ReadMEasy

Pablo Vacas García

Índice de contenidos

Abstract

Español:

ReadMEasy es una herramienta diseñada para simplificar y mejorar la creación de archivos README para repositorios y perfiles en GitHub. Su objetivo es proporcionar una plataforma que permita a los desarrolladores generar READMEs atractivos, informativos y personalizados. Entre sus funcionalidades principales se incluyen un sistema de usuarios con opciones de registro e inicio de sesión (correo, Google, GitHub), plantillas personalizadas, edición interactiva en tiempo real, y explorar otros usuarios y READMEs con un sistema de seguidores y "me gusta".

English:

ReadMEasy is a tool designed to simplify and enhance the creation of README files for GitHub repositories and profiles. Its goal is to provide a platform that allows developers to generate attractive, informative, and customized READMEs. Key features include a user system with registration and login options (email, Google, GitHub), customized templates, real-time interactive editing, and exploration of other users and READMEs with a followers and likes system.

Justificación

Características Generales

ReadMEasy es una herramienta web que facilita la creación y personalización de archivos README para repositorios y perfiles en GitHub. Su objetivo principal es ofrecer una plataforma intuitiva y rica en funcionalidades que permita a los desarrolladores generar READMEs atractivos, informativos y personalizados. Las características clave incluyen:

- Sistema de Usuarios: Registro e inicio de sesión mediante correo electrónico, Google y GitHub.
- Plantillas Personalizadas: Una variedad de plantillas predefinidas, para las tecnologías más usadas, con opciones de personalización.
- Edición Interactiva: Interfaz intuitiva para la edición en tiempo real de los README, con opciones para agregar diferentes apartados.
- Búsqueda de Usuarios y READMEs: Sistema de seguidores y "me gusta" para fomentar la interacción y descubrimiento de otros usuarios y sus READMEs.
- Aplicación Instalable: Posibilidad de instalar la aplicación en todos los dispositivos como PWA.

Restricciones Generales

- Conectividad a Internet: Requiere conexión a internet para acceder a las funcionalidades basadas en la nube y a la autenticación mediante Google y GitHub.

- Secciones limitadas: Las secciones disponibles, pese a ser bastantes y las más comunes, algún usuario podría necesitar secciones adicionales, por lo que se aceptarán sugerencias para futuras implementaciones.
- Plantillas limitadas: Al igual que las secciones, se han implementado las tecnologías más usadas, existirán usuario interesados en otras distintas, hay plantilla para proyectos sin una tecnología específica y se trabajará por cubrir más tecnologías.
- Conexión a internet: Al utilizar una base de datos online, en Firebase, no se podrá utilizar sin conexión.

Aspectos a Cubrir y No Tratar

Aspectos a Cubrir:

- Banco de trabajo sencillo e intuitivo, no se deberá conocer ni tratar el código MarkDown de forma directa.
- Posibilidad de almacenar y editar múltiples READMEs para usuarios registrados.
- Descargar los README en archivo .md para su integración en GitHub.
- Proveer plantillas y herramientas de personalización para README.
- Implementar un sistema de usuarios robusto y seguro.
- Ofrecer una interfaz de usuario interactiva y amigable.
- Facilitar la exploración y colaboración entre usuarios a través de seguidores y "me gusta".
- Ofrecer la posibilidad de instalar como PWA para facilitar el uso y acceso.
- Optimización SEO: Se utilizó la consola de búsqueda de google y se usaron prácticas “SEO” para mostrar la página en las búsquedas.

No se va a tratar:

- No se cubrirá la gestión de otros tipos de archivos o documentación fuera de los README.
- No se proporcionarán herramientas de análisis de código o documentación técnica avanzada.

Estudio de Prestaciones frente a Herramientas Existentes

Comparativa con Herramientas Existentes:

readme.so:

Pros: Generación rápida de README con formato básico.
Contras: Falta de personalización avanzada y funcionalidades interactivas.
Ventaja de READMEasy: Provee plantillas personalizadas, edición en tiempo real y un sistema de interacción entre usuarios.

StackEdit:

Pros: Editor de Markdown robusto con vista previa en tiempo real.

Contras: No está específicamente diseñado para READMEs y carece de plantillas predefinidas para GitHub.
Ventaja de ReadMEasy: Enfocado en READMEs de GitHub, con plantillas específicas y personalización visual.

Profile readme generator:

Pros: Fácil de usar para principiantes.
Contras: Opciones de personalización limitadas, sin edición interactiva avanzada.
Ventaja de ReadMEasy: Mayor personalización y edición interactiva, con funciones sociales para explorar y seguir a otros usuarios.

Make a README:

Pros: Vista previa en tiempo real.
Contras: Necesidad de saber MarkDown y sin plantillas predefinidas.
Ventaja de ReadMEasy: No necesitas saber ni conocer MarkDown, plantillas para cubrir múltiples tecnologías.

ReadMEasy se destaca por la facilidad de edición para principiantes, su enfoque en la personalización avanzada y la interacción social, aspectos que no están completamente cubiertos por otras herramientas existentes en el mercado. Esto la hace especialmente valiosa para desarrolladores que buscan crear README no solo funcionales, sino también visualmente atractivos y personalizados para sus proyectos en GitHub.

