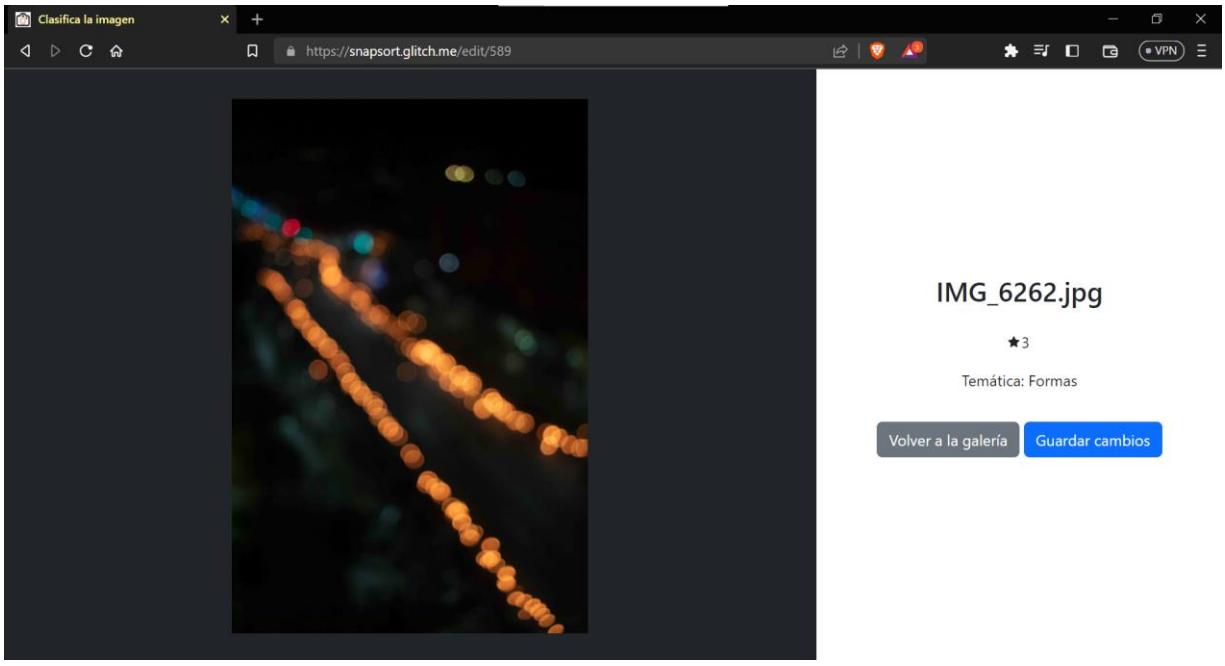

Ilustración 52. Vista de clasificación de imágenes

4.5.4 Mejoras futuras

La aplicación en su estado actual representa un hito importante en el desarrollo del proyecto, y se considera una versión funcional y completa. Como en toda aplicación, existen oportunidades para expandir y mejorar aún más sus funcionalidades en el futuro. Al mantener un enfoque de integración continua, se podrían implementar nuevas características para abordar necesidades específicas de los usuarios, así como realizar mejoras adicionales en términos de rendimiento, usabilidad y seguridad.

Se presentan a continuación algunas de las mejoras que podrían realizarse en un futuro:

- Permitir el inicio de sesión a través herramientas de terceros como OAuth
- Integración de soporte para otros servicios de almacenamiento en la nube como Dropbox, OneDrive, etc.
- Permitir a los usuarios compartir con otros usuarios sus importaciones, es decir, importaciones colaborativas.
- Soporte para que las imágenes puedan estar etiquetadas con más de una temática.

5. CONCLUSIONES

En el presente capitulo se expondrá la valoración personal del autor en relación con el uso de la plataforma Glitch, así como las posibles aplicaciones y oportunidades que se derivan de su utilización. Se reflexionará sobre la experiencia adquirida durante el desarrollo del trabajo y se analizarán las posibilidades que ofrece la herramienta en términos de desarrollo de aplicaciones. Además, se presentarán posibles líneas de trabajo y mejoras que podrían implementarse en futuros proyectos.

5.1 Conclusiones

Uno de los objetivos de este TFG era validar en qué medida la plataforma Glitch está orientada a perfiles con conocimientos técnicos escasos. Tras su estudio y análisis, podemos concluir que, efectivamente, la herramienta está diseñada para facilitar el desarrollo de aplicaciones sin que los usuarios tengan que preocuparse por la configuración del entorno de ejecución ni por la gestión de versiones. Además, dado que la plataforma sigue una metodología DevOps, la parte de operaciones es transparente al usuario, lo que hace que el proceso de desarrollo sea mucho más amigable para los perfiles con menos experiencia técnica.

