

Capítulo V: Product Implementation, Validation & Deployment

5.1. Software Configuration Management.

En este punto del informe se describe las decisiones y los principios que ayudarán al equipo a garantizar la coherencia durante el desarrollo de la solución.

5.1.1. Software Development Environment Configuration.

En este apartado se proporcionan los enlaces a las aplicaciones y productos de software creados durante el ciclo del proyecto utilizando los programas correspondientes.

Con ese fin, se organizará en las siguientes secciones:

- ☐ Project Management
- ☐ Requirements Management
- ☐ Product UX/UI Design
- ☐ Software Development
- ☐ Software Testing
- ☐ Software Documentation

Asimismo, se clasificarán los elementos de estas secciones como rutas de referencia (para software basado en modelos SaaS) o rutas de descarga (para productos que se ejecuten en las computadoras de los miembros del equipo) para cada uno de los productos de software.

Project Management

Esta disciplina se fundamenta en la administración de proyectos y busca principalmente la mejora de procesos y su entorno con el propósito de lograr los resultados esperados.

- Durante el ciclo digital del proyecto, se llevará a cabo la implementación de un producto de software basado en el modelo SaaS, el cual funcionará a través de un navegador web; no obstante, no se desarrollará una versión de la aplicación móvil correspondiente.

Requirements Management:

Este proceso se enfoca en asegurar que una organización documente, verifique y satisfaga las necesidades y expectativas de sus clientes, así como las de las partes interesadas internas o externas.

- **Jira Software:** Esta es una plataforma que facilita la gestión de historias de usuario, organizándolas en epopeyas y evaluando su importancia en el programa según su prioridad y puntos de historia. Se utiliza debido a su capacidad para permitir que cada miembro del equipo tenga una vista en tiempo real de los avances en cada proyecto, contribuyendo con diferentes secciones o ajustando el flujo del proyecto según sea necesario.

Product UX/UI Design

Esta herramienta facilita la creación digital de modelos que se integran en la vida del consumidor.

En este caso, estamos desarrollando un modelo de sitio web compatible tanto con computadoras como con dispositivos móviles.

Para lograrlo, utilizamos varias herramientas de diseño y colaboración, que incluyen:

1. **Uxpressia:** Uxpressia es una plataforma en línea especializada en el mapeo de la trayectoria del cliente. Nos ayuda a crear mapas de impacto y perfiles de usuario, como User Personas, Empathy Maps y Journey Maps. Puedes encontrar más información sobre Uxpressia en este [enlace](#).
2. **MIRO:** MIRO es una pizarra digital colaborativa en línea que se adapta a diversas actividades colaborativas, como investigación, ideación, creación de lluvias de ideas y mapas mentales. Es una

herramienta versátil que facilita el trabajo en equipo. Descubre más sobre MIRO en su [sitio web](#).

3. **Figma:** Figma es una herramienta de prototipado web y un editor de gráficos vectoriales. A diferencia de otras herramientas, Figma se ejecuta en línea, lo que permite crear modelos que funcionan tanto en navegadores web como en navegadores móviles. Puedes explorar Figma en [este enlace](#).
4. **Lucid Chart:** Esta es una aplicación de diagramación en línea que permite a los usuarios colaborar y trabajar juntos en tiempo real para crear una variedad de diseños, incluidos diagramas UML, mapas mentales, prototipos de software y otros tipos de diagramas. Puedes conocer más acerca de Lucid Chart en [este enlace](#).
5. **Overflow:** Overflow es una herramienta de diagramación que ofrece la posibilidad de colaborar en tiempo real. Utilizamos esta herramienta para crear diagramas de Userflows. Si deseas obtener más información sobre Overflow, visita su [sitio web](#).

Estas herramientas nos ayudan a dar vida a nuestros diseños digitales y a garantizar que nuestros productos sean accesibles y atractivos en diferentes plataformas.

Software Development:

El desarrollo de software es una metodología aplicada en la creación de productos de software. Esta metodología se utiliza para establecer un proceso que guía el desarrollo del software, y cada uno de sus pasos describe un enfoque específico para las distintas actividades que ocurren durante el proceso.

Aquí te presentamos algunas herramientas y tecnologías clave que utilizaremos en el proyecto:

1. **GitHub:** GitHub es una plataforma de repositorio comunitario que se utiliza para almacenar y gestionar los avances de proyectos realizados por grupos de personas. Puedes acceder al repositorio del proyecto en este [enlace](#).
2. **Webstorm:** Webstorm es un entorno de desarrollo de JetBrains, una empresa especializada en software, orientado al desarrollo web en JavaScript. Esta herramienta proporciona facilidades para probar sitios web en navegadores como Google Chrome. En nuestro proyecto, utilizaremos webstorm para trabajar con lenguajes como HTML, CSS y JavaScript. Obtén más información sobre WebStorm [aquí](#).
3. **HTML:** HTML es un lenguaje de marcado que se utiliza en el desarrollo de sitios web para crear hipertextos y enlazar a otros documentos. Este lenguaje proporciona herramientas para diseñar sitios web y se puede combinar eficazmente con CSS y JavaScript. En nuestro proyecto, utilizaremos HTML para implementar la documentación de la página web. Obtén más información sobre la edición de archivos HTML en WebStorm [aquí](#).
4. **CSS:** CSS es un lenguaje de diseño destinado al entorno web, que posibilita la mejora de la interfaz de usuario previamente diseñada al añadir elementos como colores y tamaños, entre otros. Además, es posible crear un estilo en CSS y compartirlo en el sitio web creado en HTML. Este lenguaje será empleado en la implementación del diseño de nuestra plataforma web. Puedes obtener más información sobre CSS en [enlace](#).
5. **JavaScript:** Es un lenguaje de programación que es interpretado por otros programas. Funciona bajo el paradigma de programación orientada a objetos (POO), utilizando prototipos en lugar de clases para la implementación. Este lenguaje permite crear dinámicas para los usuarios a través de la lógica de programación y será utilizado en la creación de las interacciones dinámicas en la plataforma web. Puedes encontrar más detalles sobre JavaScript en [enlace](#).

Estas herramientas y tecnologías desempeñarán un papel fundamental en la creación exitosa de nuestro producto de software.

Software Testing:

Se trata de la acción de evaluar los elementos y el funcionamiento del software sometido a prueba mediante procesos de validación y verificación.

Lenguaje Gherkin: Este lenguaje, conocido como DSL (Lenguaje Específico de Dominio), está diseñado específicamente para abordar problemas particulares. Además de poder ser interpretado en código, permite agregar historias de usuario del programa junto con sus componentes correspondientes, como Característica, Escenario, Ejemplo, Esquema de Escenario, Dado, Cuando, Entonces y Y.

Se refiere a textos escritos o ilustraciones que acompañan al software de computadora o están integrados en su código fuente. Esta documentación tiene como objetivo explicar cómo funciona el software o cómo utilizarlo.

A continuación, se describe la gestión del código fuente, también conocida por las siglas SCM (Source Code Management). Su función principal es rastrear los cambios que realizará el equipo durante el desarrollo de su proyecto en el repositorio de código fuente. Se utilizará como un sistema de control de versiones que le permitirá realizar un seguimiento de los cambios realizados por miembros o desarrolladores individuales del proyecto. Además, es importante tener en cuenta que usaremos GitHub como nuestro sistema de control de versiones.