Justificación de la Tecnología Empleada

ReadMEasy emplea tecnologías modernas y robustas para garantizar una experiencia de usuario óptima y un desarrollo eficiente. A continuación, se enumeran y describen las tecnologías elegidas, junto con la justificación de su uso:

Vue.js

Descripción:

Vue.js es un framework progresivo de JavaScript utilizado para construir interfaces de usuario. Se centra en la capa de vista del Modelo-Vista-Controlador (MVC) y es fácil de integrar con otros proyectos y bibliotecas existentes.

Justificación:

- Facilidad de Uso: Vue.js es conocido por su curva de aprendizaje suave y su simplicidad, lo que facilita su adopción tanto para desarrolladores principiantes como experimentados.
- Componentes Reutilizables: Permite la creación de componentes reutilizables, lo que mejora la organización del código y la mantenibilidad del proyecto.

- Rendimiento: Vue.js es ligero y ofrece un rendimiento excelente, asegurando que la aplicación sea rápida y responsiva.
- Ecosistema Rico: Posee una amplia gama de herramientas y bibliotecas que complementan el desarrollo, se han utilizado Vue Router para enrutamiento y Vuex para gestión del estado y Vue Draggable para la función de arrastrar y soltar.
- Comunidad y Soporte: Tiene una comunidad activa y una gran cantidad de recursos disponibles, lo que facilita la resolución de problemas y la implementación de nuevas características.

Bootstrap y SASS

Descripción:

Bootstrap es un framework CSS de código abierto que facilita el desarrollo de sitios web y aplicaciones responsivas. SASS (Syntactically Awesome Stylesheets) es un preprocesador CSS que añade funcionalidades avanzadas como variables, anidamiento y mixins, lo que permite escribir CSS más eficiente y modular.

Justificación:

- Consistencia y Estética: Bootstrap proporciona una base sólida para el diseño responsivo, asegurando que la aplicación se vea bien en cualquier dispositivo.
- Personalización: Utilizando SASS, podemos personalizar Bootstrap según las necesidades específicas del proyecto, manteniendo un diseño coherente y único.
- Reusabilidad: Al igual que Vue, Bootstrap permite la creación de estilos y componentes reutilizables en varios apartados e, incluso, en otros futuros proyectos.
- Productividad: Bootstrap incluye una amplia variedad de componentes predefinidos que aceleran el desarrollo, mientras que SASS facilita la gestión y la modularización del CSS.
- Comunidad y Documentación: Bootstrap cuenta con una extensa documentación y una comunidad activa, lo que facilita su uso y personalización.

Firebase

Descripción:

Firebase es una plataforma de desarrollo de aplicaciones móviles y web proporcionada por Google. Ofrece una variedad de servicios, incluidos base de datos en tiempo real, autenticación, almacenamiento de archivos y hosting.

Justificación:

- Autenticación: Firebase Authentication facilita la implementación de métodos de inicio de sesión seguros y variados (correo, Google, GitHub), simplificando la gestión de usuarios.
- Base de Datos en Tiempo Real: Firestore, la base de datos en tiempo real de Firebase, permite sincronizar datos entre clientes en tiempo real, mejorando la interactividad de la aplicación.

- Almacenamiento de Archivos: Firebase Storage proporciona una solución escalable y segura para almacenar y servir archivos de usuario, como imágenes y documentos.
- Hosting: Firebase Hosting ofrece un hosting rápido y seguro para aplicaciones web, con despliegue fácil y soporte para HTTPS.
- Escalabilidad: Firebase es altamente escalable y maneja automáticamente el aumento de la carga sin necesidad de intervención manual.
- Integración: La integración de Firebase con otros servicios de Google y su facilidad de uso lo convierten en una opción excelente para proyectos que requieren una rápida implementación y escalabilidad.

Proceso de Investigación y Justificación

Durante la fase de investigación, se evaluaron diversas tecnologías para determinar las más adecuadas para ReadMEasy. Las decisiones se basaron en los siguientes criterios:

- Facilidad de Uso y Curva de Aprendizaje: Vue.js y Firebase fueron seleccionados por su simplicidad y su curva de aprendizaje manejable, lo que permite un desarrollo más rápido y eficiente.
- Documentación y Comunidad: La disponibilidad de documentación extensa y el apoyo de una comunidad activa fueron factores clave en la elección de estas tecnologías.
- Flexibilidad y Personalización: La capacidad de personalizar Bootstrap con SASS y la flexibilidad de Vue.js para crear componentes reutilizables fueron decisivos para asegurar un diseño único y coherente.
- Rendimiento y Escalabilidad: Firebase ofrece soluciones escalables y de alto rendimiento para la autenticación, almacenamiento y base de datos, esenciales para una aplicación web dinámica y en tiempo real.

En resumen, la combinación de Vue.js, Bootstrap personalizado con SASS y Firebase proporciona una base sólida y eficiente para desarrollar ReadMEasy, asegurando una experiencia de usuario óptima y facilitando el mantenimiento y escalabilidad del proyecto.