A pesar de las numerosas ventajas que ofrece Glitch para los perfiles objetivo, hay que tener en cuenta que su escalabilidad es muy limitada y, por tanto, puede no ser la herramienta más adecuada para construir una aplicación para miles de usuarios. En este sentido, la propia plataforma ofrece funcionalidades de exportación a DigitalOcean en caso de que sea necesario aumentar la capacidad de cómputo o de almacenamiento. Además, aunque permite la codificación en parejas, al no poder controlar las versiones, puede resultar difícil coordinar a un equipo de desarrolladores. En cualquier caso, para proyectos más pequeños, para el prototipado de ideas o para fines educativos, Glitch puede ser una opción muy interesante.

5.2 Comentarios tras el uso

La herramienta Glitch es una excelente opción para el desarrollo de aplicaciones debido a sus numerosas funcionalidades que simplifican el proceso de codificación. Por ejemplo, es muy cómodo que los cambios realizados en el código se reflejen en la versión desplegada de manera casi instantánea. Además, la herramienta *assets* resulta muy útil, ya que permite subir imágenes a la plataforma y proporciona un enlace CDN a las mismas, lo que facilita su uso en la aplicación. Otra funcionalidad práctica que ofrece Glitch es la herramienta *rewind*, que permite visualizar los cambios realizados con la línea temporal y resalta las líneas de código modificadas, lo que resulta muy útil para la depuración de errores en el código. En conjunto, estas

funcionalidades hacen que Glitch sea una herramienta eficaz y amigable para el desarrollo de aplicaciones web.

Sin embargo, durante el desarrollo del proyecto, se han identificado ciertas limitaciones en la plataforma que podrían mejorarse en futuras versiones. Por ejemplo, la herramienta solo permite tener abierta una pestaña en el editor de código, lo que dificulta la navegación entre archivos cuando se trabaja con varios al mismo tiempo. Además, no ofrece sugerencias para autocompletar el código durante la codificación, lo que puede resultar incómodo para programadores menos experimentados. Aunque se pueden usar ramas de Git, la rama activa siempre será la que esté desplegada, lo que limita su utilidad. Finalmente, aunque el terminal es una herramienta muy útil, su uso requiere cierto conocimiento técnico y puede no ser accesible para todos los usuarios objetivo.

Durante el desarrollo de nuestra aplicación web, nos encontramos con dificultades al intentar implementar pruebas unitarias. A pesar de nuestros esfuerzos, nos enfrentamos a numerosos fallos y problemas que dificultaron la ejecución de pruebas. Identificamos que una posible razón de estos contratiempos radica en la naturaleza transparente del proceso de despliegue en Glitch. Si bien esta simplicidad puede ser beneficiosa para usuarios sin conocimientos técnicos profundos, para aquellos con un perfil técnico más avanzado, puede resultar una desventaja no tener un mayor control y visibilidad sobre el proceso subyacente.

6. BIBLIOGRAFÍA

Agile Alliance. (s.f.). Agile Alliance. Recuperado el 22 de marzo de 2023, de https://www.agilealliance.org/glossary/xp/#q=~(infinite~false~filters~(postType~(~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

Agile manifiesto. (s.f.). Recuperado el 21 de marzo de 2023, de http://agilemanifesto.org/iso/es/manifesto.html

Amazon Web Services. (s.f.). Recuperado el 16 de marzo de 2023, de https://aws.amazon.com/es/cloud9/

Amazon Web Services. (s.f.). Recuperado el 20 de marzo de 2023, de https://aws.amazon.com/es/docker/

Berry, A. (16 de septiembre de 2018). Glitch Forum. Recuperado el 11 de mayo de 2023, de https://support.glitch.com/t/language-support-on-glitch-a-list/5466

Boto, C. S. (14 de septiembre de 2020). Recuperado el 21 de marzo de 2023, de Profile: https://profile.es/blog/que-es-kanban-y-como-aplicarlo-al-desarrollo-de-software/

Carbonnelle, P. (mayo de 2023). PYPL PopularitY of Programming Language index. Recuperado el 15 de mayo de 2023, de https://pypl.github.io/ODE.html

Coalla, J. L. (23 de junio de 2021). Tribayte Technologies. Recuperado el 14 de marzo de 2023, de https://tech.tribalyte.eu/blog-que-es-react

Codeanywhere. (s.f.). Recuperado el 22 de marzo de 2023, de https://codeanywhere.com/#code-editor