- ☐ URL de la organización: SV51-MetaSoft-App-Web - <https://github.com/SV51-MetaSoft-App-Web>
- ☐ URL del repositorio de la Landing Page: ElixirControl-Landing-Page - <https://github.com/SV51-MetaSoft-App-Web/ElixirControl-Landing-Page>
- ☐ URL del repositorio del Front-End: ElixirControl-FrontEnd - <https://github.com/SV51-MetaSoft-App-Web/ElixirControl-FrontEnd>
- ☐ URL del repositorio del Back-End: ElixirControl-BackEnd - <https://github.com/SV51-MetaSoft-App-Web/ElixirControl-BackEnd>

GitFlow es un modelo alternativo para la creación de ramas en Git que se ha convertido en una herramienta esencial para muchos desarrolladores en los últimos años. Este flujo de trabajo de control de versiones, desarrollado y popularizado por Vicent Driessen, desempeña un papel crucial en la gestión de las versiones de un código, facilitando la creación ordenada de nuevas características (Features) y correcciones de problemas urgentes (Hotfixes).



- **Main Branches:**

- **Main:** Esta es la rama principal desde la cual se ramifican todas las demás. Contendrá la versión más reciente junto con las versiones anteriores creadas por los desarrolladores. Aquí se mantendrá el historial oficial de las versiones publicadas.
- **Develop:** Esta rama puede ser creada a partir de la rama principal (Main) y contendrá todas las características (Features) estables. A través de esta rama, el equipo podrá integrar las funcionalidades de manera efectiva.

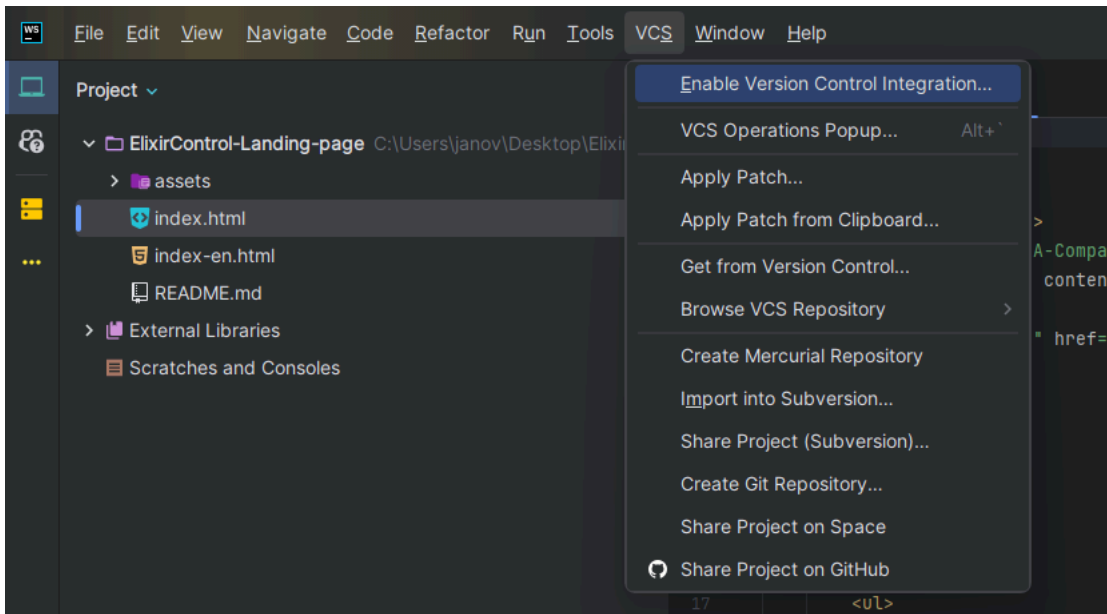
A diferencia de las ramas principales, estas ramas secundarias tienen una vida útil limitada, ya que se eliminan al fusionarse con sus ramas primarias.

- **Feature:**
 - Se ramifica de: develop
 - Debe fusionarse de nuevo en: develop
 - Se utilizan para desarrollar las nuevas funciones que se integrarán en la próxima versión. Es importante destacar que esta rama existe únicamente mientras está en proceso de desarrollo. Sin embargo, una vez que el desarrollador haya completado esa función, se fusionará nuevamente con la rama "develop".
- **Convenciones para nombrar los Features:**
 - **Feature Branch:** feature/name
Example:
 - a. feature/welcome
 - b. feature/about
 - c. feature/myfeature
 - **Conventional Commits**
El commit debe seguir la siguiente estructura:
<type> [optional scope]: <description>
[optional body]
[optional footer(s)]
 - **Type:**
 1. **feat:** Cuando se agrega un nuevo feature.
 2. **fix:** cuando corriges un error.
 3. **build:** cuando afectan los componentes de compilación como la herramienta de compilación, las dependencias o la version del proyecto.
 4. **chore:** modificaciones privadas del código.
 5. **docs:** commits que afectan solo a la documentación.
 6. **refactor:** commits que reescriben o reestructura el código, pero no cambia el comportamiento.
 7. **perf:** commits especiales que mejoran el rendimiento.
 8. **style:** commits que no afectan el programa. (espacios en blanco, formato, puntos o comas faltantes).
 9. **test:** commits que agregan pruebas.
 - **Scope**
Ofrece información contextual adicional. Aunque es opcional, es beneficioso incluirlo para proporcionar a los desarrolladores una descripción más detallada del commit.
<description>
Es una parte obligatoria del formato de los commits. Siempre debemos usar lenguaje en modo imperativo y evitar escribir en mayúsculas
[optional body]
El cuerpo es opcional y, cuando se utiliza, debe explicar la motivación detrás del cambio y contrastarlo con el comportamiento anterior. Es ideal para mencionar identificadores de problemas y sus relaciones.
[optional footer(s)]
Esta sección es opcional y puede incluir información sobre cambios significativos. Puede hacer referencia al problema por su identificación y, en esta sección, se incluyen los cambios importantes precedidos por "BREAKING CHANGES:" seguido de uno o dos saltos de línea.
Ejemplos:
 - a. feat(welcome): add welcome section
 - b. build(release): bump version to 1.0.0
 - c. style: remove empty line
 - d. feat(sign up): add the button to sign up
 - e. feat!: email the customer when product is shipped
 - f. feat: remove ticket list endpoint
refers to JIRA-1337
BREAKING CHANGES: ticket endpoints no longer supports list all entites.

