



**Universidad  
Zaragoza**

Trabajo Fin de Grado

Una wiki para regiones delimitadas sobre una malla global discreta

Autor

Diego Raúl Roldán Urueña

Director

Rubén Béjar Hernández

Grado en Ingeniería Informática

Escuela de Ingeniería y Arquitectura

Julio 2025

## 1 Resumen

Obligatorio según la normativa (revisar). Puede ser algo como Introducción + Conclusiones condensadas en, pongamos, 300 palabras.

## 2 Tabla de contenidos

### Contenido

1	Resumen.....	2
2	Tabla de contenidos.....	2
2.1	Tabla de figuras.....	3
2.2	Tabla de tablas.....	3
3	Introducción .....	5
3.1	Contexto.....	5
3.2	Fuentes de datos integrados / mostrados / procesados .....	6
4	Análisis del problema .....	7
4.1	Requisitos / Historias de Usuario / Casos de Uso.....	7
4.2	Casos de uso.....	10
4.2.1	Ver un artículo.....	10
4.2.2	Ver versiones anteriores de un artículo .....	10
4.2.3	Iniciar Sesión .....	11
4.2.4	Crear un artículo nuevo .....	11
4.2.5	Crear un artículo a partir de otro .....	11
4.2.6	Editar un articulo .....	12
4.3	Interfaces de usuario.....	13
4.4	Interfaces para la integración en aplicaciones .....	16
5	Diseño de la solución .....	18
5.1	Arquitectura.....	18
5.1.1	Contexto.....	19
5.1.2	Modelo de datos .....	19
5.1.3	Paquetes .....	20
5.1.4	Clases.....	22
5.1.5	Componentes y conectores .....	23
5.1.6	Distribución.....	24
5.2	Implementación.....	24
5.2.1	DGGS .....	26

5.2.2	¿Por qué usar un DGGS?	27
5.2.3	Elección de DGGS	28
5.3	GUI Actual	30
5.4	Procesamiento de datos	33
5.5	Pruebas	33
5.5.1	Pruebas extremo-a-extremo	34
5.5.2	Prueba de estrés	35
6	Gestión del proyecto	37
6.1	Fases del desarrollo	37
6.1.1	Análisis y diseño técnico	37
6.1.2	Desarrollo iterativo de funcionalidades	37
6.1.3	Pruebas, validación y control de calidad	38
6.1.4	Documentación, análisis de rendimiento y entrega	38
6.2	Integración y despliegue continuo (CI/CD)	39
6.3	Estimación de esfuerzo y gestión del tiempo	39
6.4	Análisis de riesgos	40
7	Conclusiones	41
8	Bibliografía	42
9	Anexos	43

## 2.1 Tabla de figuras

Figura 1.	Página principal de Locatopedia	7
Figura 3.	Locatopedia en ordenador	13
Figura 4.	Locatopedia en móvil	14
Figura 5.	Locatopedia en tableta	14
Figura 6.	Primera versión del mapa de navegación de la aplicación	15
Figura 7.	Diagrama Entidad-Relación	19
Figura 8.	Diagrama de paquetes	21
Figura 9.	Diagrama de clases	22
Figura 10.	Diagrama de componentes principales	23
Figura 11.	Diagrama de secuencia	24
Figura 12.	Mapa de navegación	31

## 2.2 Tabla de tablas

Tabla 1.	Requisitos funcionales de la aplicación	9
Tabla 2.	Requisitos no funcionales	10
Tabla 3.	Primeras ideas	¡Error! Marcador no definido.
Tabla 4.	Comparativa de funcionalidades de distintos tipos de usuarios	10

Diego Raúl Roldán Urueña

Tabla 5. Resultados test de estrés.....	36
---	----

### 3 Introducción

Uno o dos párrafos de introducción al trabajo: brevemente qué se quería hacer, y cuáles han sido los principales resultados alcanzados.

El resto del documento se estructura como sigue: explicar la estructura del documento. Puede ser buena idea señalar que se ha cumplido explícitamente con la recomendación de 20 páginas/10.000 palabras, o explicar explícitamente por qué no se ha hecho.

Este documento se basa en el desarrollo de una aplicación web. Se ha creado un sitio al estilo 'wiki' donde cada artículo se identifica por medio de un lugar o un conjunto de lugares. Cada lugar, es una celda definida sobre una malla global discreta (DGG). De esta forma, cada conjunto de celdas recibe un identificador único que será el que identifique cada artículo y se usa para asignarle un PURL (URL permanente).

El objetivo principal es que cualquier usuario pueda establecer y compartir fácilmente áreas de su interés con descripciones, que pueden incluir imágenes, tablas, enlaces a otras fuentes de datos de interés relacionado, etc. También es interesante que distintos usuarios colaboren para mejorar, completar o extender la información publicada.

Estás regiones pueden no tener vocación ni necesidad para ser permanentes, por ello se permite borrar artículos una vez ya estén creados.

#### 3.1 Contexto

Poner en contexto este TFG: ¿cómo encaja en el Mundo? ¿Es algo innovador o hay cosas similares en el mercado? Si hay cosas similares, ¿cómo se compara esta TFG con ellas? ¿Se hace en una empresa, en un grupo de investigación o es un proyecto independiente? ¿Se plantea como un primer paso hacia algo mayor o como un trabajo por sí mismo? ¿O es una mejora de algo existente? ¿Es una parte de algo que se ha hecho trabajando con más gente o es el resultado de un trabajo individual? Si es parte de un producto/sistema más grande, o una mejora o nueva versión, habrá que esforzarse en la memoria, y luego en la presentación, por trazar límites claros entre lo que había y lo que se ha hecho.

Locatopedia es una idea innovadora. Encontramos en el mercado multitud de proyectos relacionados, también, con un sitio web estilo wiki y geografía.

El primero de ellos y más conocido es Wikipedia. Esta aplicación web es una enciclopedia editada de manera colaborativa. Entre otras cosas, se permite crear y editar artículos, al igual que en Locatopedia, pero cada artículo se identifica mediante el título del propio artículo. La diferencia la encontramos en la forma de identificar cada artículo, en Wikipedia se hace mediante el título, y en Locatopedia se hace mediante un identificador único de un conjunto de celdas de una malla global discreta superpuesta sobre el mapa.

Otro sitio similar es OpenStreetMap, donde se pueden añadir lugares como carreteras, accidentes geográficos, negocios, tendidos eléctricos, etc. directamente sobre el mapa. En cierta forma es similar a Locatopedia, pero la gran diferencia, frente a OpenStreetMap, es la capacidad de poder escribir el

Diego Raúl Roldán Urueña

contenido que el usuario considere relevante, añadir imágenes, enlaces a otras fuentes de información directamente en cada región.

Wikimapia es el sitio más parecido al desarrollado en este proyecto. Los dos softwares permiten al usuario explorar lugares creados por otros, editarlos o completarlos. Wikimapia basa la localización de regiones con localizaciones exactas (usando coordenadas), y esa es la mayor diferencia con respecto a Locatopedia, que usa un conjunto de celdas de una malla global discreta superpuesta sobre el mapa.

Aunque haya sitios similares a Locatopedia, este ofrece algo distinto. Este proyecto introduce un enfoque basado en mallas globales discretas (DGGs), creando identificadores únicos para cada conjunto de celdas posible sobre el planeta. Locatopedia también permite a los usuarios crear nuevos artículos o modificar otros creados por otros usuarios.

Este proyecto se ha desarrollado de manera independiente, sin formar parte de un grupo de investigación o empresa. Sin embargo, su diseño modular y su arquitectura escalable permiten que puedan integrarse otros sistemas dentro de Locatopedia y evolucionar hacia un proyecto más grande en el futuro.

Ya sabemos que los límites de las aplicaciones están en nuestra imaginación. La aplicación hasta ahora solo es la base de lo que podría ser un gran proyecto. Se puede seguir añadiendo multitud de nuevas características, por ejemplo, categorizar los artículos en, por ejemplo, Edificios, Tiendas, Hoteles, Hospitales, Ríos, Montañas, etc. Otro posible ejemplo podría ser añadir un mapa donde se pudiesen ver y explorar todas las regiones en la aplicación directamente sobre el plano. También podría añadirse una sección de discusión en cada artículo, de una forma similar a como lo hace Wikipedia.

### 3.2 Fuentes de datos integrados / mostrados / procesados

Si es un proyecto que integra, muestra, procesa... datos, hablar de las fuentes de los mismos. De dónde salen, qué licencias tienen, cuánto ha habido que procesarlos... Si el proceso de los datos ha sido una parte importante del trabajo, habrá que contarlo en detalle en la sección 3. Si el proyecto usa datos, pero el procesado no se ha hecho como parte del trabajo, entonces basta con contarlo aquí.

## 4 Análisis del problema

Explicar brevemente, de manera general, lo que se va a hacer.

Se desarrollará una aplicación wiki en la que cada página se refiera a un lugar distinto.

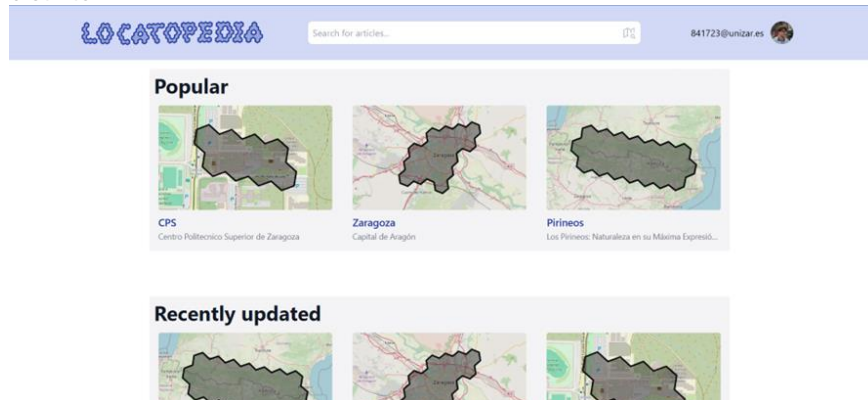


Figura 1. Página principal de Locatopedia.

Cada región geográfica se identificará por medio de unas celdas de una malla global discreta superpuesta sobre el mapa. Cada conjunto de celdas tendrá asociado un identificador único que servirá para identificar el artículo que trate este conjunto de celdas. Aparte de esto, se desarrollará un sistema de sesión, para que los usuarios puedan guardar sus artículos favoritos o acceder rápidamente a los artículos creados o editados por ellos. Para poder crear o editar artículos será necesario iniciar sesión aportando información personal, de esta forma se aplicará un control de las versiones evitando que usuarios hagan un uso incorrecto de la aplicación.


Los usuarios también tendrán la posibilidad de realizar búsquedas de artículos tanto por texto como geográficamente. Al introducir un texto en el buscador, el servidor seleccionará coincidencias exactas entre el texto buscado y todos los títulos y subtítulos de artículos en la aplicación. Además, habilitará un mapa que permitirá búsqueda geográfica. En este caso, se añadirán puntos en el mapa que crearán un polígono que indica el área de búsqueda. Todos los artículos que, de manera parcial, o completa, estén incluidos en esta región, se mostrarán como resultados de la búsqueda. Los dos tipos de búsqueda se pueden combinar, es decir, se puede buscar artículos con un título o subtítulo determinado en una región determinada por el usuario.

### 4.1 Requisitos y Casos de Uso

Se pueden combinar requisitos y casos de uso. No se suelen hacer otras combinaciones (p.ej. no tiene sentido tener requisitos e historias de usuario, dado que ambas cosas sirven para lo mismo). Los requisitos pueden dividirse en funcionales y no funcionales, pero esta distinción es a veces un poco arbitraria. Si hay requisitos estrictos de prestaciones o de seguridad,

generalmente serán no funcionales, describirlos lo más objetiva y cuantitativamente posible.

La Tabla 1 describe los requisitos funcionales del sistema, es decir, las funcionalidades que debe ofrecer desde el punto de vista del usuario y del comportamiento esperado. Cada requisito está identificado con un código único, una descripción clara y, en algunos casos, condiciones específicas para su cumplimiento.