Codeanywhere. (s.f.). Recuperado el 21 de marzo de 2023, de https://codeanywhere.com/pricing

Dash, A. (s.f.). Recuperado el 7 de marzo de 2023, de https://anildash.com/about/

Dash, A. (6 de Diciembre de 2016). Medium. Recuperado el 22 de Febrero de 2023, de https://medium.com/@anildash/introducing-gomix-aec205c421cb#.4y5odzedw

Dash, A. (13 de Marzo de 2017). Medium. Recuperado el 23 de Febrero de 2023, de Welcome to Glitch!: https://medium.com/glitch/welcome-to-glitch-fe161d0fc39b#.5q3pk2eg9

Eleventy. (s.f.). Eleventy documentation. Recuperado el 15 de marzo de 2023, de https://www.11ty.dev/docs/

Extreme Programming: A gentle introduction. (s.f.). Recuperado el 21 de marzo de 2023, de http://www.extremeprogramming.org/

Fastify. (s.f.). Recuperado el 18 de abril de 2023, de https://www.fastify.io/

Fastify. (s.f.). Recuperado el 18 de abril de 2023, de https://www.fastify.io/ecosystem/

flexera. (2023). Flexera 2023 State of the Cloud Report. Recuperado el 16 de marzo de 2023, de https://info.flexera.com/CM-REPORT-State-of-the-Cloud?lead_source=Website%20Visitor&id=Flexera.com-PR

Foro Red Hat. (31 de Marzo de 2021). Recuperado el 21 de Febrero de 2023, de https://www.redhat.com/es/topics/edge-computing/what-is-edge-computing

Freeman, E. (2019). DevOps For Dummies. New Jerset: For Dummies. Recuperado el 23 de marzo de 2023, de https://buscaenbuja.ujaen.es/permalink/34CBUA_UJA/df6b9e/alma991004055486904994

Gauchat, J. D. (2013). El gran libro de HTML5, CSS3 y Javascript. Barcelona: Marcombo.

Glitch. (s.f.). Recuperado el 9 de marzo de 2023, de https://help.glitch.com/kb/article/73-glitch-pro/

Glitch. (18 de Diciembre de 2018). Medium. Recuperado el 23 de Febrero de 2023, de https://medium.com/glitch/welcoming-thimble-to-glitch-f2097394d6c2

Glitch. (s.f.). Glitch. Recuperado el 22 de Febrero de 2023, de https://glitch.com/about

Glitch. (s.f.). Glitch Forum. Recuperado el 20 de Marzo de 2023, de https://glitch.com/faq/#what-kind-of-hosting-environment-does-glitch-use

Glitch. (s.f.). Glitch Support Center. Recuperado el 7 de marzo de 2023, de https://help.glitch.com/kb/article/17-technical-restrictions

Glitch. (s.f.). Glitch Support Center. Recuperado el 11 de mayo de 2023, de https://glitch.happyfox.com/kb/article/24-supported-languages-frameworks/

Glitch. (s.f.). Glitch Support Center. Recuperado el 12 de mayo de 2023, de https://help.glitch.com/kb/article/9-adding-a-custom-domain/

Hermitte, B. (30 de marzo de 2021). Recuperado el 10 de abril de 2023, de https://www.wimi-teamwork.com/es/blog/el-metodo-moscow-priorizacion-simple-de-tareas-en-un-proyecto/

Internet Live Stats. (s.f.). Internet Live Stats. Recuperado el 13 de Febrero de 2023, de https://www.internetlivestats.com/total-number-of-websites/

JSFiddle. (s.f.). JSFiddle docs. Recuperado el 15 de mayo de 2023, de https://docs.jsfiddle.net/use-cases/demos-for-products-or-libraries

Kastrenakes, J. (22 de mayo de 2020). The Verge. Recuperado el 23 de febrero de 2023, de https://www.theverge.com/2020/5/22/21268007/glitch-layoffs-substantial-number-coding-platform-union

Lardinois, F. (15 de marzo de 2018). Techcrunch. Recuperado el 20 de marzo de 2023, de https://techcrunch.com/2018/03/15/repl-it-lets-you-program-in-your-browser/

Luján-Mora, S. (2002). Programación de aplicaciones web: historia, principios básicos y clientes web. Editorial Club Universitario.