Requerimientos hardware y software

Requerimientos de Software

Vue.js:

- Versión: Se recomienda utilizar la versión estable más reciente de Vue.js.
- Entorno de Desarrollo: Node y npm (Node Package Manager) para la gestión de paquetes.
- Navegadores Compatibles: Compatible con los principales navegadores web como Chrome, Firefox, Safari y Edge.

SASS:

- Instalación de Node.js y npm para compilar archivos SASS a CSS.
- Un compilador SASS como node-sass o dart-sass.

Bootstrap:

- Incluir las dependencias de Bootstrap en el proyecto Vue.js mediante npm.
- Configurar y compilar los estilos personalizados utilizando SASS.

Firebase:

- Una cuenta de Google para acceder a Firebase Console y configurar los servicios.
- Conexión a internet para el uso de servicios en la nube de Firebase.

Requerimientos de Hardware

Los requerimientos de hardware son mínimos y dependen principalmente del entorno de desarrollo y la capacidad de conexión a internet para utilizar Firebase:

Para Desarrollo:

- Computadora con al menos 4 GB de RAM y un procesador dual-core.
- Espacio suficiente en disco para instalar Node.js y las dependencias del proyecto.

Para Implementación y Hosting:

La plataforma de hosting de Firebase maneja la infraestructura, por lo que no se requieren especificaciones adicionales de hardware. Firebase se encarga de escalar automáticamente según las necesidades del proyecto.

Resumen:

ReadMEasy utiliza tecnologías modernas que son compatibles con los sistemas operativos principales (Windows, macOS, Linux) y los navegadores web estándar. Los requerimientos de software incluyen un editor de código como VS Code, Node y npm para Vue.js y SASS, junto con una cuenta de Google para configurar y administrar Firebase. En términos de hardware, se recomienda una computadora con al menos 4 GB de RAM y un procesador dual-core para el desarrollo y prueba del proyecto.

Análisis y diseño

Diagrama de casos de uso

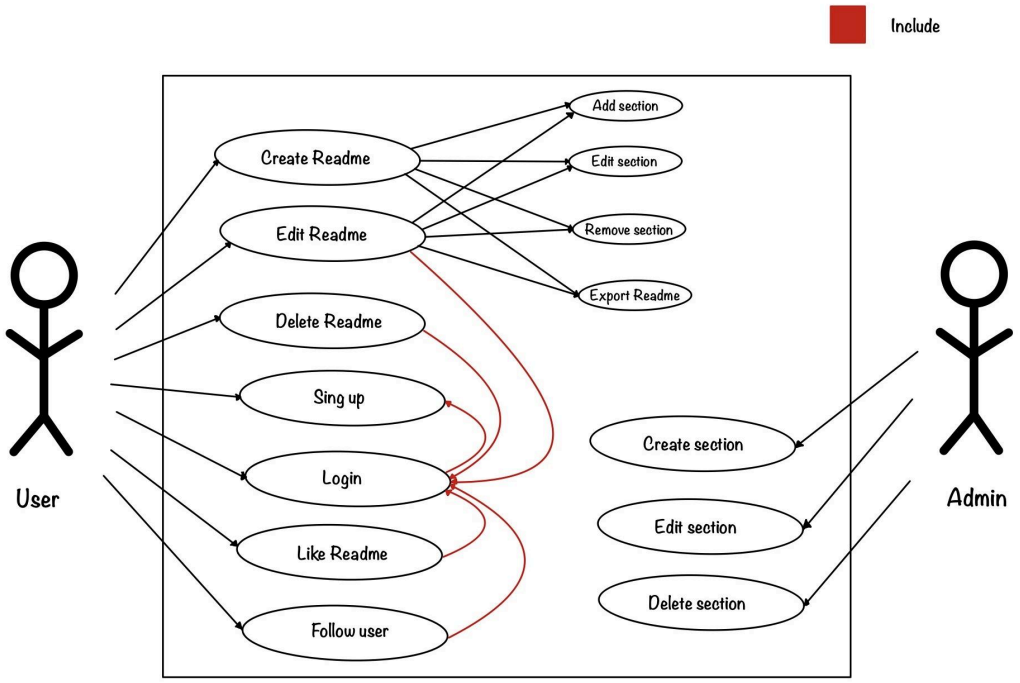

Imagen 1: Diagrama de casos de uso

Base de datos

Para el proyecto ReadMEasy se ha elegido Firebase Firestore como la base de datos no relacional. A continuación, se justifica esta elección:

- Flexibilidad en el Esquema de Datos:

  Firebase Firestore es una base de datos NoSQL que permite almacenar datos en documentos JSON, lo cual ofrece flexibilidad en la estructura de los datos. Esto es especialmente útil en aplicaciones como ReadMEasy, donde los datos pueden variar en su formato y no necesariamente siguen un esquema estricto como en las bases de datos relacionales.

- Escalabilidad Horizontal:

  Firestore está diseñado para escalar automáticamente según las necesidades del proyecto. Puede manejar grandes volúmenes de datos y garantizar un rendimiento constante a medida que la aplicación crece, sin necesidad de intervención manual en la configuración de la base de datos.