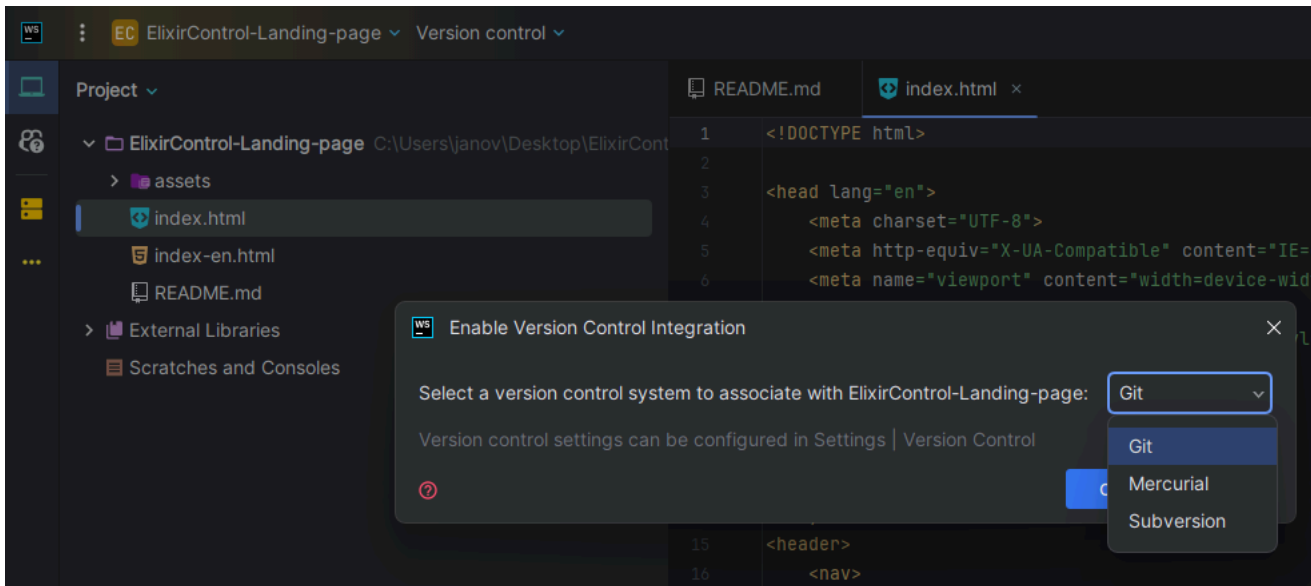
Como se mencionó previamente, la gestión de nuestro código fuente se llevará a cabo mediante GitHub. El IDE utilizado en este caso, WebStorm, debe estar vinculado directamente al repositorio creado por nuestra empresa MetaSoft. De esta manera, cada commit realizado por un miembro del equipo se subirá automáticamente y se cargará en el GitHub de la organización. Las instrucciones para completar con éxito este proceso de emparejamiento se detallan a continuación:

- **Activar el controlador de versiones del IDE**
Dado que utilizaremos GitHub para gestionar nuestro código, la opción que debe estar habilitada o seleccionada es aquella que indique que el sistema de control se realizará mediante Git. Para hacer esto, siga los siguientes pasos:

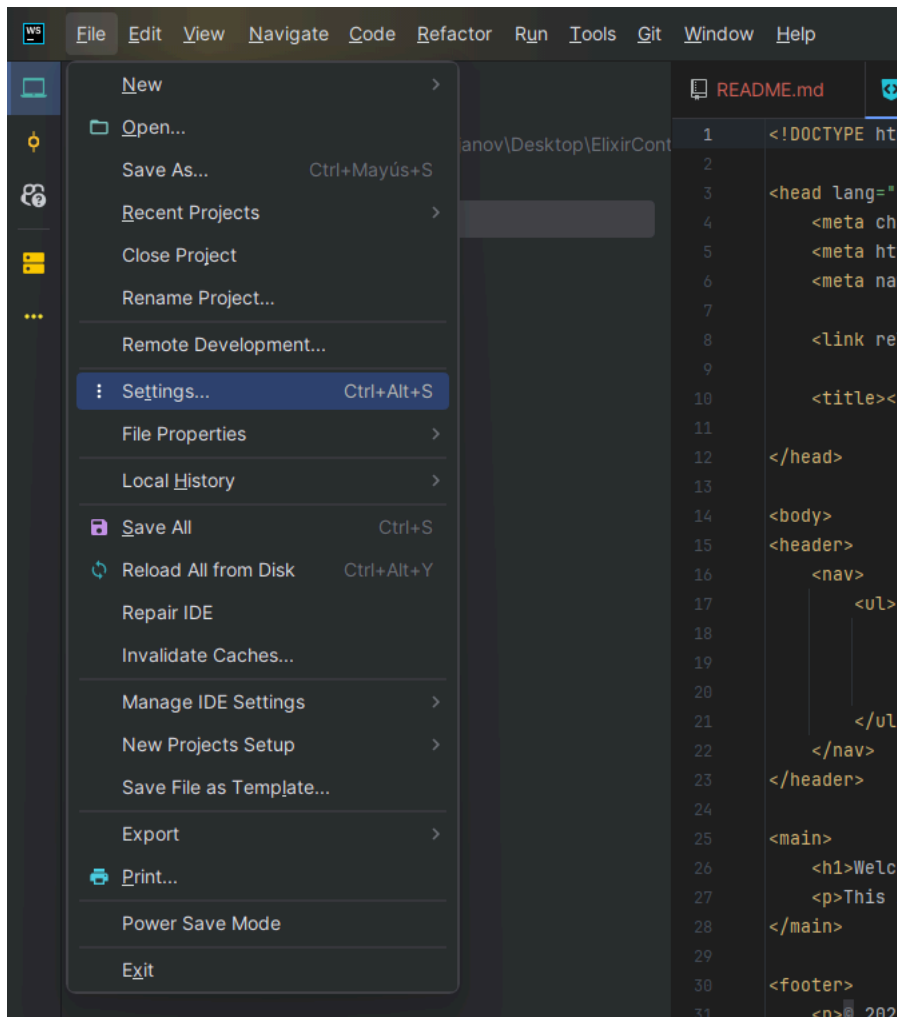
- i. Diríjase a la pestaña "VCS" en WebStorm.
- ii. Luego, seleccione la opción "Enable Version Control Integration".