McEwen, M. (15 de abril de 2020). Dev. Recuperado el 23 de febrero de 2023, de https://dev.to/glitch/boosted-apps-glitch-apps-with-more-power-26bk

Microsoft. (s.f.). Azure. Recuperado el 23 de marzo de 2023, de https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops#tabxd9ea792152394bca8ee6b0708ec7b0c4

Miniwatts Marketing Group . (s.f.). Internet World Stats. Recuperado el 12 de Febrero de 2023, de https://www.internetworldstats.com/emarketing.htm

Mozilla Development Network . (s.f.). Mozilla Web Documents. Recuperado el 8 de mayo de 2023, de https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#time_complexity

Mozilla. (s.f.). mdn web docs. Recuperado el 14 de marzo de 2023, de https://developer.mozilla.org/es/docs/Web/CSS

Mozilla. (s.f.). mdn web docs. Recuperado el 14 de marzo de 2023, de https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

OpenJS Foundation. (s.f.). Node.js docs. Recuperado el 8 de mayo de 2023, de https://nodejs.org/en/docs/es6

Planview. (s.f.). Recuperado el 21 de marzo de 2023, de https://www.planview.com/resources/guide/introduction-to-kanban/

React. (s.f.). Recuperado el 14 de marzo de 2023, de https://es.reactjs.org/

Red Hat. (19 de julio de 2022). Recuperado el 21 de marzo de 2023, de https://www.redhat.com/es/devops/what-is-agile-methodology#los-valores-de-la-metodolog%C3%ADa-%C3%A1gil

Red Hat. (31 de Enero de 2023). Recuperado el 20 de marzo de 2023, de https://www.redhat.com/es/topics/virtualization/what-is-a-virtual-machine

Replit. (s.f.). Replit Personals plans. Recuperado el 20 de marzo de 2023, de https://replit.com/pricing

Sharma, L. (19 de mayo de 2022). Fastly. Recuperado el 23 de febrero de 2023, de https://www.fastly.com/es/blog/fastly-announces-acquisition-of-glitch-a-future-of-yes-code-at-the-edge

Sourceforge. (11 de septiembre de 2002). Recuperado el 18 de abril de 2023, de https://bcrypt.sourceforge.net/

Spolsky, J. (31 de Mayo de 2016). Joel on software . Recuperado el 2 de Febrero de 2023, de https://www.joelonsoftware.com/2016/05/31/introducing-hyperdev/

Spolsky, J. (s.f.). Joel On Software. Recuperado el 7 de marzo de 2023, de https://www.joelonsoftware.com/about-me/

SQLite. (s.f.). About SQLite. Recuperado el 15 de marzo de 2023, de https://www.sqlite.org/about.html

State Of Agile. (2022). Recuperado el 21 de marzo de 2023, de https://drive.google.com/file/d/1MPEXs2rVOXqTituphel4kCuY9iuk3ee5/view

V8. (28 de septiembre de 2018). V8 blog. Recuperado el 8 de mayo de 2023, de https://v8.dev/blog/array-sort

Wikipedia. (s.f.). Recuperado el 16 de marzo de 2023, de https://en.wikipedia.org/wiki/Cloud9_IDE

Wikipedia. (s.f.). Recuperado el 21 de marzo de 2023, de https://en.wikipedia.org/wiki/Codeanywhere

Wikipedia. (s.f.). Recuperado el 21 de marzo de 2023, de https://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema

Wikipedia. (s.f.). Wikipedia. Recuperado el 14 de marzo de 2023, de https://es.wikipedia.org/wiki/HTML

Wikipedia. (s.f.). Wikipedia. Recuperado el 20 de marzo de 2023, de https://es.wikipedia.org/wiki/Replit

WIkipedia. (s.f.). Wikipedia. Recuperado el 14 de marzo de 2023, de https://es.wikipedia.org/wiki/JavaScript

7. ANEXOS

7.1 Conexión con la API de Google

Para poder acceder a los ficheros que los usuarios de la aplicación tienen alojados en Google Drive es necesario que nuestra aplicación se conecte con la API de Google. El proceso que se describe a continuación se puede encontrar en código en el fichero *googleAPI.js*.

El primer paso es entrar en Google Cloud y crear un proyecto, en el que se habilite la API de Google Drive, y una cuenta de servicio que permitirá la identificación de nuestro proyecto. Además, exportaremos las claves de la cuenta de servicio en un JSON que descargaremos en nuestro equipo local.