- Modelo de Datos Basado en Documentos:

Utiliza un modelo de datos basado en documentos y colecciones. Cada documento contiene un conjunto de pares clave-valor, similar a los documentos JSON, lo cual simplifica la manipulación y el almacenamiento de datos complejos y anidados.

- Velocidad y Rendimiento:

  Firestore ofrece operaciones de lectura y escritura rápidas y eficientes, lo que es crucial para aplicaciones web en tiempo real como ReadMEasy. La replicación de datos global y la baja latencia aseguran que los usuarios puedan acceder y modificar sus datos rápidamente desde cualquier parte del mundo.

- Integración con Otros Servicios de Firebase:

  Firebase Firestore se integra perfectamente con otros servicios de Firebase como la autenticación de usuarios, el almacenamiento de archivos y el hosting. Esto simplifica el desarrollo y la gestión de la aplicación al tener todos los servicios bajo la misma plataforma.

Descripción de los datos de los documentos:

Debido a la estructura de datos deseada, se necesitará que la base de datos sea flexible y sobre todo con buen rendimiento, para hacer múltiples consultas simultáneas, por lo que el modelo de base de datos escogido será una base de datos no relacional, cuyas colecciones serán:

- User: Esta colección representa a cada uno de los usuarios registrados en la plataforma, cada usuario tendrá un id único, nombre de usuario, nombre, contraseña y seguidores, que será una lista con los ids de los usuarios que le siguen.
- ReadME: Esta colección representa los ReadMEs generados por los usuarios. Esta tendrá id, título, contenido, que será el código del ReadME y likes, que será una lista con los ids de los usuarios que le han dado me gusta.
- Section: Esta colección representa las secciones predeterminadas disponibles para incluir en los ReadMEs de los usuarios, tendrá id, nombre y contenido.

Implementación

Hojas de Estilo

Se utilizará Bootstrap personalizado con SASS para gestionar las hojas de estilo de la aplicación. Bootstrap ofrece un conjunto de componentes y estilos predefinidos que se pueden personalizar utilizando SASS para adaptarse al diseño específico de ReadMEasy.

Plantillas

Se utilizó una plantilla de Bootstrap específicamente diseñada para implementar en proyectos de Vue.js como base, llamada Vue Argon Design System diseñada por Creative Tim.

La elección de esta plantilla se debe al diseño minimalista y profesional que ofrece y por su semejanza en colores, tipografía y diseño con GitHub, lo que concuerda perfectamente con la idea de ReadMEasy, aunque se hicieron modificaciones y retoques para una mayor personalización.

Formularios

Se implementarán formularios interactivos para permitir a los usuarios crear y editar sus README, para inicio de sesión y registro de usuarios, usando Firebase Authentication y para edición del perfil. Estos formularios estarán conectados a Firebase Firestore para el almacenamiento y recuperación de datos.

Conexión y Consultas a la Base de Datos

Firebase Firestore se utilizará como la base de datos principal para almacenar todos los datos relacionados con los usuarios y sus READMEs. Se realizarán consultas y actualizaciones en tiempo real para garantizar que la información mostrada en la aplicación sea siempre precisa y actualizada.

Uso de Fichero de Configuración

Se emplea un archivo de configuración para gestionar aspectos importantes como las credenciales de Firebase y otras configuraciones de la aplicación.
Package.json para indicar las dependencias y sus respectivas versiones para su correcto funcionamiento.
Varios formatos de sitemap, con la estructura interna de la web para mejorar el SEO.
Un sitemap.webmanifest y manifest.json para ayudar al navegador a identificar los elementos de la aplicación y optimizar el SEO.

Evaluación y prueba

El proceso de testeo de ReadMEasy se ha dividido en varias etapas para asegurar la funcionalidad y la usabilidad de la aplicación web:

- Pruebas de Integración:
  Se han llevado a cabo pruebas de integración para verificar la interacción adecuada entre diferentes componentes de la aplicación, incluyendo formularios, plantillas y la conexión con Firebase.

- Pruebas Funcionales:
  Se han realizado pruebas funcionales para evaluar la funcionalidad completa de la aplicación desde el punto de vista del usuario final. Esto incluye la creación de READMEs, la personalización de plantillas, la edición en tiempo real y la interacción social como seguir a otros usuarios y dar "me gusta" a READMEs.

- Pruebas de Interfaz de Usuario (UI):

Se han realizado pruebas de UI para asegurar que la interfaz sea intuitiva y fácil de usar para los usuarios. Se ha verificado la navegación entre secciones, la disposición de los elementos y la respuesta a diferentes tamaños de pantalla.

- Pruebas de Compatibilidad:
  Se han probado diferentes navegadores (Chrome, Firefox, Edge, Safari) y dispositivos (escritorio, tabletas y móviles) para asegurar que ReadMEasy funcione correctamente en todas las plataformas soportadas.

- Testeo de la Base de Datos:
  El testeo de Firebase Firestore se ha centrado en asegurar la integridad y la consistencia de los datos almacenados, así como la eficiencia en las operaciones de lectura y escritura:

- Validación de Datos:
  Se han implementado funciones de validación en los formularios para garantizar que los datos introducidos por los usuarios cumplan con los requisitos esperados. Por ejemplo, se verifica que los campos obligatorios estén llenos, que la contraseña sea segura y que los formatos de datos (como correos electrónicos o URLs) sean correctos.