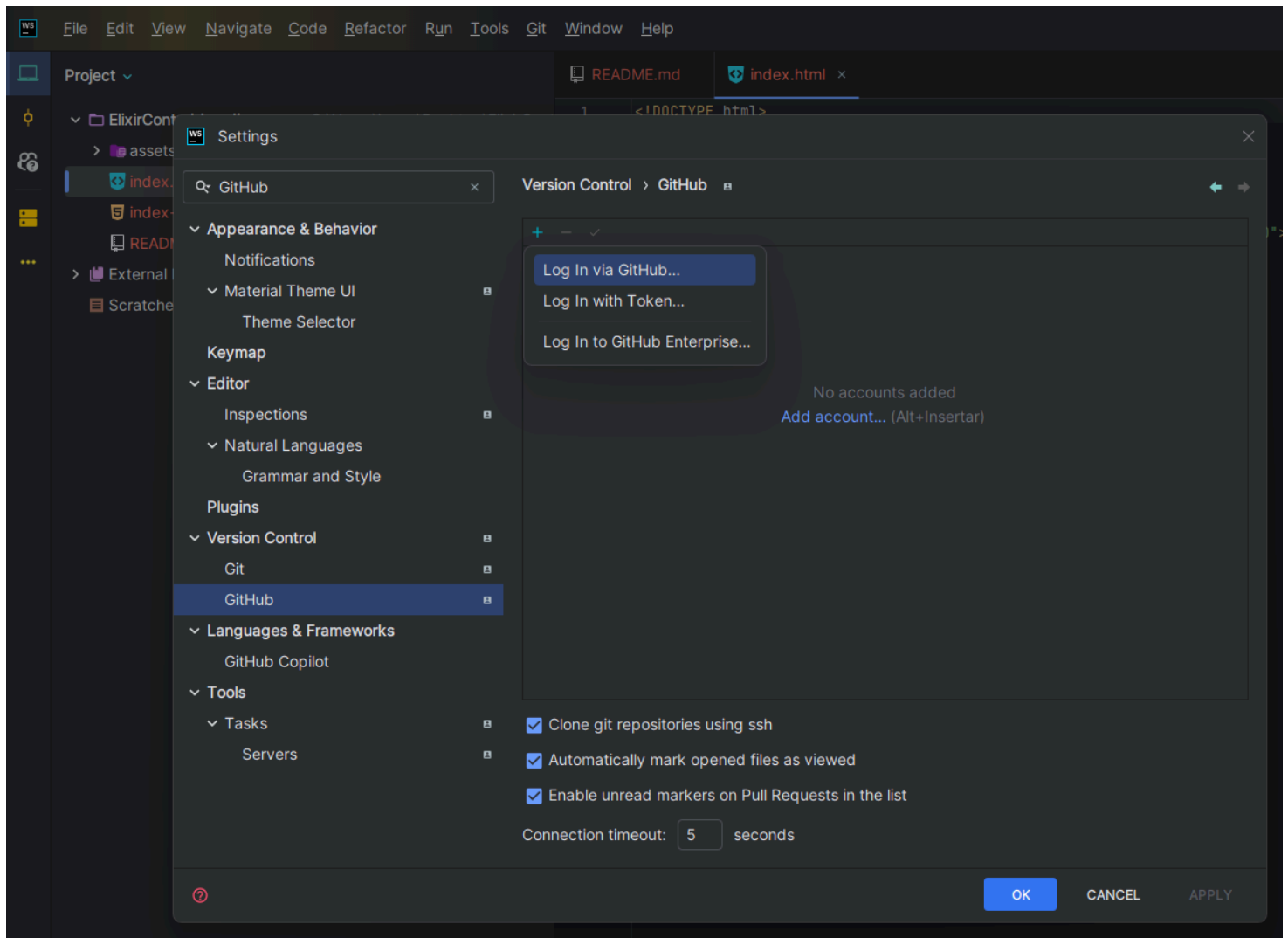


Ahora se debe seleccionar el sistema de control a través de Git y, por último aceptar los cambios.

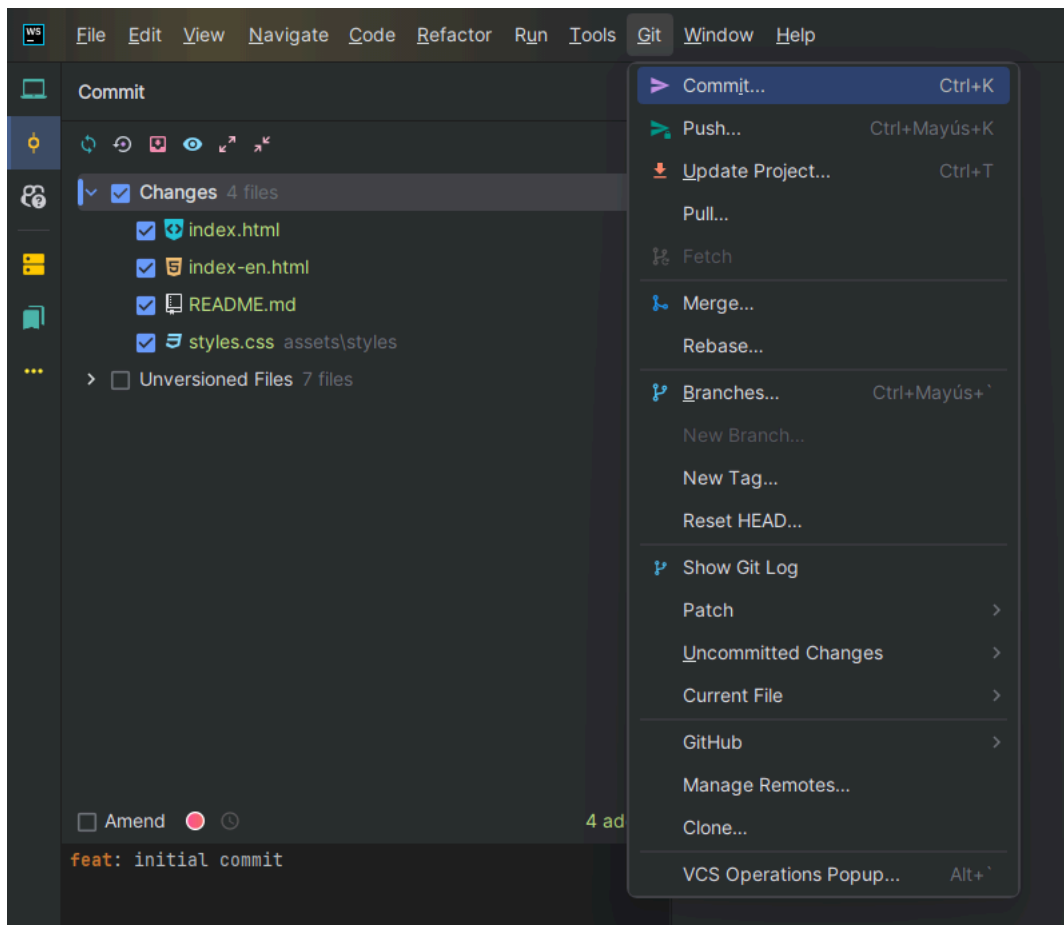


- **Agregar una cuenta de GitHub, siga estos pasos:**
 - i. Diríjase a la sección de configuración en su aplicación.
 - ii. Dentro de la pestaña 'File', busque y seleccione la opción 'Settings'.
 - iii. En la configuración, busque la sección de version control.
 - iv. Agregue su cuenta de GitHub para obtener acceso a los repositorios.