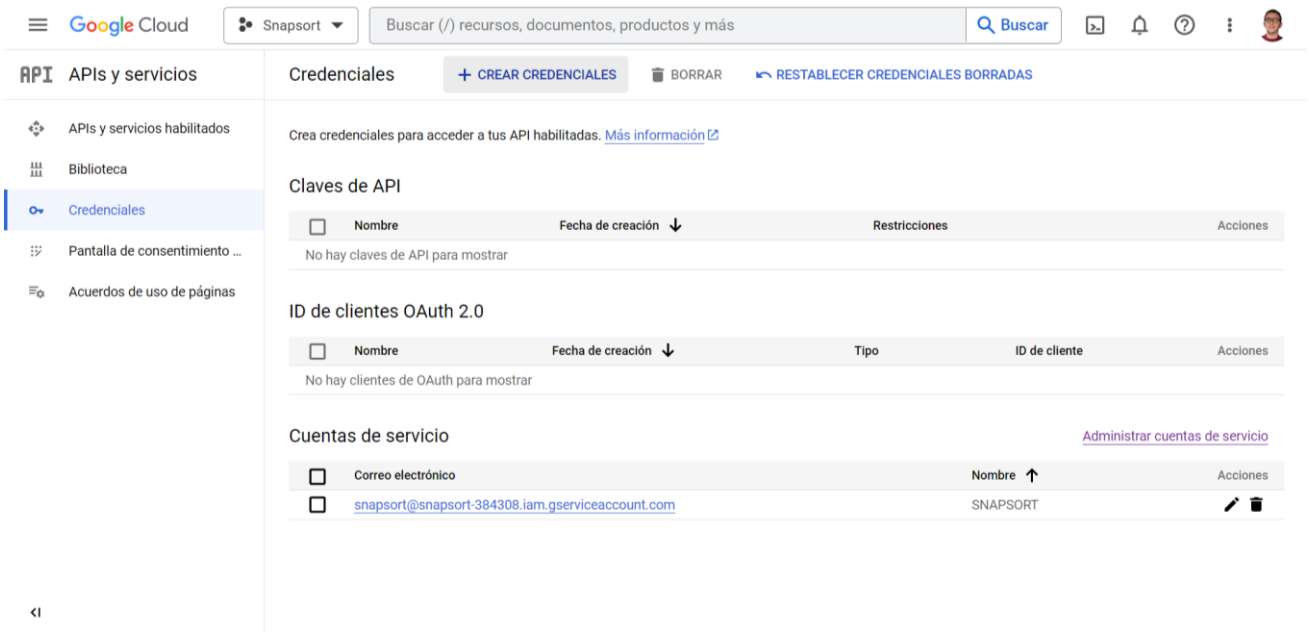

Ilustración 53. Proyecto y credenciales en Google Cloud Platform

Como hemos comentado anteriormente, los proyectos gratuitos de Glitch son públicos, por lo que subir nuestro fichero con las credenciales sería exponerlo, para evitar este problema definiremos una variable de entorno por cada uno de los campos del archivo JSON con las credenciales, ya que el fichero *.env* es el único cuya privacidad se reserva para los miembros del proyecto. En el código construiremos un objeto JSON con las credenciales, llamando a cada una de las variables de entorno previamente definidas.

```
const credentials = {
  type: process.env.type,
  project_id: process.env.project_id,
  private_key_id: process.env.private_key_id,
  private_key: process.env.private_key.replace(/\\n/g, '\n'),
  client_email: process.env.client_email,
  client_id: process.env.client_id,
  auth_uri: process.env.auth_uri,
  token_uri: process.env.token_uri,
  auth_provider_x509_cert_url: process.env.auth_provider_x509_cert_url,
  client_x509_cert_url: process.env.client_x509_cert_url
}
```

Con dicho objeto podremos identificar nuestra aplicación a través de la API, en nuestro caso, en la API de Google Drive. El siguiente paso será construir un cliente de la API a Drive con la dirección IP del usuario final para el que se realiza la llamada a la API, de forma que los límites de consulta se apliquen a cada uno de los usuarios finales y no a la aplicación.

```
const auth = new google.auth.GoogleAuth({
  credentials: (credentials),
  scopes: ["https://www.googleapis.com/auth/drive"],
});

const drive = google.drive({
  version: "v3",
  auth,
  params: { userIp: clientIp },
});
```

Tras esto nuestra aplicación podrá llamar cuando lo necesite al método que lista los elementos de una carpeta de drive `drive.files.list()` obteniendo el nombre de cada uno de los archivos, su identificador y el enlace de acceso al mismo para finalmente ser almacenados en la base de datos.

```
  const res = await drive.files.list({
    q: `'${folderId}' in parents and trashed = false`,
    fields: "files(id, name, webContentLink, webViewLink)",
    pageSize: 1000,
  });
```

7.2 Manual de usuario

Mediante el siguiente enlace, se puede acceder a un video explicativo para el usuario, en él se ejemplifica como utilizar a aplicación SnapSort desarrollada para este TFG.