- Realimentación al Usuario:
  Se ha proporcionado realimentación inmediata al usuario mediante mensajes de error o confirmación después de la interacción con formularios y acciones. Esto asegura una experiencia de usuario fluida y transparente.

Manual de estilos

Sketches

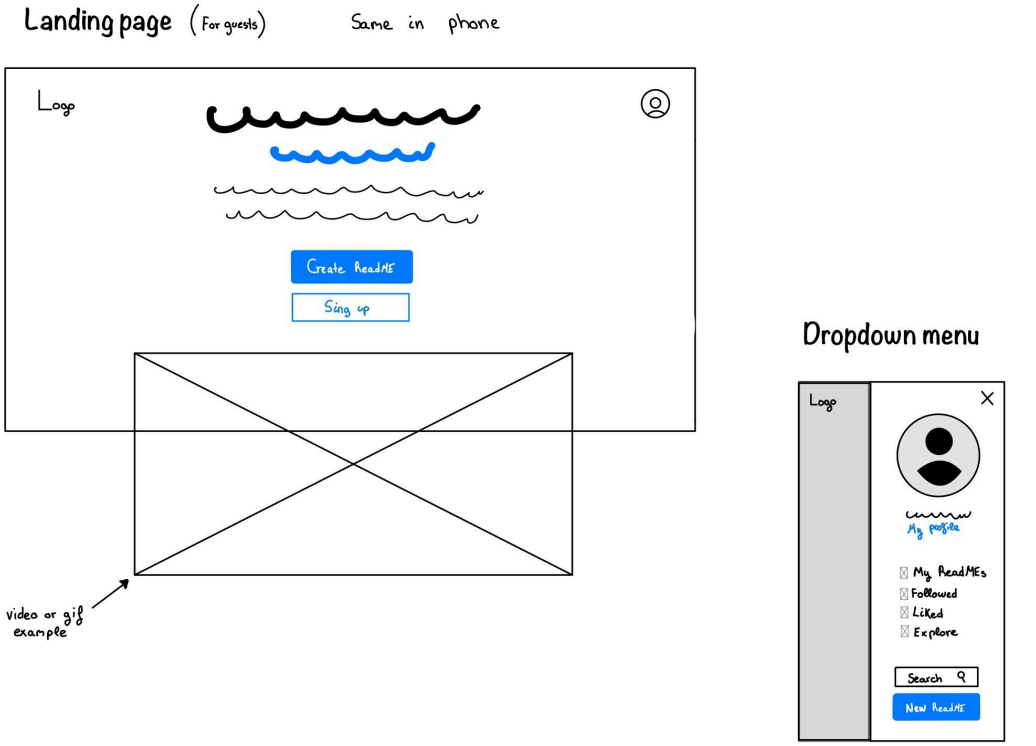

Imagen 2: Landing Page

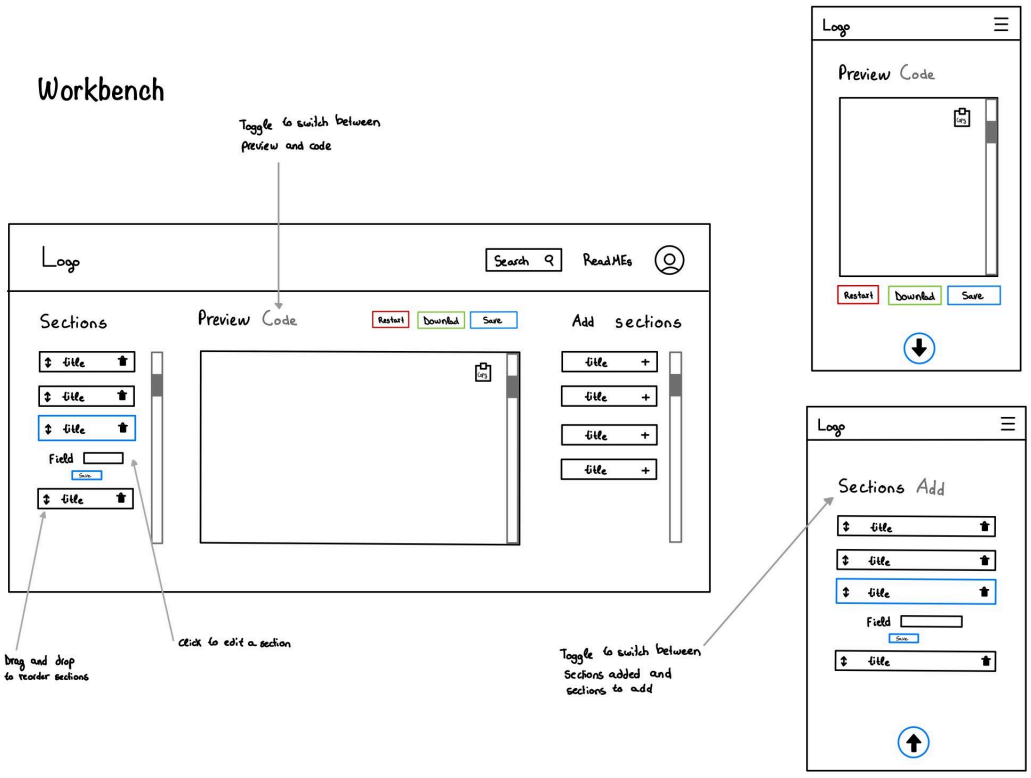

Imagen 3: Workbench

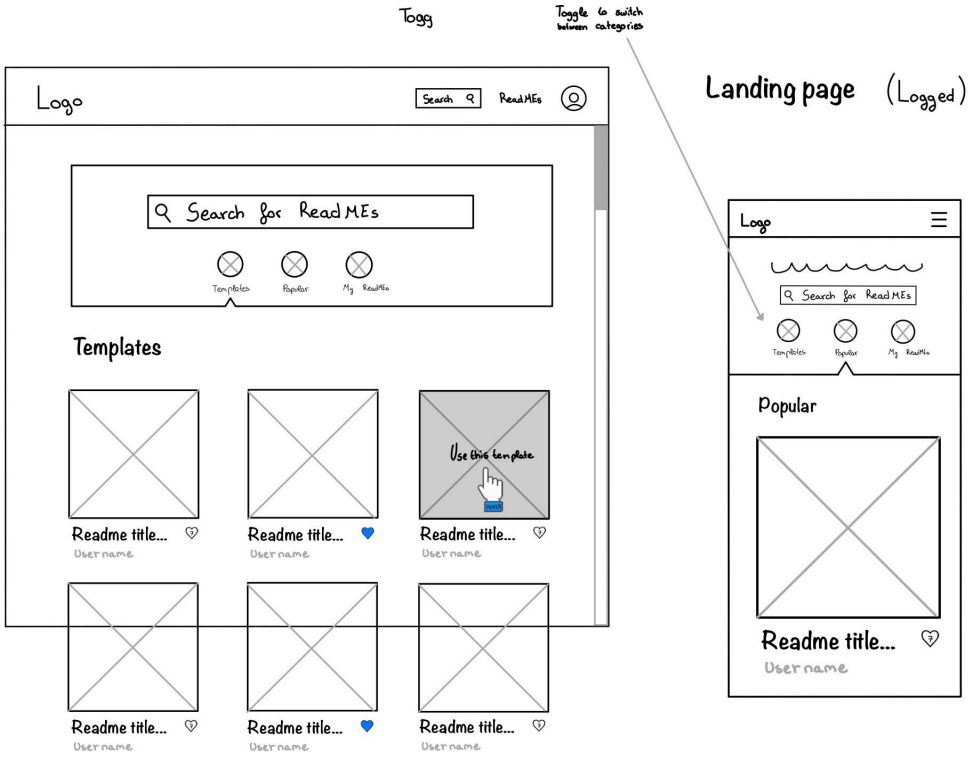

Imagen 4: Explore page