Requisito	Descripción
RF 1	El sistema debe permitir a los usuarios ver el contenido de artículos. Se debe permitir que el usuario navegue por el mapa y vea seleccionada la región sobre la que trata el artículo, y también debe poder ver el contenido del artículo.
RF 2	El sistema debe permitir a los usuarios ver el contenido de todas versiones anteriores de artículos.
RF 3	El sistema debe mostrar una página principal con los artículos más populares del momento, los artículos recientemente creados y editados y un artículo destacado.
RF 4	El sistema debe permitir a los usuarios realizar búsquedas de texto para encontrar artículos que contengan ese texto en su título o subtítulo.
RF 5	El sistema debe permitir a los usuarios realizar búsquedas geográficas para encontrar artículos que estén contenidos, de manera parcial o completa, en la región seleccionada por el usuario.
RF 6	El sistema debe permitir a los usuarios registrarse e iniciar sesión con su cuenta de Google.
RF 7	El sistema debe permitir a los usuarios registrados acceder a la información de su cuenta donde se encontrarán los artículos marcados como favoritos y artículos creados y modificados por el usuario.
RF 8	El sistema debe permitir a los usuarios registrados marcar y desmarcar como favoritos artículos.
RF 9	El sistema debe permitir a los usuarios registrados crear artículos añadiendo una región sobre la que trata el artículo, título, subtítulo y contenido. 



RF 10	El sistema debe permitir a los usuarios registrados crear artículos a partir de uno ya creado. Se usará como plantilla el artículo ya creado, y el usuario deberá modificar la región sobre la que trata el artículo, y podrá modificar título, subtítulo y contenido del artículo ya existente.
RF 11	El sistema debe permitir a los usuarios registrados editar artículos ya creados por cualquier usuario, modificando el título, subtítulo y/o contenido del artículo.
RF 12	El sistema debe permitir a los usuarios que estén creando o editando el contenido de un artículo, añadir formato al texto (negrita, cursiva, etc.), imágenes, tablas, enlaces, encabezados. Markdown es una opción válida para esto.
RF 13	El sistema debe permitir a los usuarios que estén creando o editando el contenido de un artículo, ver una previsualización del contenido del artículo.
RF 14	El sistema debe permitir a los usuarios registrados que hayan creado un artículo, eliminar dicho artículo.
RF 15	El sistema debe permitir a los usuarios administradores eliminar cualquier artículo del sistema.

Tabla 1. Requisitos funcionales de la aplicación.

A continuación, en la Tabla 2 se presentan los requisitos no funcionales, que definen las propiedades de calidad del sistema, como rendimiento, usabilidad, compatibilidad o seguridad. Estos requisitos complementan a los funcionales y establecen restricciones y estándares que debe cumplir la implementación.

Requisito	Descripción
RNF 1	La interfaz gráfica será responsiva y se adaptará correctamente a distintos tamaños de pantalla, siguiendo las pautas de diseño de Google Material Design y las recomendaciones de accesibilidad del estándar WCAG 2.1.
RNF 2	El sistema usará los mapas de OpenStreetMap.
RNF 3	El sistema será compatible con Google Chrome, Firefox y Safari.
RNF 4	La aplicación web debe usar HTTPS y el dominio <i>locatopedia.com</i> .
RNF 5	El sistema debe de respetar la privacidad de los usuarios y no publicar sus datos privados, salvo dirección de correo.
RNF 6	El sistema debe almacenar las imágenes en el servidor en formato webp, comprimidas.
RNF 7	El sistema debe permitir a los usuarios expandir cualquier mapa en la aplicación para que ocupe las dimensiones completas de la pantalla.
RNF 8	La división geográfica en celdas se hará utilizando una DGGS (Discrete Global Grid System). Se utilizará H3, desarrollada por Uber. Esta malla global discreta divide la superficie terrestre en hexágonos (y algunos pentágonos). Cada celda tiene un

Comentado [DR1]: <https://m3.material.io/>

Comentado [DR2]: Eliminar o no

	identificador único y puede subdividirse en celdas más pequeñas en niveles de mayor resolución.
--	---

Tabla 2. Requisitos no funcionales.

La Tabla 3 compara las funcionalidades disponibles para los distintos tipos de usuarios del sistema: no registrados, registrados y administradores. Esta comparativa permite visualizar de forma resumida los distintos niveles de acceso y permisos según el rol del usuario.

	Usuario No Registrado	Usuario Registrado	Usuario Administrador
Buscar artículos	Si	Si	Si
Ver artículos	Si	Si	Si
Ver versiones anteriores de artículos	Si	Si	Si
Marcar artículos como favoritos	No	Si	Si
Modificar artículos	No	Si, cualquier articulo	Si, cualquier articulo
Crear artículos	No	Si	Si
Eliminar artículos	No	Solo creados por dicho usuario	Si, cualquier articulo

Tabla 3. Comparativa de funcionalidades de distintos tipos de usuarios.

## 4.2 Casos de uso

### 4.2.1 Ver un artículo.

- Descripción: Este caso de uso permite a los usuarios de Locatopedia ver el contenido de un artículo.
- Flujo de Eventos Principal:
  1. El usuario encuentra un artículo en la página Home o de su cuenta, busca un artículo en la barra de búsqueda o con el mapa o entra en un enlace a un artículo de Locatopedia.
  2. El usuario ve el título, subtítulo, mapa y contenido. Puede deslizar hacia abajo la página y ver el contenido completo del artículo.
- Flujo de Eventos Alternativo:
  - El usuario puede deslizarse por el mapa viendo partes que no forman parte del conjunto de celdas al que el artículo se refiere.
  - El usuario puede hacer clic en el corazón para marcar este articulo como favorito. O eliminarlo de favoritos si ya estaba marcado como favorito.

### 4.2.2 Ver versiones anteriores de un artículo

- Descripción: Este caso de uso permite a los usuarios de Locatopedia ver el contenido de una versión pasada de un artículo.
- Flujo de Eventos Principal:

Diego Raúl Roldán Urueña

1. El usuario encuentra un artículo en la página Home o de su cuenta, busca un artículo en la barra de búsqueda o con el mapa o entra en un enlace a un artículo de Locatopedia.
2. El usuario desliza abajo y selecciona la opción de 'See other versions of this page'.
3. El usuario selecciona una versión.
4. El usuario ve el título, subtítulo y contenido de la versión seleccionada.

#### 4.2.3 Iniciar Sesión

- Descripción: Este caso de uso permite a los usuarios de Locatopedia iniciar sesión con su cuenta de Google.
- Flujo de Eventos Principal:
  1. El usuario pulsa el botón de 'Sign Up/Sign In'.
  2. El usuario selecciona la opción de 'Continue with Google'.
  3. El usuario inicia sesión en Google y otorga los permisos requeridos por Locatopedia.
- Flujo de Eventos Alternativo
  - El usuario puede seleccionar la opción de cancelar. En este caso volvería a la página Home sin iniciar sesión.

#### 4.2.4 Crear un artículo nuevo

- Descripción: Este caso de uso permite a los usuarios con sesión iniciada en Locatopedia crear un artículo nuevo en el sistema.
- Flujo de Eventos Principal:
  1. El usuario entra en la página personal.
  2. El usuario selecciona la opción de 'Create new article'.
  3. El usuario decide un título, subtítulo y contenido para el artículo a crear.
  4. El usuario selecciona un conjunto de celdas sobre las que tratará su artículo.
  5. El sistema comprueba que los campos de título, subtítulo y contenido no estén vacíos. También comprueba que la selección de celdas no sea ni nula ni este ya registrada en el sistema. En caso correcto, se habilita el botón de 'publish', que publica el artículo.
  6. El usuario selecciona la opción de 'publish' para publicar el artículo.
  7. El usuario confirma su elección y publica el artículo.
- Flujo de Eventos Alternativo
  - Si el usuario no tiene una sesión válida, deberá iniciar sesión previamente. Ya explicado en punto 2.
  - Si el usuario selecciona la opción de cancelar, volverá a la página personal.
  - En el paso 7, si el usuario decide no confirmar, se volverá al paso 5.

#### 4.2.5 Crear un artículo a partir de otro

- Descripción: Este caso de uso permite a los usuarios con sesión iniciada en Locatopedia crear un artículo nuevo en el sistema a partir de otro.

- Flujo de Eventos Principal:
  1. El usuario entra en el artículo que quiere usar como plantilla para el nuevo artículo.
  2. El usuario selecciona la opción de 'create similar'.
  3. El usuario decide un título, subtítulo y contenido para el artículo a crear.
  4. El usuario selecciona un conjunto de celdas sobre las que tratará su artículo.
  5. El sistema comprueba que los campos de título, subtítulo y contenido no estén vacíos. También comprueba que la selección de celdas no sea ni nula ni esté ya registrada en el sistema. En caso correcto, se habilita el botón de 'publish', que publica el artículo.
  6. El usuario selecciona la opción de 'publish' para publicar el artículo.
  7. El usuario confirma su elección y publica el artículo.
- Flujo de Eventos Alternativo
  - Si el usuario no tiene una sesión válida, deberá iniciar sesión previamente. Ya explicado en punto 2.
  - Si el usuario selecciona la opción de cancelar, volverá a la página personal.
  - En el paso 7, si el usuario decide no confirmar, se volverá al paso 5.

#### 4.2.6 Editar un artículo

- Descripción: Este caso de uso permite a los usuarios con sesión iniciada en Locatopedia editar un artículo ya existente en el sistema.
- Flujo de Eventos Principal:
  1. El usuario entra en el artículo a editar.
  2. El usuario selecciona la opción de 'edit page'.
  3. El usuario decide un nuevo título, subtítulo y/o contenido para el artículo a editar.
  4. El sistema comprueba que los campos de título, subtítulo y contenido no estén vacíos y al menos se haya modificado uno de ellos. En caso correcto, se habilita el botón de 'save changes', que publica el artículo.
  5. El usuario selecciona la opción de 'save changes' para publicar el artículo.
  6. El usuario confirma su elección y modifica el artículo.
- Flujo de Eventos Alternativo
  - Si el usuario no tiene una sesión válida, deberá iniciar sesión previamente. Ya explicado en punto 2.
  - Si el usuario selecciona la opción de cancelar, volverá a la página anterior.
  - En el paso 6, si el usuario decide no confirmar, se volverá al paso 4.

Diego Raúl Roldán Urueña

### 4.3 Interfaces de usuario

Para productos con GUI, orientados principalmente a humanos. Desde el punto de vista de la captura de requisitos: borradores, bocetos, navegación entre pantallas etc. No la GUI tal y como queda finalmente, que se contaría en la sección 3.

Locatopedia es una aplicación web, por lo tanto, debe tener una buena interfaz de usuario. La idea principal del sitio es que esté disponible y funcione correctamente en ordenador. De todas maneras, se optimizará el diseño para cualquier tipo de dispositivo, incluido teléfonos móviles y tabletas. En las Figura 2, Figura 3 y Figura 4, se muestra la GUI final en distintos dispositivos:

**Comentado [DR3]:** No tengo claro que esto vaya aquí

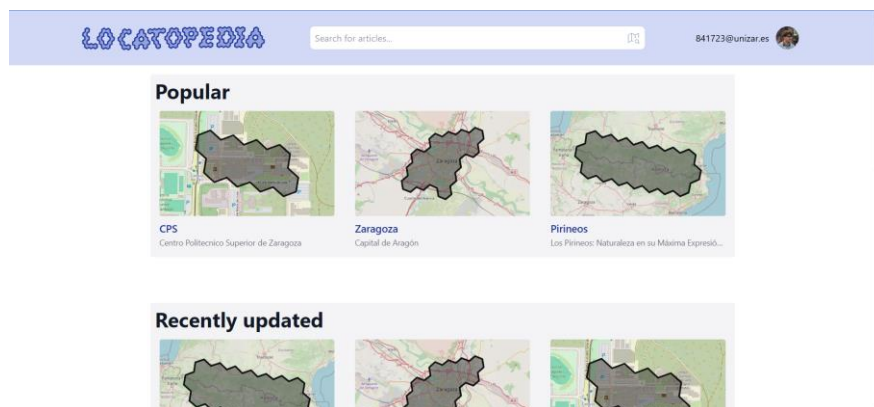


Figura 2. Locatopedia en ordenador.

Diego Raúl Roldán Urueña



Figura 3. Locatopedia en móvil.