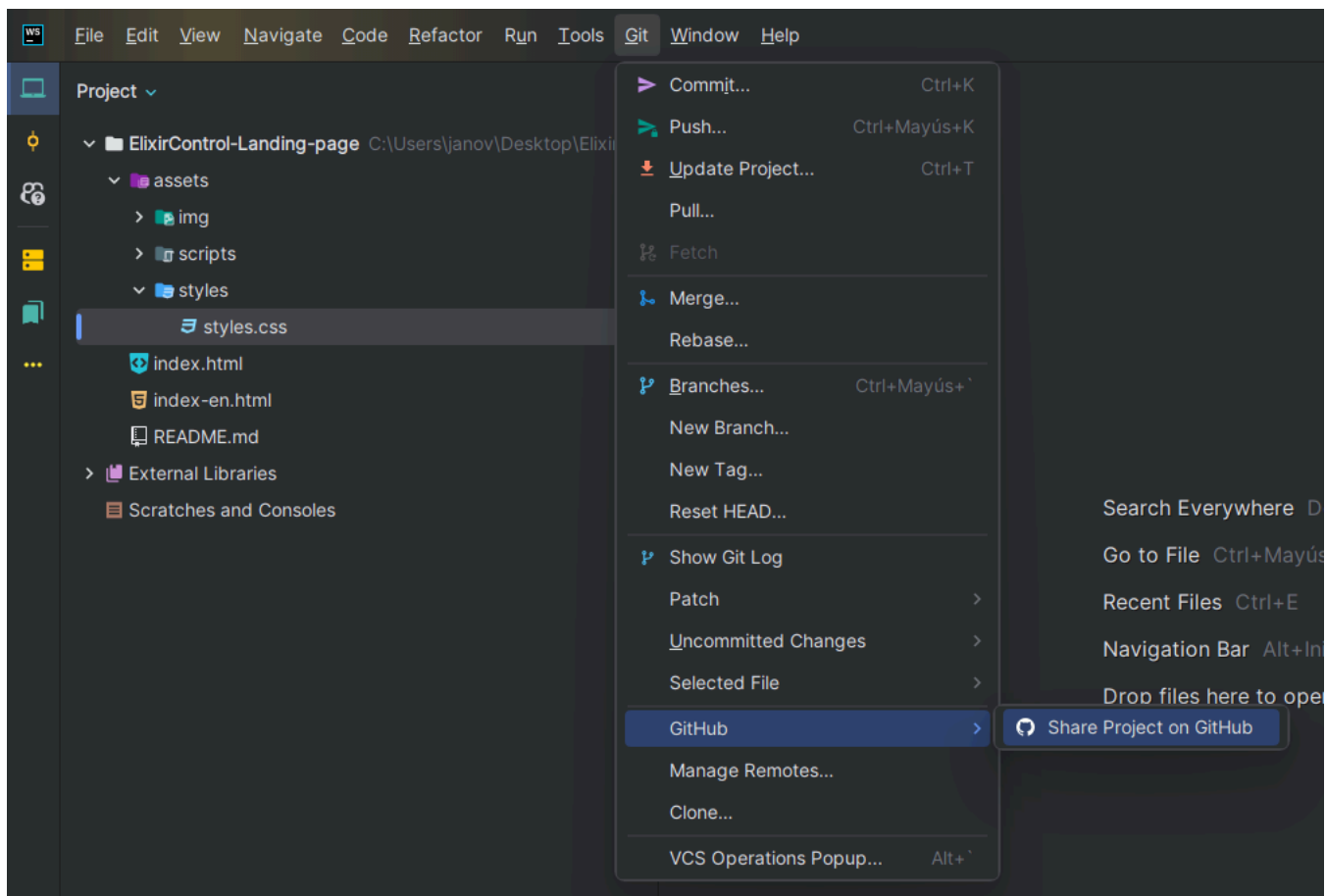


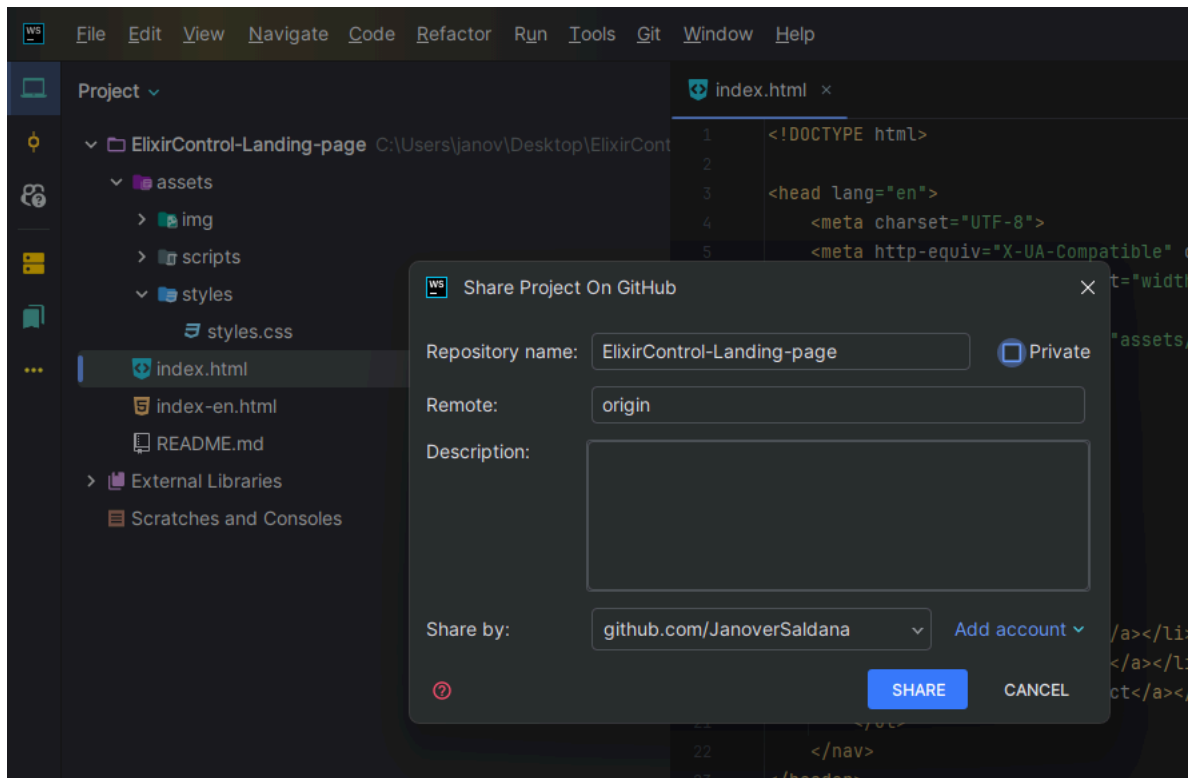


- **Configurar el nombre de usuario de Git:** Una vez que hayas establecido el sistema de control de versiones que se vinculará con tu IDE, deberás ingresar la cuenta que utilizarás. Para hacerlo, sigue estos pasos:
 - i. Realiza un commit en tu proyecto. Durante este proceso, se te solicitará que ingreses tu nombre de usuario de Git.
 - ii. Después de haberlo añadido, todos los cambios se guardarán en el repositorio especificado en esa plataforma, siempre y cuando des la orden correspondiente.
 - iii. Para configurar tu nombre de usuario de Git, primero selecciona la opción 'commit' que se encuentra dentro de la pestaña 'Git'.