Ilustración 54. Código QR al video demostrativo de SnapSort (redirige a youtu.be/1iwTEGFWqJc)

7.3 Video Ilustrativo del proceso de Creación de aplicaciones

A través del siguiente enlace, encontraremos un video en el que se ejemplifica el uso de Glitch para la creación de una aplicación web. En dicho video también se muestra el funcionamiento de las principales funcionalidades de la herramienta.

Ilustración 55. Código QR al video de demostrativo de Glitch (redirige a youtu.be/YKKljk1hKw0)

7.4 Vistas en la versión Móvil

El uso de Bootstrap 5 ha sido fundamental para el desarrollo de vistas responsivas. Esta herramienta de diseño web ofrece una amplia variedad de estilos, componentes y funcionalidades que han permitido crear interfaces de usuario adaptables a diferentes tamaños de pantalla. Gracias a las clases predefinidas de Bootstrap 5, he podido diseñar vistas para dispositivos móviles y de escritorio de manera rápida y eficiente.

Finalmente, nuestra aplicación puede ser utilizada desde un dispositivo móvil gracias a unas pequeñas adaptaciones en el esquema lógico de la interfaz que permiten una navegación sencilla y cómoda desde pantallas más pequeñas. De esta forma, los usuarios podrán acceder a la aplicación desde

sus dispositivos móviles sin experimentar dificultades en la visualización o en la interacción con la misma.

La herramienta Grid de Bootstrap ha sido de gran utilidad en el desarrollo de la interfaz gráfica de nuestra aplicación. Esta herramienta permite no solo dividir el espacio en filas y columnas, sino que también se adapta de forma automática al tamaño de la pantalla del dispositivo utilizado, lo que asegura una correcta visualización y uso de la aplicación en cualquier dispositivo. Además, la flexibilidad del sistema Grid ha permitido una fácil reorganización de los elementos de la interfaz en función de las necesidades del usuario.

Precedemos a comentar las adaptaciones que han sufrido las vistas.

Ilustración 56. Vista index en la versión móvil

Como se puede observar en la *Ilustración 56* el cuadro sobre el que se engloban los elementos de la vista ocupa un mayor porcentaje del ancho de la pantalla que en la versión de ordenador. Además, la imagen de fondo se recorta de forma automática para adaptase a la relación de aspecto.

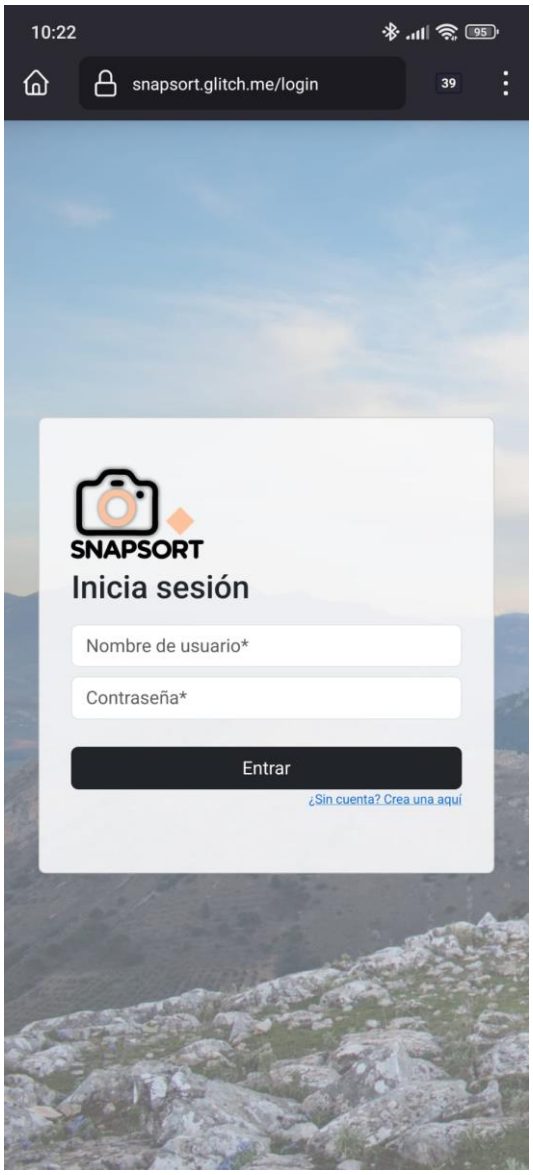

Ilustración 57. Vista login en la versión móvil

La principal adaptación en la vista de inicio de sesión es que el botón de confirmar los datos y en enlace que redirecciona a la pantalla de registro pasan de ocupar una misma fila, a ocupar una fila cada uno ocupando todo el ancho disponible, como puede comprobarse en la *Ilustración 57*. De forma similar se comporta la Vista de registro en la versión móvil, en la que además cada campo rellenable ocupa una única fila para facilitar la lectura.