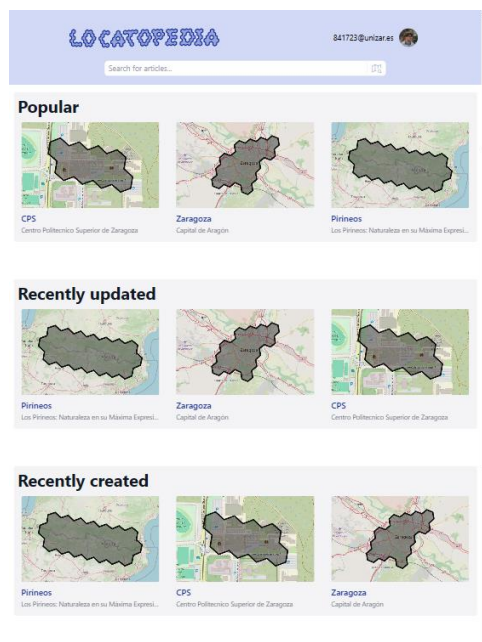


Figura 4. Locatopedia en tableta.

#### 4.3.1 GUI

La Figura 5 muestra la primera versión del mapa de navegación final. El diseño podrá cambiar algunos detalles de la interfaz grafica en función de necesidades técnicas a la hora de implementar Locatopedia.

Comentado [DR4]: Cambiar nombre a esta seccion

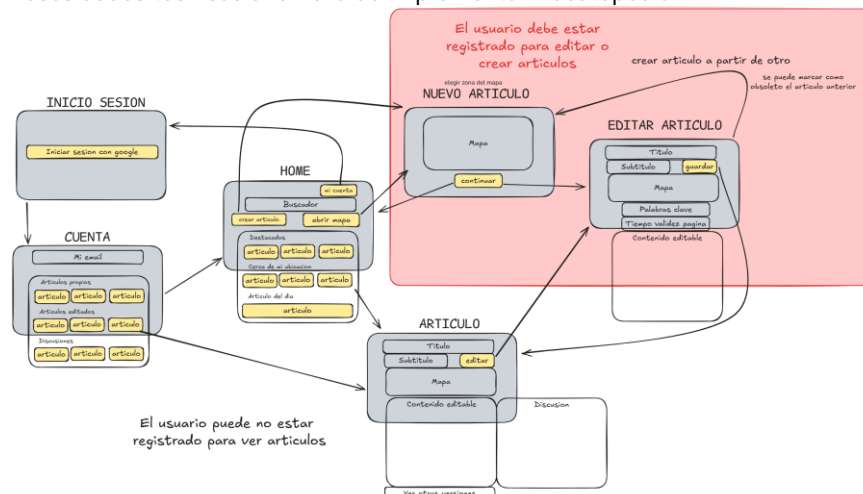


Figura 5. Primera versión del mapa de navegación de la aplicación.

La aplicación tendrá un conjunto de pantallas reducido y cada una será sencilla e intuitiva. Todas las páginas de la aplicación tendrán la misma cabecera que mostrará un enlace a la página inicial y a la sección de **MI CUENTA**.

La primera sección que aparecerá al entrar en la aplicación será la página **INICIO**. Tendrá un resumen de los artículos más populares y los artículos recientemente modificados y creados. En la parte superior se podrán buscar artículos por texto o seleccionado una zona en el mapa. Los criterios de búsqueda podrían combinarse.

La página **ARTICULO** mostrará el contenido del artículo. Aparecerán campos como título, subtítulo o contenido. También se verá un mapa interactivo donde se resaltará la región del artículo en cuestión. Se permite ver versiones anteriores del artículo por medio de un botón en la parte inferior de la página. Además, se podrá ver la discusión de los usuarios en relación con el contenido, e incluso se podrá participar en dicha discusión. Por último, encontraremos una opción para editar el artículo según nuestro criterio.

Las opciones de editar artículo y participar en la discusión solo podrán realizarlas usuarios registrados.

En la sección **EDITAR ARTICULO**, aparecerán los mismos campos que en la página de **ARTICULO**, pero con campos de contenido editable. Es decir, un usuario registrado será capaz de editar el título, subtítulo y/o contenido de cualquier artículo. No será posible modificar la región que ocupa el artículo, ya que esta región es el identificador del artículo.

Para crear un artículo nuevo será necesario usar el mapa de la página *NUEVO ARTICULO* para seleccionar la región que tratará el nuevo artículo. Después se establecerá el contenido en una pestaña similar a *EDITAR ARTICULO*. La página *INICIO SESION* se encargará de redirigir al usuario a servicios de terceros para que inicie sesión usando las cuentas que tenga el usuario en estos servicios. Google será el principal. Una vez se haya iniciado sesión, se podrá navegar a la sección *MI CUENTA* que mostrará información personal del usuario, como el email y artículos relacionados con el usuario. Aparecerán los artículos creados y modificados y en los que el usuario haya participado en sus discusiones.

#### 4.4 Interfaces para la integración en aplicaciones

Para productos sin GUI, o con una GUI de administración o similar pero cuyo principal valor está en proporcionar servicios a otros sistemas software (p.ej. ofreciendo una API). Una API no significa "una API REST". Una API pueden ser las clases y operaciones públicas principales de una biblioteca de software, un conjunto de scripts de procesamiento de datos... En cualquier caso, aquí se describe esa API desde el punto de vista de la captura de requisitos: qué se quiere ofrecer a otros sistemas. El resultado final se describiría en la sección 3.

La aplicación GUI de la página web sigue una estructura organizada para todas las secciones que hay. Cada sección tiene una ruta API distinta (por ejemplo */account* corresponde a la sección personal) con posibles parámetros (por ejemplo, el identificador de un artículo). Es decir, si conocemos la documentación, podemos navegar por la aplicación sin necesidad de hacer clic en botones dentro de la aplicación.

Por otro lado, el servidor del sitio web debe almacenar y proporcionar toda la información necesaria para ser mostrada en el sitio web. Aparte de esto, también debe ser posible modificar esa información, acción requerida al crear un nuevo artículo o editar uno ya existente.

Para ello se ha desarrollado un conjunto de API REST que ejecutan todas las acciones requeridas y envían la información solicitada. Estas consultas se dividen en 2 grandes grupos.

Las primeras, son consultas no protegidas, es decir, para acceder a estas APIs no hace falta que el usuario haya iniciado sesión previamente. En primer lugar, el servidor ofrece una ruta para recuperar toda la información de un artículo (RF 1). Locatopedia ofrece también un historial de versiones, por lo tanto, se ofrece una ruta API para recuperar versiones anteriores de un artículo (RF 2). Además de esto, se ofrecen rutas APIs que devuelven los artículos populares (los que más usuarios han marcado como favoritos), los recientemente creados y editados y un artículo aleatorio que se mostrará como artículo del día (RF 3). Por último, se deben añadir rutas para permitir a los usuarios hacer búsquedas, tanto por texto como geográficas. Para ello, se ha añadido una ruta que devuelve los artículos que coinciden, en texto, con un parámetro de búsqueda (RF 4). La coincidencia geográfica se resuelve en el cliente (RF 5).

**Comentado [DR5]:** Resaltar idea de API. Cada página tiene PURL, pero */edit* es para editar. Es decir, si te sabes la url no tienes que navegar



Diego Raúl Roldán Urueña

Para iniciar sesión, no se ha añadido ninguna ruta específica. Se utiliza OAuth 2.0, y es Google quién gestiona registro, inicio de sesión en la aplicación y validación de sesiones (RF 6).

Por otro lado, encontramos las consultas API que requieren de autenticación en la aplicación. El servicio API recupera el token de identificación, que se debe añadir a la consulta, y el servidor proporciona información privada correspondiente asociada al usuario relacionado con dicho token. Una de las rutas protegidas que el servidor proporciona permite marcar y desmarcar como favorito un artículo (RF 8), otra, permite recuperar los artículos favoritos de un usuario y los artículos creados y modificados por un usuario (RF 7). Aparte de esto, el servicio de API proporciona rutas para crear nuevos artículos (RF 9 y RF 10) y editar artículos ya existentes (RF 11) incluyendo la información necesaria en la solicitud API. También se ha añadido una ruta API para permitir borrar un artículo al usuario creador (RF 14) y a usuarios administradores (RF 15).

**Comentado [DR6]:** Quizas combinar la parte de diseño

## 5 Diseño de la solución

### 5.1 Arquitectura

Describir la idea general: estilo arquitectural principal del sistema, frameworks utilizados y qué implicaciones arquitecturales tienen. Si hay consideraciones importantes de prestaciones o de seguridad que ha habido que tener en cuenta.

Locatopedia sigue más de una arquitectura para aplicaciones web.

En primer lugar, se ha desarrollado una arquitectura multicapa de tres niveles.

- **Cliente.** Es la aplicación con la que interactúa el usuario final. Le da las funcionalidades necesarias para poder ver, modificar y editar artículos y su cuenta. También se conoce como *front-end* o interfaz gráfica de usuario GUI y se desarrolla con el framework *React*. Proporciona una interfaz dinámica, intuitiva y responsiva ante distintos tamaños de pantalla. Esta capa aplica medidas de seguridad para evitar que un usuario acceda a información sensible, como datos personales de otros usuarios.
- **Servidor.** Es la aplicación que se ejecuta en la nube que proporciona la lógica al cliente. Se desarrolla utilizando la tecnología moderna llamada Node.js que actúa como una capa intermedia que gestiona la lógica de negocio, maneja la autenticación y se comunica con la base de datos. Permite que las modificaciones realizadas por el usuario se vean reflejadas en todos los clientes. Esta capa también aplica medidas de seguridad. Los recursos protegidos requieren el envío de una credencial proporcionada por el servicio de tercero, para autenticar y autorizar el acceso o edición de un cliente a un recurso.
- **Base de datos.** Almacena de forma estructurada todos los datos de la aplicación para proporcionar un acceso rápido y seguro. Se utiliza un sistema PostgreSQL ya que es de código abierto y muy escalable. Por ahora el sistema no almacena mucha información, pero debe ser capaz de soportar un gran volumen de datos. PostgreSQL proporciona una escalabilidad excelente, múltiples conexiones simultáneas y replicación que lo convierte en una de las mejores bases de datos actualmente.

Para la parte del *backend* se ha seguido una arquitectura híbrida. En la mayor parte funciona como un *backend* monolítico, donde es propio sistema el que responde a todas las peticiones de la API, pero se ha añadido un servicio en Python. Esto se debe a que la librería de gestión de hashes e identificadores únicos de celdas está escrita en Python. Se ha desarrollado una API interna con Flask para proporcionar acceso a los métodos y funciones ya desarrollados en Python.

Tanto el cliente como el servidor se ejecutan en la misma máquina.

### 5.1.1 Contexto

Esta sección seguramente solo tendrá sentido para sistemas que son parte de otros mayores, para dejar claro qué parte es lo del TFG, y qué parte es lo que ya había.

La mayor parte de este proyecto es elaboración propia. Se han utilizado librerías públicas como turf (para gestionar las formas geométricas de las celdas), leaflet (para los mapas), react-oauth (para gestionar la sesión con Google) o h3 (para gestionar las celdas de la malla global discreta). También se ha utilizado la librería en Python mencionada previamente desarrollada por Rubén Béjar (tutor de este trabajo). El resto del desarrollo es parte del TFG.

**Comentado [DR7]:** Cambiar a vista de moduls, paquetes clases

### 5.1.2 Modelo de datos

Un modelo de entidades y relaciones (diagrama de clases UML o, si no hay otro remedio, notación prehistórica E-R). Más las explicaciones fundamentales: qué es cada entidad, qué es cada atributo, qué implican las relaciones.

Se puede complementar con un modelo de implementación (modelo relacional, documentos JSON en MongoDB o lo que sea), pero lo más crítico es el modelo de entidades y relaciones.

Este es diagrama del modelo entidad-relación que almacena toda la información necesaria para Locatopedia.