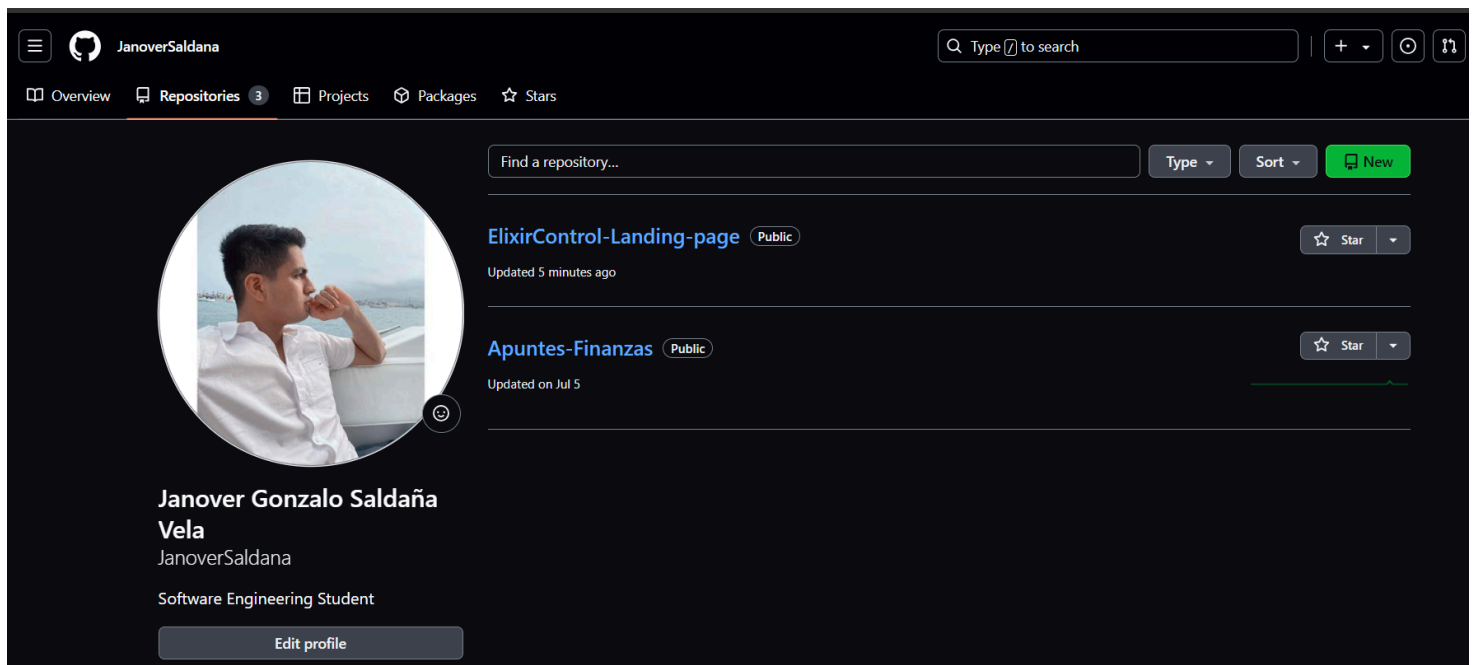


- **Guardar el progreso en GitHub:** Con todo configurado en WebStorm, ahora puedes subir tu código a GitHub sin problemas. Simplemente dirígete a la opción 'GitHub' que se encuentra en la pestaña 'Git' y comparte el proyecto.





- **Configurar la propiedad del repositorio en GitHub:** Ahora, solo necesitas configurar la ubicación del repositorio. El código ya debería estar guardado en GitHub, pero solo estará presente en tu propia cuenta. Para cambiar la propiedad y transferirla a la organización deseada, sigue estos pasos:
 - i. Ingresa al repositorio creado en GitHub.
 - ii. Selecciona la pestaña 'settings'
 - iii. Dirigite al apartado de 'DangerZone'
 - iv. Luego da click en 'transfer'
 - v. Finalmente elegimos el nuevo lugar para guardar el repositorio.



JanoverSaldana / ElixirControl-Landing-page

Q Type [f] to search

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

ElixirControl-Landing-page

Public

Pin

Unwatch 1

master

1 Branch

0 Tags

Go to file

Add file

<> Code

JanoverSaldana

feat: initial commit

d7e475b · 11 minutes ago

1 Commit

assets/styles	feat: initial commit	11 minutes ago
README.md	feat: initial commit	11 minutes ago
index-en.html	feat: initial commit	11 minutes ago
index.html	feat: initial commit	11 minutes ago

Danger Zone

Change repository visibility

This repository is currently public.

Change visibility

Disable branch protection rules

Disable branch protection rules enforcement and APIs

Disable branch protection rules

Transfer ownership

Transfer this repository to another user or to an organization where you have the ability to create repositories.

Transfer

Archive this repository

Mark this repository as archived and read-only.

Archive this repository

Delete this repository


Once you delete a repository, there is no going back. Please be certain.

Delete this repository

Transfer repository: JanoverSaldana/ElixirControl-Landing-page

Transfer this repository to another user or to an organization where you have the ability to create repositories.


Required fields are marked with an asterisk (*).

 To understand admin access, teams, issue assignments, and redirects after a repository is transferred, see [Transferring a repository](#) in GitHub Help.

Transferring may be delayed until the new owner approves the transfer.

New owner *


☒ Select one of my organizations

 SV51-MetaSoft-App-Web

☐ Specify an organization or username

Repository name *

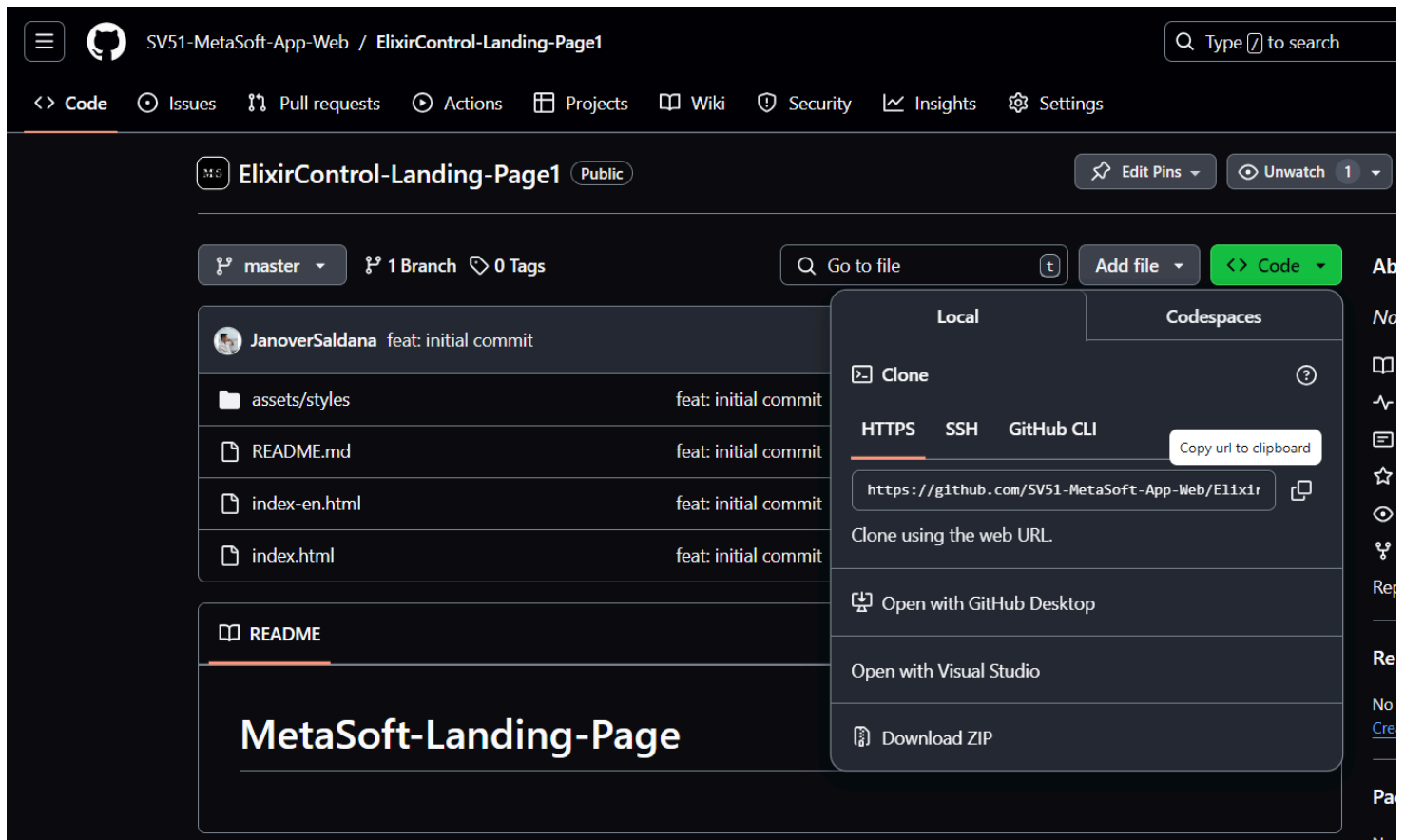
ElixirControl-Landing-Pag

 ElixirControl-Landing-Page1 is available.