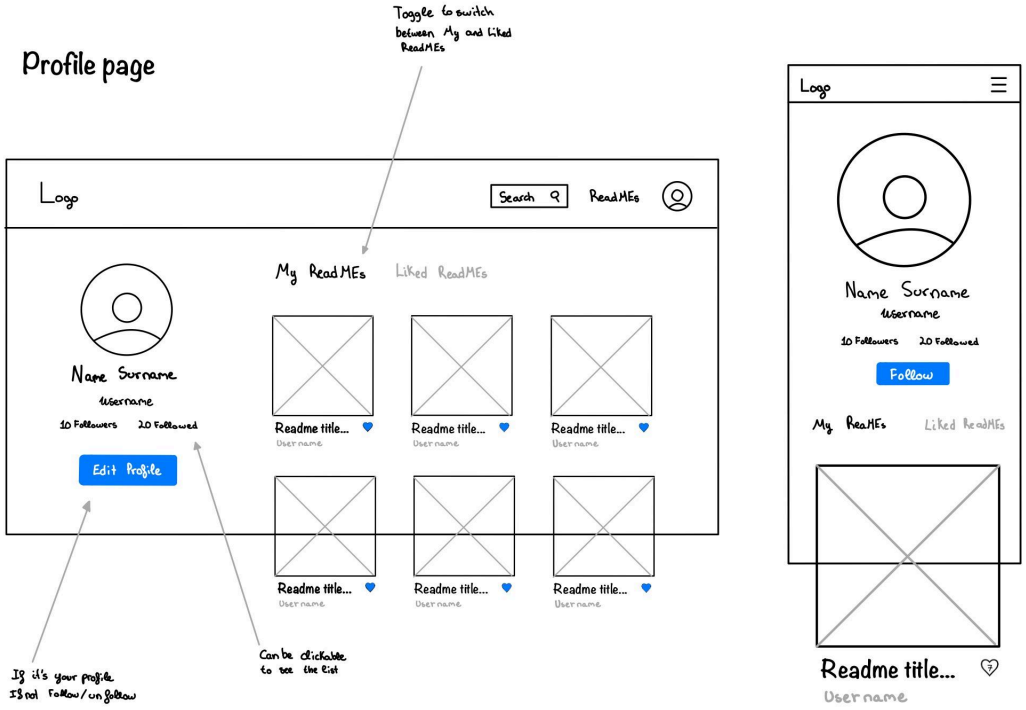

Imagen 5: Profile page

Criterios de accesibilidad

1. Navegación y Operabilidad:

    - Teclado: Asegurar que todas las funciones y acciones sean accesibles utilizando solo el teclado, sin depender de dispositivos de apuntado como el ratón.
    - Focus Visible: Garantizar que los elementos interactivos (botones, enlaces, formularios) tengan un enfoque visible y claro para los usuarios que navegan con teclado.