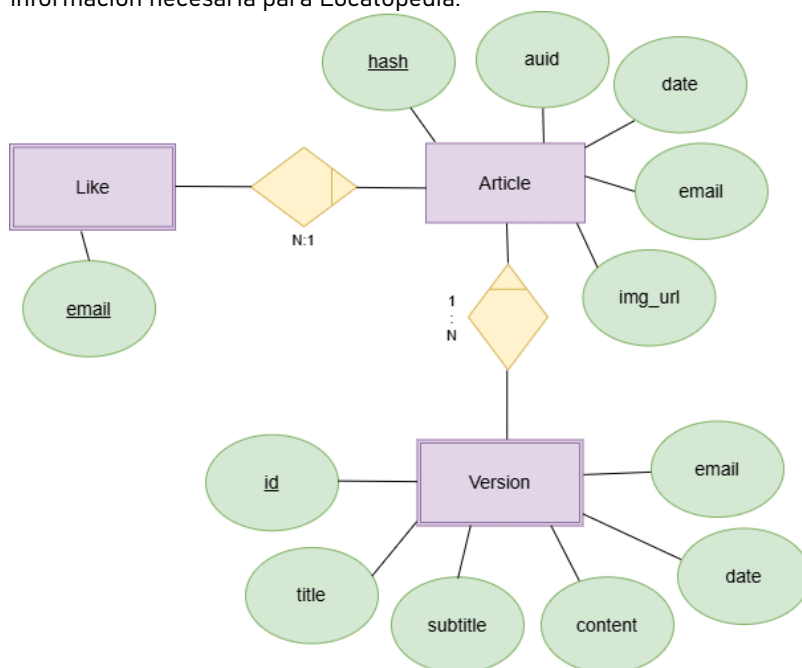


Figura 6. Diagrama Entidad-Relación.

**Comentado [DR8]:** Quizas mejor en UML

Las entidades se representan con un rectángulo de color morado, y tienen doble marco en caso de ser una entidad débil. Se pueden apreciar 3 entidades, dos de ellas entidades débiles y una entidad fuerte. Recordemos que las entidades débiles dependen de otra entidad, en cambio, las entidades fuertes existen por sí mismas. En este caso, las entidades de 'Like' y de 'Version', deben tener asociadas una entidad 'Article' para que tengan sentido. Por ello, las relaciones 'Like' - 'Article' y 'Version' - 'Article' sufren una dependencia. Si se eliminase un artículo, todo elemento en 'Like' o 'Version' carece de sentido. Además, la cardinalidad de estas relaciones es de uno a muchos (1 : N). Esto quiere decir que un artículo se puede relacionar con muchos *likes*, pero un *like* solo puede relacionarse con un único artículo. De la misma forma, un artículo puede tener muchas versiones, pero una versión siempre está relacionada con un único artículo.

Vemos que todas las entidades tienen atributos. La entidad 'Article' tiene en total 5. Se puede observar que el atributo *hash* está subrayado, esto indica que este campo forma parte de la clave primaria de la entidad, es decir, solo puede haber un artículo con este *hash* en el sistema. Otros atributos, como 'email' o 'date', entre otros, pueden repetirse en el sistema.

En el caso de la entidad 'Version', también se pueden ver varios atributos, uno de ellos es parte de la clave primaria. Es importante recordar que, al ser una entidad débil, añade a sus atributos la clave primaria de la entidad con la que está relacionada, que funciona como segunda parte de la clave primaria. En este caso, el campo *hash* también se encuentra en la entidad 'Version'.

La entidad 'Like' tiene un comportamiento similar. La primera parte de la clave primaria es el atributo *email*, y la segunda parte se hereda de la entidad fuerte de la que depende 'Article'.

### 5.1.3 Paquetes

- Un diagrama UML con los paquetes principales. Tanto del front-end como del back-end, si hay de ambos. Sin mucho detalle, es una idea general.

Un diagrama de paquetes muestra los principales módulos del sistema, organizados en paquetes con nombres descriptivos, y las relaciones entre ellos. Refleja la estructura lógica del proyecto y ayuda a entender cómo se organiza el código y cómo se conectan sus componentes principales.

A continuación, se muestra el diagrama de paquetes correspondiente a la estructura del proyecto:

**Comentado [DR9]:** Añadir tabla usuario para guardar administradores

**Comentado [DR10]:** Quizas cambiar el nombre por contenido

**Comentado [DR11]:** Explicar no describir

**Comentado [DR12]:** No olvidarse de API

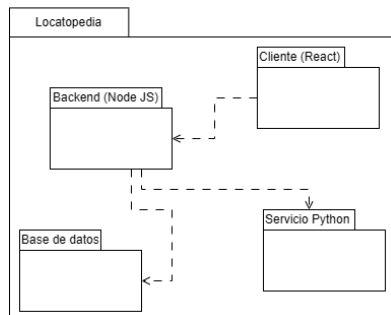


Figura 7. Diagrama de paquetes.

Locatopedia tiene 2 paquetes principales.

El primer bloque corresponde al backend del sistema. En el diagrama, los paquetes principales desarrollados en este proyecto se muestran en color azul, mientras que las librerías externas utilizadas aparecen en color blanco. En total, el backend cuenta con dos paquetes propios.

El paquete principal, denominado “node.js”, implementa la lógica central del backend. Es responsable de recibir todas las solicitudes provenientes del frontend, procesarlas y devolver la información solicitada al usuario. Para resolver ciertos tipos de peticiones específicas, este paquete delega en otro componente del sistema: el paquete “app.py”, una API auxiliar escrita en Python. Esta API se encarga principalmente de operaciones relacionadas con la conversión y manipulación de regiones geoespaciales representadas mediante conjuntos de identificadores de celdas (por ejemplo, transformaciones entre índices H3, regiones geográficas y códigos AUID). Para ello, utiliza una tercera librería especializada que proporciona estas funcionalidades de procesamiento geográfico.

Además, el paquete “node.js” incorpora varias librerías externas para gestionar funcionalidades clave del sistema. Entre ellas se encuentra google-auth-library, que permite implementar la autenticación OAuth 2.0 con Google de forma segura y sencilla, y se encarga de verificar y validar los tokens de acceso de los usuarios. También se utiliza pg, una librería oficial para Node.js que facilita la conexión y la interacción con bases de datos PostgreSQL, permitiendo ejecutar consultas, manejar conexiones y realizar operaciones sobre la base de datos de forma eficiente y segura.

El segundo se refiere al frontend, es decir, a la parte de la aplicación con la que interactúa directamente el usuario final. El paquete principal se representa en color azul e incluye diversas librerías externas. Las bibliotecas más relevantes se han incorporado al diagrama y aparecen en color blanco.

Locatopedia utiliza Leaflet como librería principal para la visualización de mapas interactivos en el navegador. Leaflet permite renderizar mapas ligeros y altamente personalizables, con soporte para capas, marcadores, eventos y controles de navegación, todo con un rendimiento eficiente incluso en dispositivos modestos.

Para la gestión y análisis espacial basado en celdas hexagonales, se emplea h3-js, la implementación en JavaScript de la biblioteca H3 desarrollada por

Uber. Esta herramienta permite indexar coordenadas geográficas en un sistema jerárquico de celdas hexagonales, facilitando operaciones como asignación de puntos a celdas, búsqueda de vecinos, generación de anillos concéntricos, y agregación espacial.

Complementando este conjunto, se utiliza Turf.js, una librería de análisis geoespacial que ofrece una amplia colección de funciones para trabajar con geometrías GeoJSON. Turf permite realizar cálculos como distancias, áreas, intersecciones, buffers o simplificación de geometrías, lo que resulta útil tanto para validaciones como para operaciones más avanzadas sobre los datos geográficos del sistema.

#### 5.1.4 Clases

- Un diagrama UML de clases con las clases principales. De nuevo tanto de front-end como de back-end si tiene sentido. Centrarse en las clases principales, en sus relaciones y en sus operaciones principales.

Un diagrama de clases representa las estructuras de datos y relaciones entre clases dentro del sistema. Muestra los atributos y métodos principales de cada clase, así como las asociaciones, herencias o dependencias entre ellas.

A continuación, se muestra el diagrama de clases correspondiente a la lógica del proyecto:

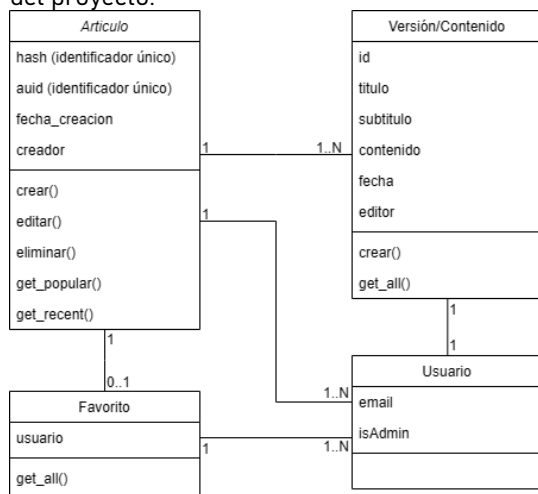


Figura 8. Diagrama de clases.

Y, por supuesto, todas las explicaciones pertinentes que permitan entender esos diagramas: descripción de los paquetes, papel de las clases principales en el sistema, patrones de diseño y estilos arquitecturales empleados, principales decisiones tomadas (decisiones implica que había varias opciones sensatas y se ha elegido una de ellas basándose en criterios racionales),

La clase principal del modelo espacial es CeldaDGGS, encargada de almacenar la información básica de cada celda, incluyendo su identificador,

coordenadas geográficas o las celdas vecinas. Estas celdas forman parte de regiones geográficas que se identifican mediante dos propiedades clave: un hash y un auid. Ambos representan la misma región, es decir, contienen los identificadores de las celdas que la conforman, pero tienen distintos usos. El hash es una cadena más corta, adecuada para usarse en URLs como identificador público. En cambio, el auid es una cadena más larga que permite reconstruir directamente todos los identificadores de las celdas que componen la región. Esta dualidad permite mantener una representación compacta en la interfaz y, al mismo tiempo, conservar una estructura eficiente para el procesamiento interno.

La clase Artículo hereda las propiedades hash y auid, ya que cada artículo está relacionado con una región geográfica específica. Además, incluye información adicional como fecha de creación y más metadatos necesarios para su gestión dentro de Locatopedia.

Dado que los artículos pueden modificarse a lo largo del tiempo, existe una clase separada llamada Versión/Contenido, que almacena cada versión del contenido de un artículo. Esta clase permite consultar el historial completo de ediciones y restaurar versiones anteriores si es necesario.

Finalmente, la clase Favorito representa la relación entre usuarios y artículos marcados como favoritos.

Por otro lado, la clase Usuario representa a los usuarios del sistema, e incluye un identificador único y un campo que indica si ese usuario tiene privilegios de administrador.

Tanto la clase Artículo, como Versión/Contenido y Favorito, incluyen una propiedad que referencia al nombre de usuario correspondiente al creador, editor o usuario que realizó la acción, lo que permite mantener trazabilidad y control sobre las interacciones dentro del sistema.

Este diagrama es conceptual.

#### 5.1.5 Componentes y conectores

- Un diagrama UML con los componentes principales y sus conectores. Dependiendo del tipo de sistema esos componentes serán objetos, o serán procesos que agruparán a muchos objetos, o serán servicios web...

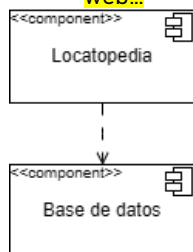


Figura 9. Diagrama de componentes principales.

- Algún diagrama UML (secuencia, interacción, máquinas de estados...) ilustrando el comportamiento dinámico del sistema suele ser

interesante, especialmente si este comportamiento, o alguna parte de este, tiene cierta complejidad.

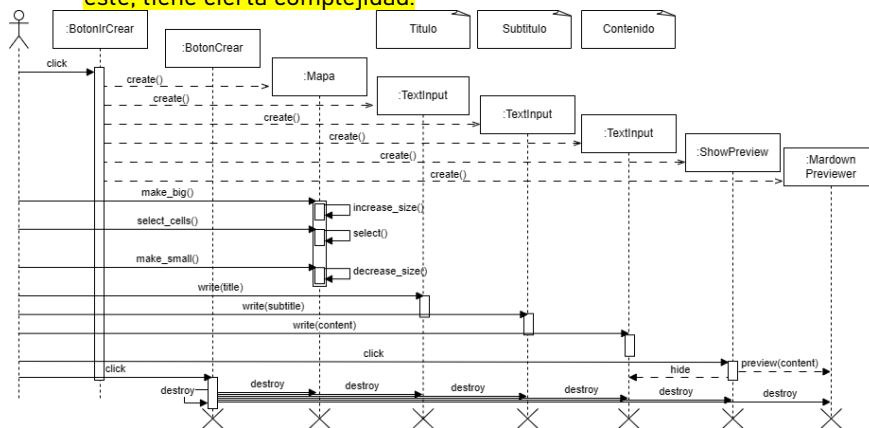


Figura 10. Diagrama de secuencia.