Type JanoverSaldana/ElixirControl-Landing-page to confirm. *

[I understand, transfer this repository.](#)

- **Configurar control remoto en Git:** Por último, dado que el repositorio ahora está bajo la propiedad de la empresa y depende de ella, es necesario acceder al control remoto del código. Para hacerlo, simplemente ingresa al repositorio creado y copia la URL del repositorio.



SV51-MetaSoft-App-Web / ElixirControl-Landing-Page1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

ElixirControl-Landing-Page1 Public

master 1 Branch 0 Tags

Go to file Add file Code

Local Codespaces

Clone

HTTPS SSH GitHub CLI

Copy url to clipboard

https://github.com/SV51-MetaSoft-App-Web/ElixirControl-Landing-Page1

Clone using the web URL

Open with GitHub Desktop

Open with Visual Studio

Download ZIP

JanoverSaldana feat: initial commit

assets/styles feat: initial commit

README.md feat: initial commit

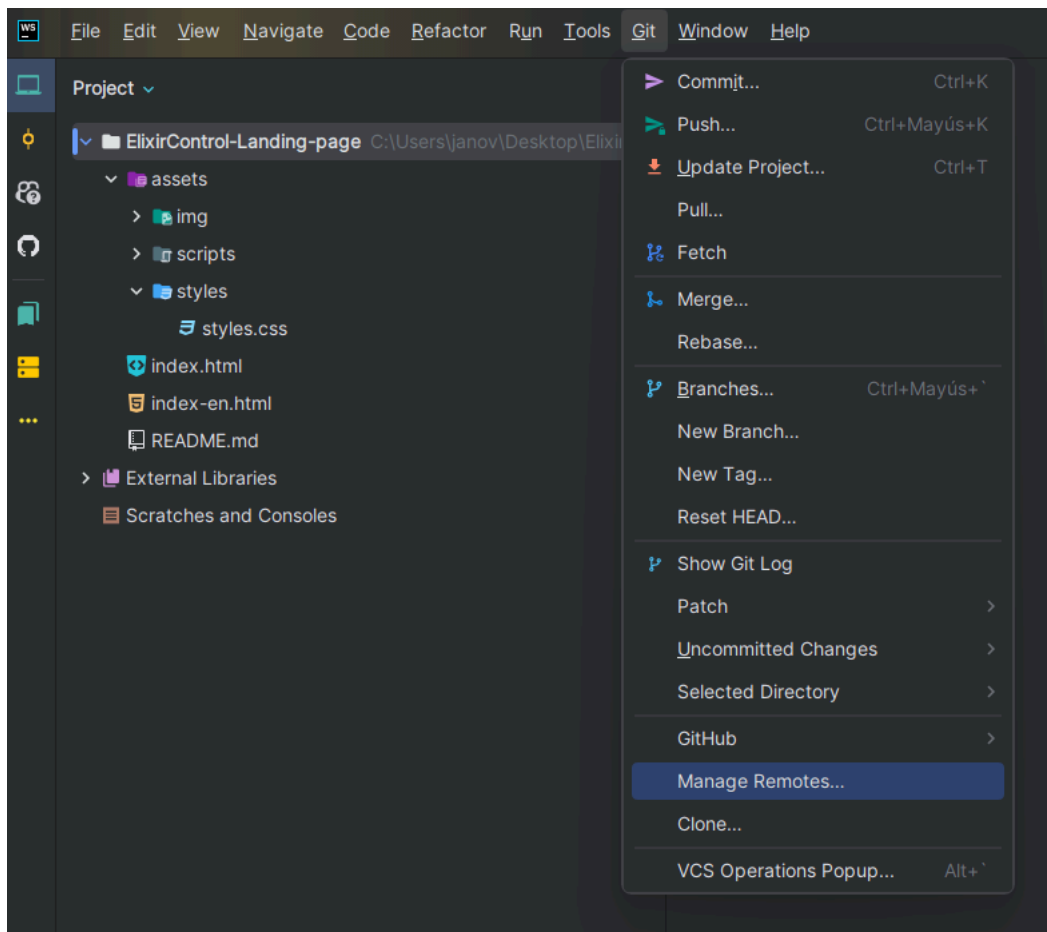
index-en.html feat: initial commit

index.html feat: initial commit

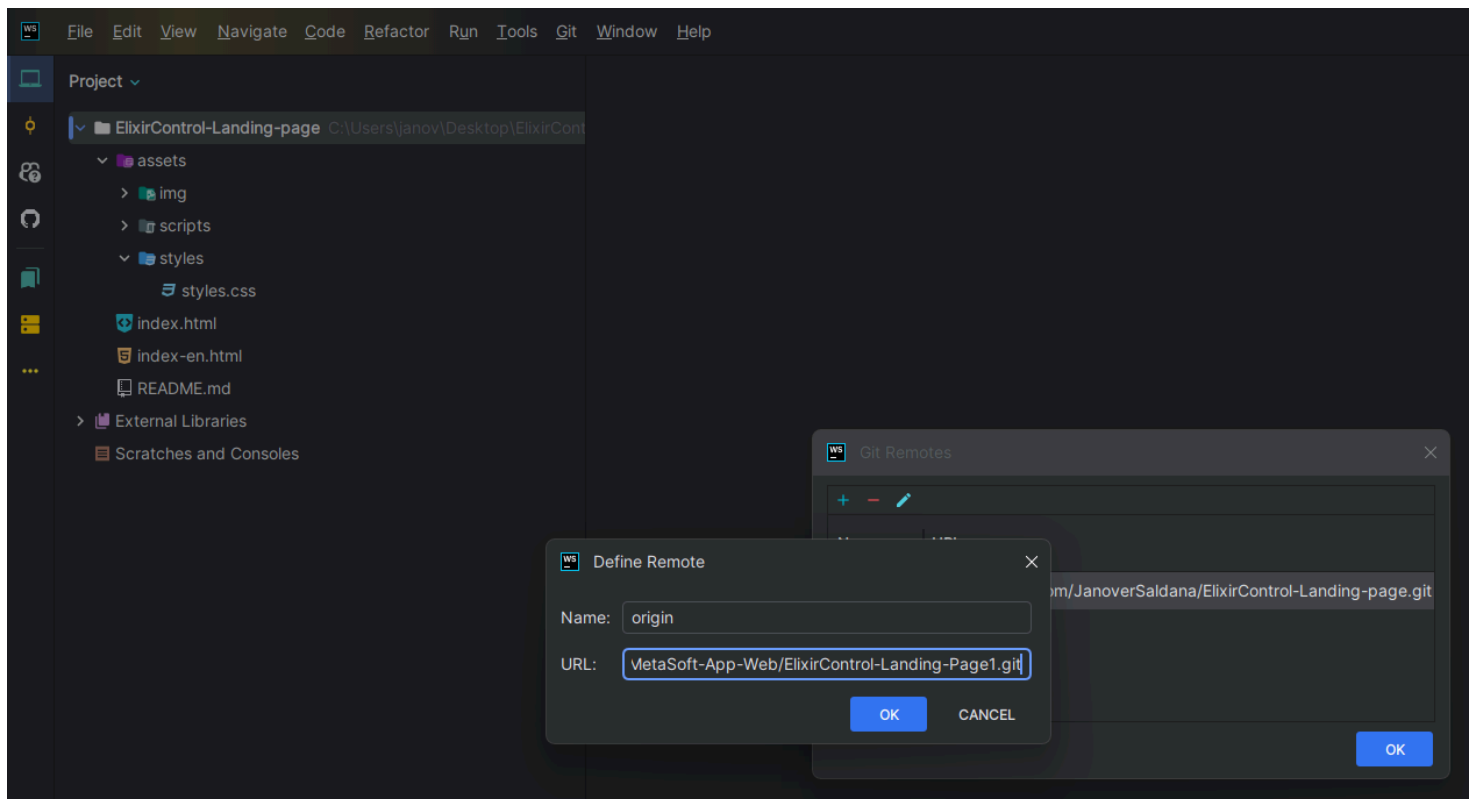
README

MetaSoft-Landing-Page

Ahora, en el IDE, dirígete a la pestaña 'Git' y elige la opción 'Manage Remotes'.



Finalmente, como último paso, debes pegar el enlace copiado en el campo de dirección que solicita el IDE para el control remoto en Git.



Si has seguido correctamente todos los pasos y directrices mencionados, entonces has completado la configuración con éxito. Ahora, solo necesitas realizar un commit y los cambios que hayas efectuado se guardarán en el repositorio de GitHub, ya sea que hayas realizado modificaciones en el código, creado nuevas ramas u otras acciones.

5.1.3. Source Code Style Guide & Conventions.

En esta sección, se presentarán las pautas, convenciones, estilos y principios que se aplicarán a cada uno de los lenguajes utilizados en la creación de nuestra aplicación. La observancia de este conjunto de directrices reviste una importancia fundamental, ya que tiene el propósito de mantener la calidad estructural del software, mejorar la legibilidad del código fuente y simplificar el mantenimiento del mismo.

Dado que en este proyecto se emplearán varios lenguajes, como HTML, CSS, JavaScript, C# y TypeScript para el desarrollo de la plataforma web, así como Gherkin para el proceso de pruebas del programa, a continuación, se detallarán y describirán las reglas y recomendaciones generales que se tendrán en cuenta al utilizarlos.

Nomenclatura General

Para los nombres de variables, objetos, elementos y funciones que se utilicen en el proyecto, se emplearán términos en inglés que estén relacionados con lo que representan. No se utilizarán mayúsculas en estos nombres, ya que, de acuerdo con W3Schools (sin fecha), la combinación de mayúsculas y minúsculas puede dificultar la legibilidad del código. En su lugar, se optará por utilizar exclusivamente letras minúsculas, lo que contribuirá a una mayor claridad en el código.

Ejemplos de nomenclatura estándar, siguiendo las recomendaciones de Google (s.f.):

```
.gallery {}  
.video {}  
.login {}
```

Estas pautas de nomenclatura ayudarán a mantener una coherencia en el código y facilitarán su comprensión.

Sangría

Cuando se trabaje con HTML, CSS y/o JavaScript, se aplicará un espaciado de dos espacios antes de cada línea que se encuentre dentro de un bloque. Según W3Schools (sin fecha), no se recomienda el uso de la tecla "Tabulación". A continuación, se muestra un ejemplo de la sangría estándar en HTML siguiendo las directrices de W3Schools (s.f.):

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Título del Documento</title>  
  </head>  
  <body>  
    <h1>Encabezado Principal</h1>  
    <p>Este es un párrafo dentro del cuerpo del documento.</p>  