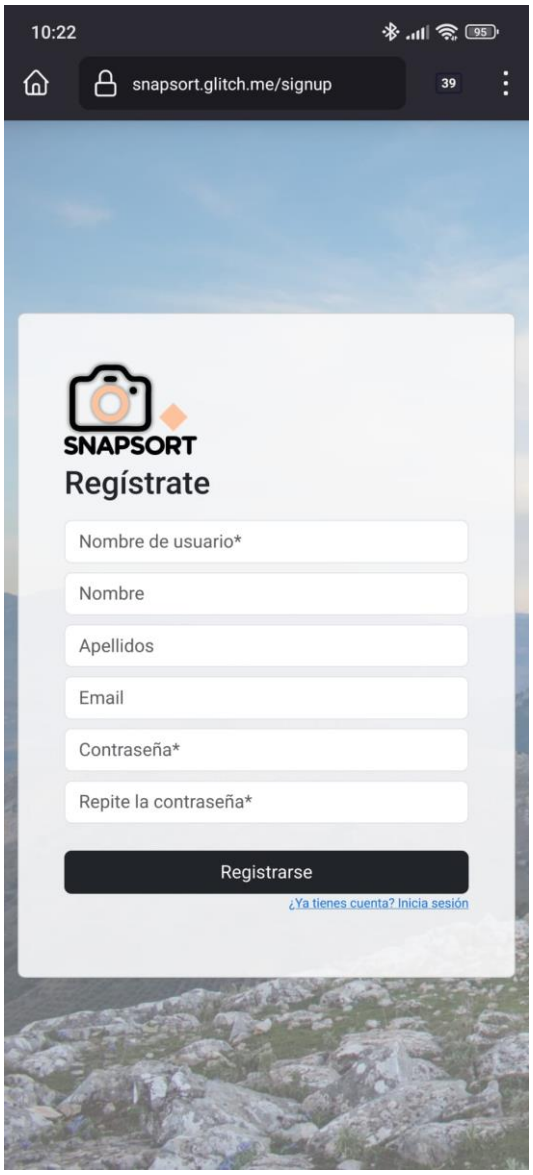

Ilustración 58. Vista de registro en la versión móvil

En la vista inicial de edición de imágenes (ver *Ilustración 59*), la información se muestra en la parte inferior en la vista móvil, facilitando que la imagen propia ocupe el ancho de pantalla del dispositivo.

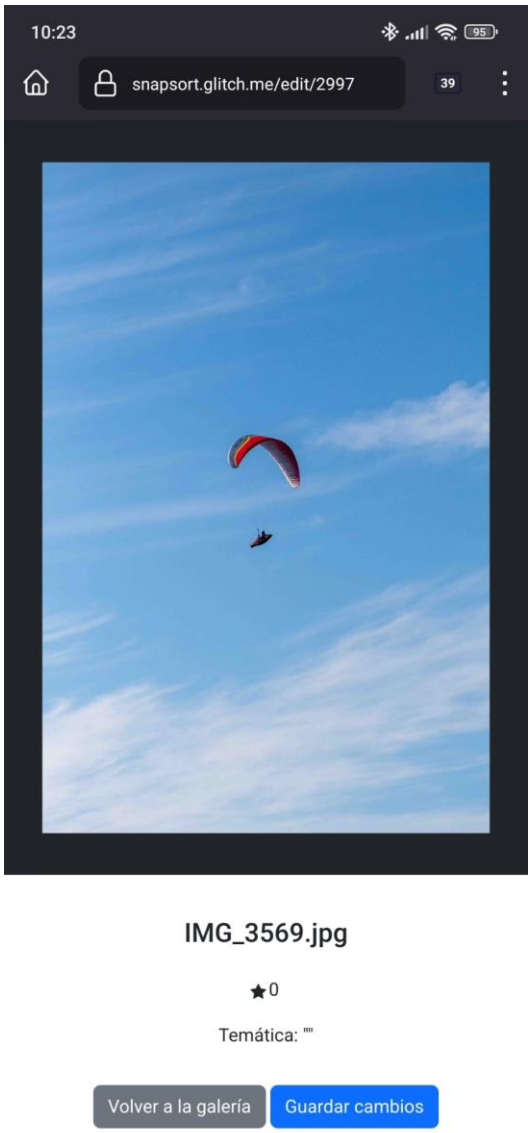

Ilustración 59. Vista de edición en la versión móvil

Finalmente, la pantalla *home* quedaría organizada en una única columna permitiendo la correcta visualización de las imágenes, y el modal para añadir una nueva importación ocuparía casi el total del ancho de la pantalla.