Y, por supuesto, todas las explicaciones pertinentes que permitan entender esos diagramas: qué es cada componente, qué permite cada conector, documentación de las interfaces (puertos/API) más relevantes del sistema, aspectos de concurrencia/asincronicidad/paralelismo/operaciones periódicas, estilos y patrones arquitecturales y de diseño que tiene más sentido explicar aquí que en la parte de módulos, decisiones de diseño relevantes en esta parte etc.

Relacionar componentes con módulos es interesante (cada línea de código que hemos puesto en un módulo debería ejecutarse en algún componente), sobre todo si ayuda a clarificar algún aspecto complejo.

### 5.1.6 Distribución

- Un diagrama UML del despliegue (artefectos en nodos interconectados) y/o de la instalación (artefectos sobre el sistema de ficheros).

Y, por supuesto, todas las explicaciones pertinentes que permitan entender esos diagramas: qué es cada nodo, qué tipo de rutas los interconectan, cómo se relacionan los artefactos de estos diagramas con los componentes de la sección anterior... Todo esto es especialmente interesante si hay un despliegue automatizado, cosas en Cloud, en contenedores (Docker directamente, o se usa Kubernetes o similar...) y habría que explicar todo eso en esta sección.

## 5.2 Implementación

Si esto es lo bastante grande, puede tenerse un capítulo entero (sería el 4) para la Implementación.

**Comentado [DR13]:** Explicar que es DGGS, como se ha utilizado



Aspectos interesantes de la implementación que sean de bajo nivel de abstracción, o estén localizados en módulos/componentes particulares y que no se hayan considerado lo bastante interesantes para el punto de arquitectura se pueden poner aquí.

Aquí se puede describir la GUI que ha quedado al final del sistema, e incluso poner un diagrama de navegación definitivo si tiene sentido.

Locatopedia desarrolla una interfaz gráfica de usuario sencilla y funcional, diseñada para que cualquier persona pueda interactuar con ella sin necesidad de una curva de aprendizaje pronunciada. Se ha seguido un estilo minimalista mejorando la experiencia del usuario.

En todas las páginas encontramos una cabecera donde se incluye una barra de búsqueda de texto y un mapa para hacer búsqueda geográfica para que los usuarios puedan leer los artículos que más les interesen en todo momento. Además, encontramos enlaces tanto a la página principal de Locatopedia y otro enlace a la página de cuenta personal. En caso de no haber iniciado sesión, este enlace te permitirá hacerlo.

La página principal muestra un resumen general de los artículos existentes, artículos más populares del momento, y que se han editado o modificado recientemente permitiendo a los usuarios explorar algunos de los artículos disponibles de manera intuitiva.

Cada artículo se muestra en una página distinta, identificada en la URL de forma permanente por el conjunto de celdas que contiene dicho artículo. Esto significa que el enlace web de un artículo no cambiará nunca, aunque se modifique el título, subtítulo y/o contenido de un artículo. Aquí se muestra otra información relevante sobre el artículo, como el autor, fecha y hora de la última modificación del artículo correspondiente. Además, se puede acceder a la página de versiones del artículo, donde se puede acceder al contenido de todas las versiones anteriores de dicho artículo.

En la página de cuenta personal, accesible solo con sesión iniciada, se muestran los artículos marcados como favoritos y los artículos creados y modificados por el usuario. Aquí aparece un botón para cerrar la sesión iniciada y un botón para crear un artículo nuevo. En esta página, se debe elegir un título, un subtítulo, un contenido y el conjunto de celdas sobre las que tratará el nuevo artículo.

Para crear un nuevo artículo o para editar uno ya existente, existe una página que permite cambiar los valores de título, subtítulo y contenido y permite modificar la selección de celdas en caso de tratarse de ser creación de un artículo.

Los botones son grandes y bien espaciados, asegurando accesibilidad y facilidad de uso. Cada artículo se puede visualizar con un solo clic, mostrando su contenido estructurado con descripciones, imágenes y enlaces relevantes. La edición y colaboración también se simplifican con un sistema intuitivo de modificación de contenido, donde los usuarios pueden agregar o actualizar información sin perder la coherencia del sitio. También pueden ver una previsualización en legible del texto Markdown que escriben en el contenido de un artículo.

Diego Raúl Roldán Urueña

El diseño prioriza la claridad y la funcionalidad, evitando elementos visuales innecesarios que puedan dificultar la experiencia. Los colores y tipografías están elegidos para ofrecer una lectura cómoda, con contrastes adecuados y un esquema visual limpio. En general, la interfaz busca ser lo más ligera posible sin sacrificar usabilidad, asegurando que la aplicación sea rápida y eficiente en cualquier dispositivo.

### 5.2.1 DGGs

Locatopedia incorpora el uso de un Sistema de Mallas Globales y Discretas (Discrete Global Grid Systems, DGGs), una tecnología espacial emergente que ofrece una alternativa estructurada y jerárquica a los sistemas tradicionales de representación geográfica. Un DGGs se basa en la división sistemática y recursiva de la superficie terrestre en celdas discretas, las cuales mantienen propiedades topológicas y métricas bien definidas. A diferencia de los sistemas basados en coordenadas continuas (como latitud y longitud), los DGGs permiten representar ubicaciones y áreas mediante identificadores únicos de celdas, lo cual resulta especialmente útil en aplicaciones como Locatopedia.

En un DGGs, la Tierra se proyecta sobre una figura poliédrica (como un icosaedro, un octaedro o un hexágono), la cual es posteriormente subdividida en celdas de menor tamaño siguiendo una lógica jerárquica. Esta estructura permite realizar operaciones espaciales de forma eficiente y coherente en múltiples escalas de resolución. Entre las ventajas más destacadas se encuentran la posibilidad de realizar indexación espacial uniforme, la agregación de datos multiescala, y la compatibilidad con algoritmos de análisis distribuidos o en paralelo.

Locatopedia, implementa un mapa interactivo en el cual se muestran celdas DGGs que varían dinámicamente en función del nivel de zoom. Esta funcionalidad permite que la superficie terrestre se divida en unidades espaciales jerárquicas y discretas, las cuales pueden visualizarse y seleccionarse de manera intuitiva por parte del usuario. Cada artículo dentro del sistema se ha vinculado a un conjunto específico de estas celdas, lo que permite representar su alcance geográfico de forma precisa y estandarizada. Este enfoque aporta múltiples ventajas. En primer lugar, al utilizar celdas DGGs como unidades básicas de georreferenciación, se ha logrado una representación espacial uniforme e independiente de proyecciones cartográficas particulares, lo que mejora la consistencia del sistema a escala global. En segundo lugar, al permitir que los artículos estén asociados a un conjunto de celdas —y no a simples coordenadas puntuales— se posibilita una representación más rica y flexible del contenido geográfico, adecuada tanto para elementos puntuales como para áreas de interés más extensas. En conjunto, esta estrategia de organización y visualización basada en celdas DGGs no solo mejora la precisión y la claridad de la representación

Diego Raúl Roldán Urueña

geográfica, sino que también sienta las bases para un sistema escalable, interactivo y fácilmente ampliable.

### 5.2.2 ¿Por qué usar un DGGS?

El uso de un Sistema de Mallas Globales y Discretas (DGGS) ha proporcionado una base estructural sólida y coherente para el diseño y funcionamiento de esta aplicación, aportando ventajas fundamentales que se alinean de forma directa con sus objetivos. A través de una partición jerárquica, regular y recursiva del espacio terrestre, se ha logrado representar el territorio mediante celdas discretas, cada una de las cuales posee propiedades espaciales bien definidas y puede ser referenciada mediante un identificador único. Esta representación ha permitido que el contenido de la aplicación se organice de forma precisa, sin ambigüedad ni dependencia de proyecciones cartográficas específicas.

El empleo de estas celdas como unidades atómicas de contenido ha facilitado notablemente tareas críticas para el funcionamiento del sistema. Por un lado, la indexación espacial se ha simplificado, ya que cada artículo se ha asociado a un conjunto específico de celdas, lo cual ha permitido realizar búsquedas eficientes por ubicación, establecer relaciones entre zonas adyacentes, y filtrar contenido en función del área seleccionada. Por otro lado, el almacenamiento y la agregación de información espacial se ha visto beneficiado por la estructura jerárquica del DGGS, que permite trabajar de forma coherente a diferentes niveles de resolución. Esto ha hecho posible representar con la misma lógica tanto artículos centrados en áreas pequeñas como otros que abarcan regiones extensas, sin necesidad de modificar la estructura del sistema.

Una ventaja adicional ha sido la estandarización de la georreferenciación, que en lugar de depender de coordenadas continuas o polígonos arbitrarios, se ha basado en identificadores discretos, lo cual ha facilitado la comparación, agregación y análisis del contenido espacial. Esta representación también ha resultado más robusta frente a errores, ya que la lógica de asignación y consulta se ha mantenido constante en todas las partes del sistema.

Desde el punto de vista de la interacción, el uso de un DGGS ha permitido implementar un mapa interactivo clicable, sobre el que los usuarios pueden seleccionar visualmente celdas para consultar o editar su contenido. Este componente ha contribuido significativamente a mejorar la usabilidad del sistema, ya que ha transformado operaciones espacialmente complejas en acciones directas e intuitivas. En lugar de requerir conocimientos técnicos en geolocalización o dibujo de polígonos, los usuarios han podido trabajar con regiones geográficas simplemente seleccionando celdas sobre el mapa, lo cual ha reducido la fricción en el uso y ha ampliado el público potencial de la aplicación.

Diego Raúl Roldán Urueña

Más allá de los beneficios funcionales inmediatos, el enfoque basado en DGGS ha establecido una base técnica sólida y escalable para futuras ampliaciones del sistema. Gracias a la estructura regular de las celdas, podrían integrarse fácilmente mecanismos de visualización de capas temáticas, análisis estadísticos por región, detección de solapamientos entre artículos, o incluso interoperar con otras aplicaciones que utilicen modelos de cuadrícula discreta. Además, se ha abierto la posibilidad de combinar los datos espaciales con otros sistemas de clasificación, como los derivados de inteligencia artificial, para enriquecer aún más la organización y exploración del contenido.

En conjunto, puede afirmarse que el uso de un DGGS no constituye un mero componente técnico más dentro del sistema, sino que representa el fundamento conceptual sobre el cual se ha construido toda la lógica de representación espacial y navegación. El proyecto perdería gran parte de su sentido sin este modelo subyacente, ya que tanto la organización de los artículos, como la interacción en el mapa, la escalabilidad de la información y el potencial de crecimiento futuro, dependen directamente de la elección de este enfoque.

### 5.2.3 Elección de DGGS

Existen múltiples Sistemas de Mallas Globales y Discretas (DGGS, por sus siglas en inglés) que han sido diseñados para representar de forma uniforme y jerárquica la superficie terrestre. Estos sistemas se han desarrollado con distintos enfoques y objetivos, por lo que presentan características variadas en cuanto a la forma de las celdas, su distribución espacial, la escalabilidad, y la disponibilidad de herramientas y documentación.

Algunos de los DGGS más conocidos y utilizados son los siguientes:

El primero de ellos, H3 (Hexagonal Hierarchical Spatial Index), que ha sido desarrollado por Uber, se basa en celdas hexagonales. Se ha diseñado para indexar de forma eficiente grandes cantidades de puntos móviles, como vehículos, lo que ha hecho que sea ampliamente adoptado en entornos de producción. Se ha valorado por su jerarquía clara, su baja distorsión angular, y por ofrecer una vecindad regular entre celdas. Además, se ha publicado como proyecto de código abierto con soporte en múltiples lenguajes y buena documentación, lo que ha facilitado su integración en numerosos proyectos.

rHEALPix es un sistema, de origen académico, basado en una adaptación de la proyección HEALPix. Utiliza celdas cuadradas y es especialmente adecuado para datos que requieren representación continua sobre toda la superficie terrestre, como observaciones climáticas o datos satelitales. Aunque se ha usado en proyectos científicos, principalmente en Canadá y Australia, su adopción en entornos comerciales ha sido menor. Sin embargo, ha sido valorado por su baja distorsión de área y la facilidad para trabajar con jerarquías espaciales.

Diego Raúl Roldán Urueña

S2 Geometry fue creado por Google. Este sistema utiliza celdas cuadradas proyectadas sobre la esfera. Aunque no se diseñó específicamente como un DGGS, se ha utilizado ampliamente para realizar cálculos geométricos precisos sobre la superficie terrestre. A diferencia de otros sistemas, S2 pone más énfasis en la exactitud matemática de las operaciones geoespaciales. Su uso es común en sistemas que requieren indexación espacial a nivel global, como bases de datos y motores de búsqueda.