2. Contraste y Legibilidad:

    - Contraste de Color: Utilizar combinaciones de colores que cumplan con los estándares de contraste para asegurar que el texto sea legible para personas con baja visión o daltonismo.
    - Tamaño de Fuente: Proporcionar opciones para aumentar el tamaño del texto para mejorar la legibilidad.
    -

3. Compatibilidad con Tecnologías de Asistencia:

    - Screen Readers: Asegurar que todos los elementos de la interfaz de usuario sean accesibles mediante lectores de pantalla, utilizando etiquetas semánticas HTML (por ejemplo, alt para imágenes).
    - Compatibilidad con ARIA (Accessible Rich Internet Applications): Implementar atributos ARIA para mejorar la accesibilidad de componentes interactivos complejos como menús desplegables o carruseles.

4. Formularios y Validación:

    - Validación en el Lado del Cliente: Implementar validación de formularios en el lado del cliente para proporcionar realimentación inmediata al usuario sobre errores y requerimientos de entrada.
    - Mensajes de Error Claros: Utilizar mensajes de error descriptivos y claros para ayudar a los usuarios a corregir problemas en los formularios.

5. Imágenes y Multimedia:

    - Texto Alternativo (alt): Incluir texto alternativo significativo para todas las imágenes y gráficos utilizados en la aplicación.
    - Transcripciones y Subtítulos: Proporcionar transcripciones para contenido audiovisual como videos y subtítulos para facilitar el acceso a personas con discapacidad auditiva.

6. Diseño Responsivo: Asegurar que la aplicación sea completamente funcional y fácil de usar en una variedad de dispositivos y tamaños de pantalla, desde computadoras de escritorio hasta dispositivos móviles.

Criterios de usabilidad

1. Facilidad de Aprendizaje:

    - Intuitividad: La aplicación debe ser fácil de entender y utilizar desde la primera interacción.
    - Instrucciones claras: Proveer guías o instrucciones claras para nuevos usuarios sobre cómo usar las funcionalidades principales de la aplicación.

2. Eficiencia de Uso:

    - Interacción Eficiente: Minimizar la cantidad de pasos y clics necesarios para realizar tareas comunes como crear un nuevo README, editar contenido, o interactuar con otros usuarios.
    - Accesos Directos y Atajos: Implementar accesos directos de teclado y otras funcionalidades que permitan a los usuarios avanzados trabajar de manera más rápida.

3. Memoria:

    - Facilitar que los usuarios frecuentes puedan recordar cómo realizar tareas sin necesidad de volver a aprender cada vez que usan la aplicación.

4. Errores y Recuperación:

    - Prevención de Errores: Diseñar formularios y procesos que minimicen la posibilidad de errores.
    - Mensajes de Error Claros: Proporcionar mensajes de error descriptivos y sugerencias para la corrección de errores cuando sea necesario.

5. Satisfacción del Usuario:

    - Diseño Estético: Utilizar un diseño atractivo y coherente que mejore la experiencia visual del usuario.
    - Retroalimentación Positiva: Celebrar logros y acciones positivas de los usuarios, sistema de “likes” y seguidores.

6. Consistencia:

    - Consistencia Visual: Mantener un diseño y estilo coherentes en toda la aplicación para que los usuarios puedan predecir dónde encontrar opciones y acciones.
    - Consistencia Funcional: Garantizar que las acciones y funciones se comporten de manera predecible en diferentes partes de la aplicación.

7. Flexibilidad y Adaptabilidad:

    - Adaptabilidad: Asegurar que la aplicación funcione de manera eficiente en una variedad de dispositivos y tamaños de pantalla, utilizando un diseño responsivo.

8. Feedback de Usuario:

    - Interacción en Tiempo Real: Proporcionar retroalimentación inmediata sobre las acciones del usuario, como la actualización en tiempo real de la vista previa mientras se edita un README.