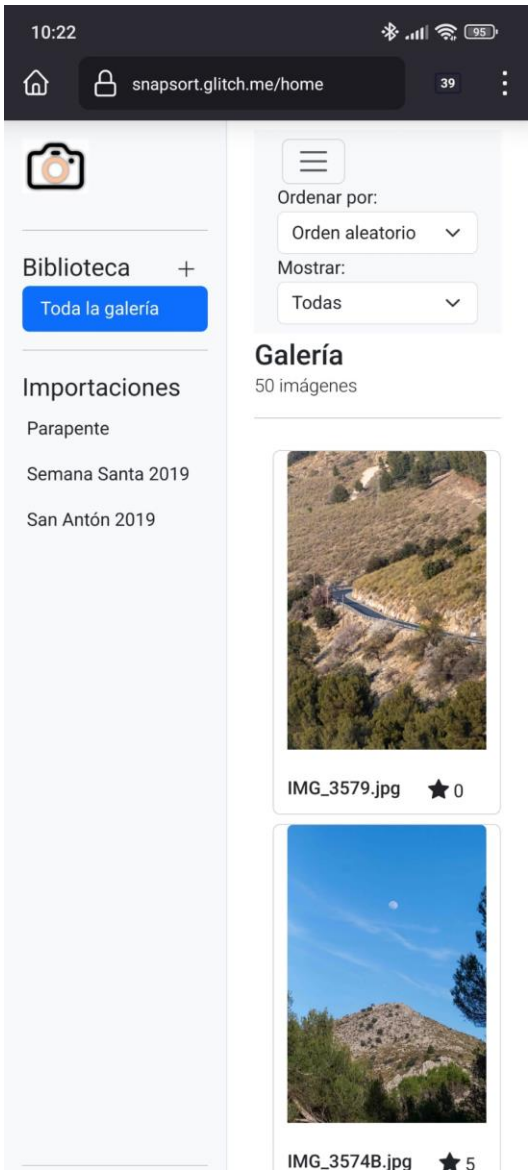

Ilustración 60. Vista home en la versión móvil

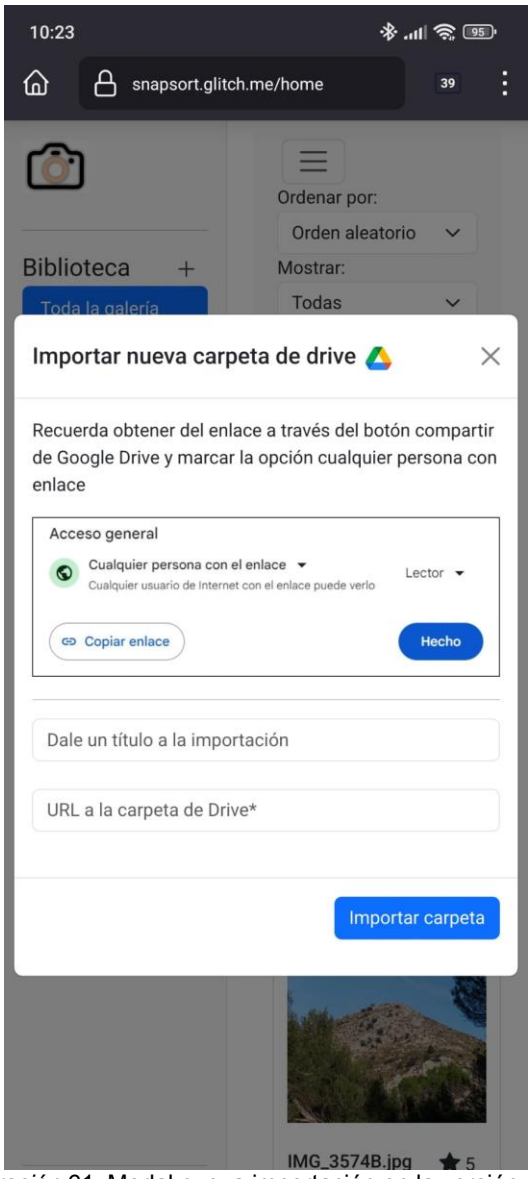

Ilustración 61. Modal nueva importación en la versión móvil