Por último, encontramos ISEA3H. Este sistema utiliza una proyección icosaédrica con celdas hexagonales. Se basa en la proyección ISEA (Icosahedral Snyder Equal Area), que preserva el área y permite una subdivisión uniforme de la esfera. Aunque proporciona buenas propiedades geométricas, sus implementaciones disponibles tienden a estar menos mantenidas, lo que puede dificultar su adopción en proyectos a gran escala.

La elección de H3 en este proyecto ha sido motivada tanto por sus ventajas técnicas como por su ajuste a los objetivos funcionales. H3 es un sistema de cuadrícula jerárquico y global basado en celdas hexagonales, desarrollado por Uber para la indexación eficiente de datos espaciales.

#### *5.2.3.1 ¿Por qué hexágonos en lugar de cuadrados?*

Una de las principales ventajas del uso de celdas hexagonales en sistemas como H3 es la uniformidad en la distancia entre celdas adyacentes. Cada celda hexagonal mantiene una distancia constante respecto al centro de sus vecinas, lo que simplifica considerablemente operaciones como el cálculo de vecinos, la difusión espacial o los algoritmos de suavizado. En contraste, las celdas cuadradas presentan distintos tipos de vecinos —por borde o por esquina—, lo que introduce irregularidades y una mayor complejidad en estos procesos.

Otra ventaja destacable es la menor distorsión visual y analítica. Si bien ningún sistema de proyección de polígonos regulares está completamente libre de distorsiones en contextos cartografía, los hexágonos utilizados en H3 ofrecen una adaptación visual más coherente, especialmente cuando se hacen zooms amplios sobre el mapa. Esta característica contribuye a una representación espacial más uniforme y legible, y mejora la calidad de los análisis visuales.

Finalmente, los hexágonos ofrecen una mejor eficiencia geométrica. Su menor relación perímetro-área, en comparación con otras formas como los cuadrados o triángulos, ayuda a reducir efectos no deseados en los bordes durante análisis agregados. Además, su forma permite representar de forma más precisa y fluida curvas y estructuras naturales del espacio geográfico, lo que resulta particularmente útil en aplicaciones de modelado ambiental o territorial.

Diego Raúl Roldán Urueña

#### *5.2.3.2 Características técnicas de H3*

Una de las características más potentes de H3 es su jerarquía escalable, que permite trabajar con distintos niveles de resolución geoespacial, desde escalas globales hasta extremadamente locales. El sistema divide el globo en celdas hexagonales que pueden ajustarse desde aproximadamente 4,3 millones de kilómetros cuadrados en el nivel 0, hasta menos de un metro cuadrado en el nivel 15. Esta jerarquía uniforme hace posible realizar análisis multiescalares sin cambiar de sistema ni comprometer la consistencia entre niveles.

Además, H3 proporciona una indexación eficiente, lo que permite asignar a cada punto o región geográfica un identificador único de celda. Esto simplifica tareas complejas como la agregación de datos, la unión entre conjuntos espaciales, el análisis de proximidad o el trazado de rutas, todo sin necesidad de manipular geometrías detalladas ni realizar operaciones geoespaciales costosas en términos de rendimiento.

Finalmente, H3 se beneficia de un amplio ecosistema y una documentación madura, fruto de su adopción en entornos industriales y su mantenimiento activo. Dispone de bindings en múltiples lenguajes de programación, ejemplos prácticos, herramientas auxiliares y una comunidad activa, lo que garantiza su viabilidad y soporte técnico en el largo plazo, tanto para proyectos personales como a escala empresarial.

### 5.3 GUI Actual

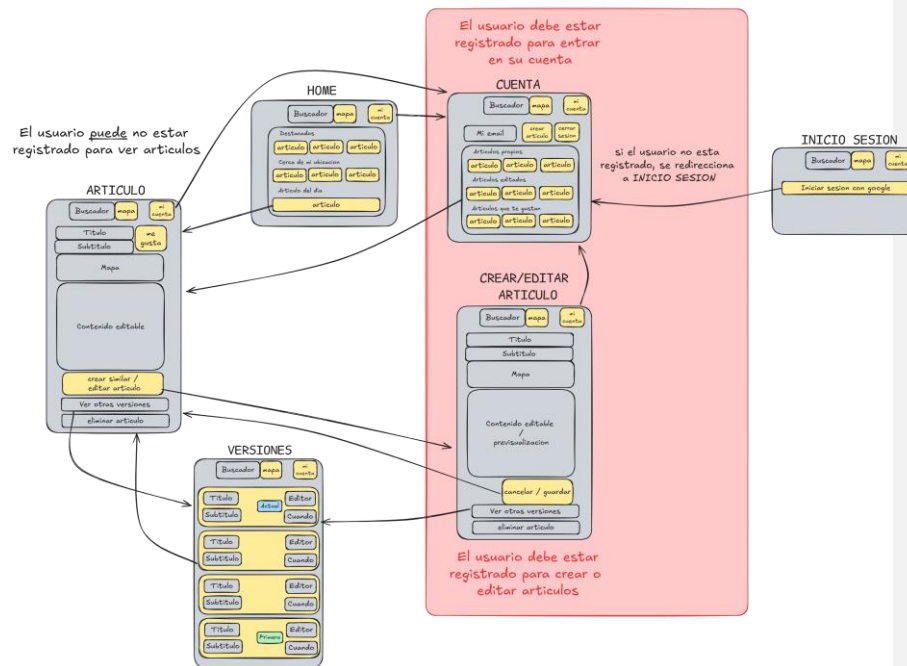


Figura 11. Mapa de navegación.

**Comentado [DR14]:** Hacer 2 cosas: diagrama de navegación y wireframe (enseñar que tiene cada pantalla)

La aplicación está diseñada para recibir dos tipos de usuarios principales con niveles de interacción distintos. Los más frecuentes son los usuarios no registrados, quienes simplemente acceden, navegan por la aplicación leyendo y obteniendo información de los artículos ya existentes sin necesidad de realizar ninguna acción adicional. Para ellos, la interfaz es extremadamente sencilla, con una estructura clara que les permite explorar el contenido sin obstáculos ni configuraciones innecesarias.

Por otro lado, los usuarios más avanzados son aquellos que editan y crean nuevos artículos o buscan versiones anteriores de artículos existentes. Para ellos, la interfaz debe ofrecer herramientas adicionales que les permitan gestionar dicha información de manera eficiente. Aunque la aplicación sigue manteniendo un enfoque minimalista, en este caso la GUI se vuelve algo más compleja, ya que incluye opciones para modificar el contenido de los artículos o crear nuevos artículos. Estos usuarios deben aprender a utilizar estas funcionalidades, por lo que la interfaz está diseñada para guiarlos en el proceso sin ser abrumadora.

Este equilibrio entre simplicidad y funcionalidad asegura que la experiencia sea accesible para los visitantes ocasionales, pero suficientemente potente para aquellos que desean contribuir activamente al contenido de la plataforma.

Si la construcción se ha automatizado (TravisCI, Github Actions o similares), describir aquí el pipeline de construcción. Describir el uso de sistemas de control de versiones (cuántos repositorios, qué workflows de Git etc.).

Para el control de versiones del proyecto, se ha utilizado Git como sistema de gestión del código fuente, en conjunto con GitHub como plataforma de alojamiento remoto del repositorio. Gracias a esta combinación, se ha podido llevar un seguimiento preciso de los cambios realizados a lo largo del desarrollo, facilitando la identificación y reversión de errores en caso necesario. GitHub ha proporcionado además un entorno centralizado para el almacenamiento seguro del código, así como herramientas adicionales para la gestión del proyecto, como issues y documentación. El uso de ramas ha permitido organizar las distintas funcionalidades de manera modular, favoreciendo una estructura de trabajo más ordenada incluso en un entorno de desarrollo individual.

Se ha seguido una estructura de monorepo para organizar el proyecto, integrando tanto el frontend como el backend en un único repositorio. Esta decisión ha permitido centralizar el desarrollo y facilitar la gestión conjunta de ambas partes de la aplicación. Dentro del repositorio, se han creado directorios independientes `—/frontend y /backend—` con el fin de mantener una separación clara, permitiendo al mismo tiempo compartir configuraciones comunes y simplificar tareas como el despliegue y la ejecución de scripts de desarrollo.

Para el despliegue de la aplicación, se ha utilizado Docker con el objetivo de garantizar un entorno reproducible, aislado y coherente entre desarrollo y producción en distintos dispositivos. Se han definido cuatro contenedores distintos: una base de datos PostgreSQL, el backend, el frontend y un servicio adicional implementado en Python. La base de datos se ha configurado a partir de una imagen oficial de PostgreSQL, mientras que los otros tres servicios disponen de sus propios archivos Dockerfile, a través de los cuales se han construido imágenes personalizadas con las dependencias y configuraciones necesarias. Posteriormente, se ha utilizado `docker-compose` para orquestar el sistema completo: mediante un archivo de configuración, se han definido los servicios, sus redes y volúmenes, y se ha automatizado tanto la construcción como la puesta en marcha de los contenedores. De esta forma, se ha conseguido un despliegue coherente y fácilmente replicable, facilitando tanto las tareas de desarrollo como las de mantenimiento.

En el archivo `docker-compose.yml` se han definido los servicios necesarios para ejecutar la aplicación, especificando tanto los puertos expuestos como la configuración adicional requerida por cada uno. El servicio `postgresql`, basado en la imagen oficial de PostgreSQL, se ha configurado mediante variables de entorno para establecer el nombre del usuario, la contraseña y la base de datos por defecto. No se ha expuesto ningún puerto para este contenedor, de esta manera, solo será accesible a través de la red interna de Docker ya que el backend es el único servicio que debe conectarse con la base de datos y no desde el exterior. Además, se ha definido un volumen persistente (`postgresql-data`) con el fin de conservar los datos incluso si el contenedor se reinicia o elimina.

Comentado [DR15]: Añadir archivo a anexos



Diego Raúl Roldán Urueña

El servicio backend se ha expuesto en el puerto 3000, que es donde escucha la API desarrollada, permitiendo su acceso tanto desde otros servicios como desde el exterior si fuera necesario. Se ha indicado una dependencia explícita de este servicio respecto a la base de datos mediante la directiva `depends_on`, asegurando que el contenedor de PostgreSQL se inicie antes que el backend. El servicio `www`, correspondiente al frontend, se ha expuesto en el puerto 80, que es el puerto estándar para tráfico HTTP, facilitando su acceso directo desde el navegador sin necesidad de configuración adicional. El script de configuración inicial de este contenedor se encarga de configurar un servicio web `nginx` para servir el sitio web compilado generado por `npm`. Por último, el servicio `dgstools`, es un servicio auxiliar ubicado en el backend desarrollado en Python que se comunica de forma interna con el resto del sistema Docker. De esta forma, se permite replicar el entorno de despliegue de manera sencilla y automática.

## 5.4 Procesamiento de datos

Si el proyecto ha tenido una parte importante de procesamiento de datos, explicarlo aquí. Si no, es probable que esta sección sobre.

## 5.5 Pruebas

Se incluyen tests automáticos y tests manuales para validación y verificación, pero también tests de prestaciones o tests de estrés (sobrecarga) si el tipo de proyecto los requiere. Herramientas usadas, planificación de las pruebas, guiones de las pruebas manuales, integración de las pruebas automáticas en el pipeline de construcción. Si se ha usado Sonar o algún otro analizador estático de código explicarlo aquí.

Cualquier tipo de proyecto va a requerir pruebas: cómo se validan los resultados es crítico se haga lo que se haga.

Contar con una estrategia de testing automatizado es esencial en cualquier sistema moderno. Los tests permiten detectar errores rápidamente, aseguran la estabilidad de la aplicación a lo largo del tiempo, y brindan confianza al momento de implementar nuevas funcionalidades o realizar refactorizaciones. En particular, los tests de extremo a extremo (end-to-end, E2E) son especialmente valiosos porque validan que todo el sistema —desde la interfaz de usuario hasta la lógica de negocio y las comunicaciones con el backend— funciona correctamente desde el punto de vista del usuario final.

Además de mejorar la calidad del software, los tests automáticos reducen la necesidad de pruebas manuales repetitivas, lo cual ahorra tiempo y mejora la eficiencia del equipo de desarrollo.