Tipografía

- Fuente principal: *'Open Sans', sans-serif*

- Fuente para READMEs (simula la tipografía de GitHub): *-apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji"*

Tamaños:

- base: 1rem
- xl: (base * 1.5)
- lg: (size-base * 1.25)
- sm: (size-base * .875)
- xs: (base * .75)

Grosor:

- light: 300
- normal: 400
- bold: 600

Títulos:

- h1: base * 2.5
- h2: base * 2
- h3: base * 1.75
- h4: base * 1.5
- h5: base * 1.25
- h6: base

Mapa de colores

La elección de la paleta de colores se ve muy influenciada por el propósito de la aplicación, por lo que los colores deben de recordar a GitHub, añadiendo un toque personal, por lo que los colores escogido y sus usos son:

Muestra	Nombre	RGB	Hex.	Uso
	White	255, 255, 255	#fff	Color de fondo
	Dark	33, 37, 41	#212529	Color de letra
	Primary	94, 114, 228	#5e72e4	Color primario para elementos a destacar
	Secondary	244, 245, 247	#f4f5f7	Bordes y sombras
	Success	45, 206, 137	#2dce89	Botones de guardar o descargar
	Danger	245, 54, 92	#f5365c	Errores y botones de eliminar
	Warning	251, 99, 64	#fb6340	Botón de rebobinar y avisos
	Info	17, 205, 239	#11cdef	Para botones y texto a resaltar

Dispositivos/vistas para las que se ha diseñado

ReadMEasy está diseñado para ser responsivo y adaptarse a cualquier tamaño de pantalla. Hay dos versiones diferenciadas, que cambian a los 992px (lg devices), como el menú o workbench, y para los demás puntos de ruptura se hacen pequeñas modificaciones para adaptarse.

Ejemplo:

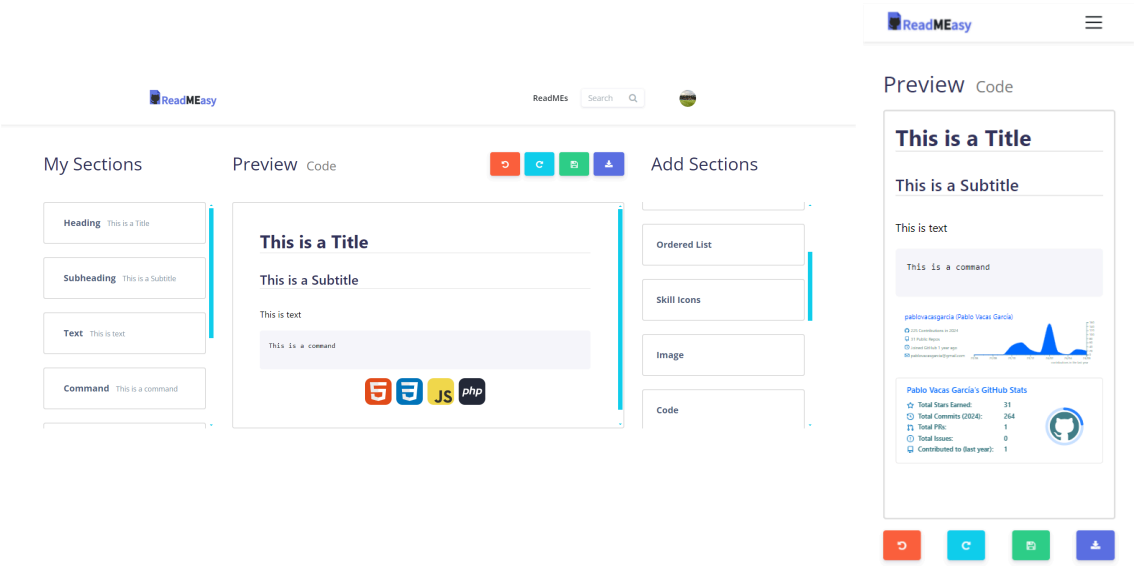

Imagen 6: Diseño Responsivo

Breakpoints: xs: 0, sm: 576px, md: 768px, lg: 992px, xl: 1200px

Software utilizado

Para la implementación de ReadMEasy se ha utilizado una variedad de software para diferentes propósitos. A continuación, se presenta una lista detallada del software utilizado y su función específica en el desarrollo de la aplicación:

- Visual Studio Code: IDE principal para el desarrollo de código frontend y backend.
- Node y npm (Node Package Manager): Plataforma de ejecución de JavaScript y gestor de paquetes para instalar bibliotecas y dependencias.
- GitHub: Control de versiones y probar los READMEs

Posibles mejoras y aportaciones

1. Añadir plantillas y secciones: Añadir progresivamente plantillas para más tecnologías y secciones disponibles en el banco de trabajo, para abarcar el máximo público posible.
2. Uso offline: Habilitar el uso de la aplicación (PWA) sin conexión a internet.
3. Elección de idioma y tema: Ofrecer la posibilidad de cambiar el idioma y tema oscuro o claro al usuario, para una mayor personalización.
4. Sistema de comentarios: Para que los usuarios den su opinión a otros usuarios sobre sus READMEs.

Bibliografía

Para la elaboración tanto del proyecto como de la propia documentación se han consultado recursos como:

- Stackoverflow: https://stackoverflow.com/
- Codepen: https://codepen.io/
- Vue Docs: https://vuejs.org/
- Bootstrap Docs: https://getbootstrap.com/
- SASS Docs: https://sass-lang.com/
- Vue Draggable Docs: https://github.com/SortableJS/Vue.Draggable
- Canva: https://www.canva.com/
- Figma: https://www.figma.com/
- Font Awesome: https://fontawesome.com/v4/icons/