Locatopedia implementa tests automáticos de extremo a extremo utilizando Playwright. Estos tests simulan interacciones reales con la aplicación, verificando que los principales flujos de usuario funcionan correctamente de principio a fin.

Diego Raúl Roldán Urueña

### 5.5.1 Pruebas extremo-a-extremo

Estos son los flujos que incluyen los tests extremo a extremo:

#### 5.5.1.1 *Ver un artículo*

Asegura que se pueda hacer clic en la pantalla principal para navegar a un artículo concreto y verifica que se carga el contenido correcto.

#### 5.5.1.2 *Buscar artículo*

Utiliza el buscador de texto de la aplicación para verificar que se muestran todos los artículos con el contenido buscado, y que se cargue el contenido correcto al seleccionar un artículo de la lista de resultados de búsqueda.

#### 5.5.1.3 *Ver perfil*

Comprueba que un usuario autenticado pueda acceder a su perfil y visualizar sus artículos y actividad.

#### 5.5.1.4 *Crear artículo*

Testea que un usuario autenticado pueda crear un nuevo artículo correctamente. También comprueba que el contenido del artículo creado sea el mismo que el contenido que se ha elegido al crear el artículo.

#### 5.5.1.5 *Dar «me gusta» a un artículo*

Valida que el sistema de «me gustas» funcione y que la UI refleje correctamente el cambio. Comprueba que funciona correctamente tanto seleccionar como deseleccionar «me gusta».

#### 5.5.1.6 *Editar artículo*

Garantiza que un usuario pueda modificar un artículo existente y guardar los cambios. Para ello edita un artículo, comprueba que los cambios se hayan guardado con éxito, vuelve a editar el artículo como al principio y comprueba, de nuevo, que los cambios se hayan guardado con éxito.

#### 5.5.1.7 *Eliminar artículo*

Confirma que un artículo puede eliminarse correctamente y que desaparece de las vistas correspondientes.

Hay algunos tests que requieren autenticación para realizar acciones reservadas a usuarios registrados. Locatopedia usa OAuth 2.0 con Google MFA, dificultando el inicio de sesión dentro de la prueba automática. Por ello, se configura previamente la cookie de sesión correspondiente para que el navegador simulado por Playwright ejecute Locatopedia como si fuese un usuario registrado. De la misma forma, envía al backend el token de autenticación.

### 5.5.2 Prueba de estrés

Además de los tests funcionales y de extremo a extremo, el sistema también incorpora una prueba de estrés cuyo objetivo es evaluar la robustez, estabilidad y capacidad de respuesta de la aplicación cuando se enfrenta a una carga elevada. Este tipo de prueba es fundamental para identificar cómo se comporta el sistema cuando se acerca a sus límites operativos, y para detectar posibles cuellos de botella, errores de concurrencia, fugas de memoria o fallos inesperados en condiciones extremas.

El test de estrés se lleva a cabo mediante la creación masiva de artículos en la plataforma. Cada artículo generado contiene contenido normal y estructurado, como el que se esperaría en un uso cotidiano por parte de los usuarios, evitando introducir datos anómalos o especialmente pesados que pudieran distorsionar los resultados. El objetivo no es probar validaciones ni tolerancia a datos corruptos, sino observar cómo reacciona el sistema al aumento progresivo en el número de operaciones legítimas, similares a las que se producen en un escenario real de alta actividad.

Desde el punto de vista técnico, este test se implementa utilizando un script automatizado que simula un gran número de solicitudes concurrentes al backend, utilizando para ello la API pública de creación de artículos. El script autentica una sesión de usuario (siguiendo el mismo enfoque que en los tests E2E, es decir, inyectando las cookies de sesión y el header de autorización con un token válido) y comienza a enviar peticiones POST para crear nuevos artículos de forma programada. Es importante mencionar que cada artículo tiene una imagen, thumbnail, que muestra las celdas seleccionadas proyectadas en el mapa. La imagen se genera en el navegador cuando se crea un nuevo artículo y después se envía al backend el contenido de la imagen en formato base64 en la petición POST. Por ello, la petición puede llegar a transferir hasta 8MB. Al recibir la imagen, el backend comprime esta imagen transformándola en formato webp y la guarda en el sistema de ficheros. Este proceso no es el óptimo, la solicitud POST envía una cantidad de datos demasiado grande. Si Locatopedia se desplegase de manera pública y la cantidad de usuarios activos aumentase de forma significativa, este proceso debería ser diseñado de nuevo para evitar estos problemas.

Durante la ejecución del test se monitorizan métricas clave del sistema, como el uso de CPU y memoria en el servidor, la latencia de las respuestas, el número de errores HTTP, el rendimiento de la base de datos y cualquier fallo registrado en los logs del backend. De esta forma, es posible analizar si el sistema continúa respondiendo correctamente, si degrada su rendimiento de forma progresiva o si colapsa completamente al superar cierto umbral.

Diego Raúl Roldán Urueña

Se añade una tabla con los resultados del test de estrés realizado:

Numero artículos creados	Tiempo total (s)	Tiempo promedio POST (ms)	Éxitos	Errores	Uso de CPU (%)	Uso de RAM (MB)
100	1,3	77/s	100	0	204%	11MB
500	64	128	500	0	41	370
2000	315	142	1984	16	76	680

*Tabla 4. Resultados test de estrés.*

## 6 Gestión del proyecto

Señalar la metodología de gestión que se ha seguido. Dependiendo de la misma incluir planes, división del trabajo en iteraciones/sprints, horas de esfuerzo realizadas, análisis de riesgos realizado, etc.

El desarrollo del proyecto ha seguido una metodología ágil y flexible, ajustada a las particularidades de un entorno de trabajo individual. Aunque no se adoptó una metodología formal como Scrum o Kanban, se aplicaron muchos de sus principios esenciales: desarrollo iterativo, priorización dinámica de tareas, mejora continua y atención constante a los cambios. Esta estrategia permitió un control eficaz del proyecto sin necesidad de una planificación excesivamente estructurada, lo que resulta especialmente útil en proyectos desarrollados por una sola persona, donde la adaptabilidad y la autogestión juegan un papel fundamental.

Desde el inicio, el proyecto se concibió como una aplicación geoespacial interactiva centrada en el concepto de regiones definidas por celdas DGGS. A partir de esta idea central, se planificó una arquitectura compuesta por un frontend moderno (React), un backend en Node.js, una API auxiliar en Python para tareas geográficas especializadas, y una base de datos PostgreSQL. Esta decisión técnica marcó el punto de partida de la planificación y ejecución del sistema.

### 6.1 Fases del desarrollo

Aunque no se utilizaron “sprints” oficiales ni cronogramas rígidos, el trabajo se fue dividiendo de forma natural en cuatro fases diferenciadas, que pueden entenderse como equivalentes a iteraciones con objetivos concretos.

#### 6.1.1 Análisis y diseño técnico

Durante esta fase se definieron los objetivos funcionales y técnicos del sistema. Se estudiaron distintas opciones para representar regiones geográficas y se optó por utilizar el sistema de indexación H3, por su capacidad jerárquica, su baja distorsión y su eficiencia espacial. También se diseñó la arquitectura general del sistema, incluyendo la separación entre frontend, backend, API geográfica y base de datos.

En esta etapa se tomaron decisiones clave, como la implementación de autenticación mediante OAuth 2.0 con Google, el uso de imágenes generadas en el navegador y codificadas en base64, y la inclusión de funcionalidades como edición, versiones, favoritos y sistema de permisos.

#### 6.1.2 Desarrollo iterativo de funcionalidades

Diego Raúl Roldán Urueña

Esta fue la fase más extensa y se desarrolló de manera incremental. Se implementaron los módulos funcionales uno por uno, integrando y probando continuamente el sistema en su conjunto. Entre los módulos principales desarrollados están:

- Visualización de artículos y regiones
- Búsqueda por hash
- Creación y edición de artículos (incluyendo imágenes)
- Gestión de usuarios y login con Google
- Sistema de favoritos
- Historial de versiones y contenidos
- API geoespacial en Python para procesar auids

Las funcionalidades se probaron a medida que se integraban, con correcciones y refactorizaciones frecuentes. Esta forma de trabajo permitió evitar acumulaciones de errores y mantener la estabilidad del sistema.

### 6.1.3 Pruebas, validación y control de calidad

Una vez que el sistema era funcional de extremo a extremo, se desarrollaron tests automáticos E2E (end-to-end) utilizando la herramienta Playwright. Estos tests automatizan las interacciones del usuario con la aplicación, comprobando que las funcionalidades críticas operan correctamente: ver y buscar artículos, gestionar perfiles, crear, editar o eliminar contenido, y marcar artículos como favoritos.

Como el sistema usa autenticación OAuth 2.0, se optó por simplificar el proceso de login en los tests mediante la inyección directa de las cookies necesarias y el token de autorización en las cabeceras HTTP. Esta solución permitió automatizar las pruebas sin necesidad de realizar el login de forma manual en cada ejecución, manteniendo realismo y eficiencia.

### 6.1.4 Documentación, análisis de rendimiento y entrega

En la etapa final, se elaboró una documentación completa del sistema, incluyendo diagramas de clases, paquetes, flujos de datos y ejemplos de uso. Además, se realizaron pruebas de carga para simular la creación masiva de artículos con imágenes, con el objetivo de medir el comportamiento del sistema bajo condiciones de estrés. Estas pruebas revelaron los principales cuellos de botella, especialmente en la transferencia de imágenes en base64 y su posterior procesamiento.

Se detectó, por ejemplo, que al crear artículos con imágenes generadas en el navegador, el tamaño de la carga podía alcanzar los **8MB** por solicitud POST. Esta imagen se enviaba codificada en base64 y se almacenaba en el sistema tras una compresión a formato web. Aunque funcional, este enfoque no resulta escalable en contextos de alta concurrencia o uso público, y se

Diego Raúl Roldán Urueña

recomienda rediseñarlo en el futuro para mejorar la eficiencia y reducir la carga en red y servidor.

## 6.2 Integración y despliegue continuo (CI/CD)

Como parte del enfoque profesional del desarrollo, se ha configurado un sistema de CI/CD (Integración y Despliegue Continuos) que automatiza procesos clave como la ejecución de pruebas, el control de versiones y el despliegue del backend.

Se ha utilizado GitHub Actions para definir flujos de trabajo que se ejecutan en cada push o pull request hacia las ramas principales del repositorio. Este pipeline incluye:

- Instalación de dependencias del frontend y backend.
- Ejecución de los tests E2E con Playwright.
- Validación del estado del código (linting y formateo).
- Build automático del frontend.
- Configuración del servicio nginx con https.
- Despliegue del backend a un servidor de desarrollo Azure mediante SSH en contenedores docker.

Gracias a este sistema, cada modificación en el código es verificada automáticamente, y el entorno de pruebas siempre se mantiene actualizado, garantizando estabilidad y reduciendo el riesgo de errores en producción. El CI/CD también facilita futuras extensiones del sistema, ya que permite a los usuarios externos probar Locatopedia, permitiendo recibir feedback al desarrollador.

## 6.3 Estimación de esfuerzo y gestión del tiempo

Aunque el proyecto no utilizó una herramienta específica de gestión como Trello o Notion, se mantuvo una lista estructurada de tareas y objetivos semanales, con una priorización flexible según la evolución del trabajo. El avance fue continuo y progresivo, con revisiones constantes y rediseños puntuales cuando se detectaban problemas estructurales o nuevas oportunidades de mejora.

Se estima que el esfuerzo total invertido en el desarrollo del sistema fue de aproximadamente 280 horas, distribuidas del siguiente modo:

- Desarrollo backend y API Python: 75 horas
- Frontend con React: 90 horas
- Despliegue: 10 horas
- Pruebas automáticas y testing manual: 25 horas
- Documentación y elaboración de diagramas: 25 horas
- Diseño y validación de modelos geográficos y base de datos: 35 horas
- Otros (pruebas de estrés, refactorizaciones, correcciones, etc.): 20 horas

Diego Raúl Roldán Urueña

Esta distribución es aproximada, pero refleja el equilibrio entre implementación técnica, validación del sistema y redacción de la documentación necesaria para garantizar la comprensión y el mantenimiento del proyecto.

## 6.4 Análisis de riesgos

Desde el principio, se identificaron ciertos riesgos técnicos que podían afectar al rendimiento o a la viabilidad del sistema en producción. Los principales fueron:

- Escalabilidad del backend frente al envío de imágenes grandes en base64.
- Gestión de sesiones y tokens con OAuth 2.0 de Google.
- Posible complejidad en el manejo de regiones cuando estas contienen miles de celdas H3.

A medida que avanzó el desarrollo, se tomaron decisiones técnicas para mitigar estos riesgos, aunque algunas de ellas —como el rediseño del sistema de imágenes o el uso de almacenamiento externo— se han dejado documentadas como mejoras necesarias para una futura versión pública del sistema.



## 7 Conclusiones

Resumir los resultados/aportaciones principales del proyecto. Dar ideas de por dónde podría seguir el trabajo. Se puede incluir una valoración personal del proyecto y una descripción de las incidencias y problemas encontrados y resueltos durante el mismo.

Debe destacarse que Locatopedia en este trabajo constituye una primera versión funcional de la aplicación, pero no representa el alcance completo del sistema que podría construirse a partir de esta base. La aplicación ha sido diseñada de forma modular y escalable, permitiendo su futura expansión en múltiples direcciones.

Una de las ampliaciones más relevantes sería la incorporación de edición colaborativa en tiempo real, la cual permitiría que varios usuarios modifiquen simultáneamente un mismo documento o artículo, con sincronización inmediata de los cambios. Esto podría lograrse mediante tecnologías como WebSockets o servicios como Firebase, y requeriría un control de concurrencia robusto para evitar conflictos y garantizar la consistencia de los datos.

Otra funcionalidad valiosa sería la integración de sistemas de discusión o comentarios asociados a cada elemento de contenido. De una forma similar a la sección de discusión de Wikipedia, los usuarios podrían intercambiar ideas, hacer sugerencias o plantear dudas directamente en el contexto del artículo correspondiente, fomentando así una mayor interacción y enriquecimiento colectivo del conocimiento.

Por otra parte, el uso de técnicas de inteligencia artificial para la clasificación automática de contenido permitiría etiquetar artículos, identificar temas principales o incluso sugerir agrupaciones temáticas, reduciendo la carga manual de organización y mejorando la experiencia de navegación. Esto podría implementarse mediante modelos de procesamiento de lenguaje natural (NLP), entrenados para analizar los textos e inferir categorías relevantes.

Además, se ha considerado la posibilidad de mejorar los mecanismos de búsqueda espacial, no solo mediante la optimización de índices geográficos, sino también integrando filtros contextuales o visualizaciones interactivas en mapas, que permitirían explorar los contenidos de manera más intuitiva y eficiente según la ubicación del usuario.

## 8 Bibliografía

Las citas bibliográficas que se hayan hecho en el texto. Numeradas consecutivamente según se indica en las recomendaciones de la EINA.

<https://www.uber.com/en-DE/blog/h3/>  
<https://leafletjs.com/>  
<https://h3geo.org>  
<http://s2geometry.io>  
<https://pypi.org/project/rHEALPixDGGS/>  
[https://raichev.net/files/rhealpix\\_dggs\\_preprint.pdf](https://raichev.net/files/rhealpix_dggs_preprint.pdf)

## 9 Anexos

Manual de administración si tiene sentido.

Manual de despliegue/instalación si tiene sentido.

Manual de usuario si tiene sentido.

Diagramas detallados que en el cuerpo principal de la memoria eran demasiado.

Resultados detallados de pruebas que en el cuerpo principal de la memoria eran demasiado.

Documentación completa de la/s API más relevantes.

Etc.

100 vus	
100	<p>TOTAL RESULTS</p> <p>checks_total.....: 100 78.927971/s  checks_succeeded.....: 100.00% 100 out of 100  checks_failed.....: 0.00% 0 out of 100</p> <p>✓ status is 201</p> <p>HTTP</p> <p>http_req_duration.....: avg=543.78ms  min=311.32ms med=575.94ms max=1.08s p(90)=636.65ms p(95)=1.05s  { expected_response:true }.....: avg=543.78ms  min=311.32ms med=575.94ms max=1.08s p(90)=636.65ms p(95)=1.05s  http_req_failed.....: 0.00% 0 out of 100  http_reqs.....: 100 78.927971/s</p> <p>EXECUTION</p> <p>iteration_duration.....: avg=573.51ms  min=323.39ms med=589.16ms max=1.23s p(90)=724.9ms p(95)=1.07s  iterations.....: 100 78.927971/s  vus.....: 6 min=6 max=6  vus_max.....: 100 min=100  max=100</p> <p>NETWORK</p> <p>data_received.....: 480 kB 379 kB/s  data_sent.....: 34 MB 27 MB/s</p> <p>running (00m01.3s), 000/100 VUs, 100 complete and 0 interrupted iterations</p> <p>default ✓ [=====] 100 VUs</p>

	00m01.3s/10m0s 100/100 shared iters
500	<p>TOTAL RESULTS</p> <p>checks_total.....: 500 77.45441/s  checks_succeeded.....: 100.00% 500 out of 500  checks_failed.....: 0.00% 0 out of 500</p> <p>✓ status is 201</p> <p>HTTP</p> <p>http_req_duration.....: avg=2.78s  min=37.85ms med=2.31s max=5.73s p(90)=5.15s p(95)=5.44s  { expected_response:true }.....: avg=2.78s  min=37.85ms med=2.31s max=5.73s p(90)=5.15s p(95)=5.44s  http_req_failed.....: 0.00% 0 out of 500  http_reqs.....: 500 77.45441/s</p> <p>EXECUTION</p> <p>iteration_duration.....: avg=3s min=164.3ms  med=2.32s max=6.47s p(90)=5.62s p(95)=6.12s  iterations.....: 500 77.45441/s  vus.....: 44 min=44 max=492  vus_max.....: 500 min=500  max=500</p> <p>NETWORK</p> <p>data_received.....: 2.4 MB 372 kB/s  data_sent.....: 171 MB 27 MB/s</p>
2000	<p>TOTAL RESULTS</p> <p>checks_total.....: 2000 75.277839/s  checks_succeeded.....: 100.00% 2000 out of 2000  checks_failed.....: 0.00% 0 out of 2000</p> <p>✓ status is 201</p> <p>HTTP</p> <p>http_req_duration.....: avg=1.29s  min=240.64ms med=1.32s max=2.41s p(90)=1.57s p(95)=1.7s  { expected_response:true }.....: avg=1.29s  min=240.64ms med=1.32s max=2.41s p(90)=1.57s p(95)=1.7s  http_req_failed.....: 0.00% 0 out of 2000  http_reqs.....: 2000 75.277839/s</p> <p>EXECUTION</p> <p>iteration_duration.....: avg=1.31s  min=264.07ms med=1.33s max=2.42s p(90)=1.59s p(95)=1.71s  iterations.....: 2000 75.277839/s</p>

	<p>vus.....: 66    min=66    max=100  vus_max.....: 100    min=100  max=100</p> <p>NETWORK  data_received.....: 9.6 MB 362 kB/s  data_sent.....: 685 MB 26 MB/s</p> <p>running (00m26.6s), 000/100 VUs, 2000 complete and 0 interrupted iterations  default    ✓    [=====]    100    VUs  00m26.7s/10m0s 2000/2000 shared iters</p>
5000	<p>TOTAL RESULTS</p> <p>checks_total.....: 5000 60.500031/s  checks_succeeded.....: 100.00% 5000 out of 5000  checks_failed.....: 0.00% 0 out of 5000</p> <p>✓ status is 201</p> <p>HTTP  http_req_duration.....:                    avg=1.62s  min=304.59ms med=1.68s max=2.87s p(90)=1.96s p(95)=2.13s  { expected_response:true }.....:                    avg=1.62s  min=304.59ms med=1.68s max=2.87s p(90)=1.96s p(95)=2.13s  http_req_failed.....: 0.00% 0 out of 5000  http_reqs.....: 5000 60.500031/s</p> <p>EXECUTION  iteration_duration.....:                    avg=1.64s  min=308.56ms med=1.69s max=2.88s p(90)=1.98s p(95)=2.15s  iterations.....: 5000 60.500031/s  vus.....: 60    min=60    max=100  vus_max.....: 100    min=100  max=100</p> <p>NETWORK  data_received.....: 24 MB 291 kB/s  data_sent.....: 1.7 GB 21 MB/s</p> <p>running (01m22.6s), 000/100 VUs, 5000 complete and 0 interrupted iterations</p>

	default ✓ [=====] 100 VUs 01m22.9s/10m0s 5000/5000 shared iters
1000 vus	
1000	<p>TOTAL RESULTS</p> <p>checks_total.....: 1000 51.685964/s  checks_succeeded.....: 98.90% 989 out of 1000  checks_failed.....: 1.09% 11 out of 1000</p> <p>X status is 201  ↳ 98% — ✓ 989 / X 11</p> <p>HTTP  http_req_duration.....: avg=8.6s  min=546.16ms med=9.79s max=15.97s p(90)=12.44s p(95)=12.71s  { expected_response:true }.....: avg=8.55s  min=546.16ms med=9.79s max=15.96s p(90)=12.38s p(95)=12.7s  http_req_failed.....: 1.09% 11 out of 1000  http_reqs.....: 1000 51.685964/s</p> <p>EXECUTION  iteration_duration.....: avg=9.32s  min=572.7ms med=9.83s max=17.18s p(90)=15.42s p(95)=16.5s  iterations.....: 1000 51.685964/s  vus.....: 12 min=12 max=1000  vus_max.....: 1000 min=1000  max=1000</p> <p>NETWORK  data_received.....: 4.7 MB 246 kB/s  data_sent.....: 343 MB 18 MB/s</p> <p>running (00m19.3s), 0000/1000 VUs, 1000 complete and 0 interrupted iterations  default ✓ [=====] 1000 VUs  00m19.3s/10m0s 1000/1000 shared iters</p>
5000	<p>TOTAL RESULTS</p> <p>checks_total.....: 5000 42.012529/s  checks_succeeded.....: 19.13% 957 out of 5000  checks_failed.....: 80.86% 4043 out of 5000</p> <p>X status is 201</p>

	<p>↳ 19% — ✓ 957 / ✗ 4043</p> <p>HTTP</p> <p>http_req_duration.....: avg=4.79s min=0s med=0s max=25.84s p(90)=20.02s p(95)=23.39s { expected_response:true }.....: avg=9.25s min=2.52s med=9.16s max=21.73s p(90)=19.44s p(95)=20.42s http_req_failed.....: 80.86% 4043 out of 5000 http_reqs.....: 5000 42.012529/s</p> <p>EXECUTION</p> <p>iteration_duration.....: avg=22.06s min=894.99ms med=30s max=31.98s p(90)=30.22s p(95)=30.69s iterations.....: 5000 42.012529/s vus.....: 55 min=55 max=1000 vus_max.....: 1000 min=1000 max=1000</p> <p>NETWORK</p> <p>data_received.....: 4.6 MB 39 kB/s data_sent.....: 671 MB 5.6 MB/s</p> <p>running (01m59.0s), 0000/1000 VUs, 5000 complete and 0 interrupted iterations default ✓ [=====] 1000 VUs 01m59.4s/10m0s 5000/5000 shared iters</p>
2000	<p>TOTAL RESULTS</p> <p>checks_total.....: 2000 117.716328/s checks_succeeded.....: 60.15% 1203 out of 2000 checks_failed.....: 39.85% 797 out of 2000</p> <p>✗ status is 201 ↳ 60% — ✓ 1203 / ✗ 797</p> <p>HTTP</p> <p>http_req_duration.....: avg=7.22s min=697.23ms med=7.15s max=13.73s p(90)=12.38s p(95)=13.46s { expected_response:true }.....: avg=6.11s min=697.23ms med=6.02s max=11.58s p(90)=10.71s p(95)=11.42s http_req_failed.....: 39.85% 797 out of 2000 http_reqs.....: 2000 117.716328/s</p> <p>EXECUTION</p>

iteration_duration.....:	avg=7.71s
min=794.56ms med=7.22s max=17.04s p(90)=15.5s p(95)=15.7s	
iterations.....: 2000	117.716328/s
vus.....: 797	min=797
max=1000	
vus_max.....: 1000	min=1000
max=1000	
NETWORK	
data_received.....: 5.8 MB	340 kB/s
data_sent.....: 685 MB	40 MB/s
running (00m17.0s), 0000/1000 VUs, 2000 complete and 0 interrupted iterations	
default ✓ [=====]	1000 VUs
00m17.1s/10m0s 2000/2000 shared iters	