  </body>  
</html>
```

Este estilo de sangría proporciona una estructura clara y organizada al código, lo que facilita su lectura y mantenimiento.

Ejemplo de formato estándar de sangría en CSS, conforme a las recomendaciones de W3Schools (s.f.):

```
html {  
  background: #fff; /* Fondo blanco */  
  color: #404; /* Color de texto gris */  
}
```

Ejemplo de nomenclatura estándar de la sangría en JavaScript según W3School (s.f.):

```
function toCelsius(fahrenheit) {  
  return (5 / 9) * (fahrenheit - 32);  
}
```

Especificaciones generales

A continuación, detallaremos las reglas específicas necesarias para comprender el código de nuestra aplicación en cada lenguaje.

HTML:

HTML, acrónimo de HyperText Markup Language en inglés, es un lenguaje de marcado que se utiliza para definir la estructura de una página web. También incluye funcionalidades que permiten controlar el comportamiento de diferentes elementos del contenido de la página, como cambiar el tamaño del texto o aplicar formato cursiva, entre otros. En nuestro proyecto, emplearemos HTML5, y a continuación, se presentan las características y directrices que debemos seguir para utilizar este lenguaje de la siguiente manera:

- **Declare Document Type**

La declaración del tipo de documento debe realizarse en la primera línea del código. Según las recomendaciones de Google (s.f.), se prefiere la sintaxis de HTML5 para todos los documentos HTML. Para declararla, simplemente copia lo siguiente:

```
<!DOCTYPE html>
```

- **Blank Lines**

Cada vez que comiences un nuevo bloque, lista o tabla de gran longitud, es recomendable dejar una línea en blanco después del elemento anterior para mejorar la legibilidad y la presentación del código, de acuerdo con las pautas de W3Schools (s.f.):

```
<!DOCTYPE html>
<html>
<head>
<title>Animales Exóticos</title>
</head>
<body>
<h1>Lemur de Madagascar</h1>
<p>El lémur de Madagascar es un primate endémico de la isla de Madagascar en el Océano Índico.</p>

<h1>Pangolín</h1>
<p>El pangolín es un mamífero cubierto de escamas que se encuentra en regiones de África y Asia.</p>

<h1>Ocelote</h1>
<p>El ocelote es un felino salvaje que habita en América del Sur y Central, conocido por su pelaje moteado.</p>
</body>
</html>
```

Esta práctica de dejar una línea en blanco mejora la estructura y legibilidad del código HTML.

- **Quote attribute Values**

Para los valores de los atributos, es común utilizar comillas dobles alrededor de ellos, aunque esta característica no sea obligatoria. Según W3Schools (sin fecha), esto mejora la legibilidad del código y es una práctica común entre los desarrolladores. Aquí tienes un ejemplo:

```
<table class="striped">
```

Este enfoque de usar comillas dobles alrededor de los valores de los atributos es ampliamente aceptado y recomendado en la comunidad de desarrollo web.

- **Never Skip the <title> Element**

El elemento `<title>` permite que las páginas aparezcan en la lista de resultados al realizar búsquedas en un navegador web. Además, este elemento es responsable de proporcionar el nombre de la página cuando se agrega a marcadores o favoritos. A continuación, se muestra un ejemplo de su uso:

```
<title>Guía de Estilo HTML y Convenciones de Codificación</title>
```

Este elemento es esencial para mejorar la identificación y accesibilidad de una página web.

- **HTML Line-Wrapping**

A pesar de que en un documento HTML no exista un límite estricto en la cantidad de palabras por línea, no se recomienda generar líneas de código excesivamente largas. De hecho, hacerlo dificulta la legibilidad del código. Para continuar en la siguiente línea, se deben utilizar al menos cuatro espacios para distinguir elementos secundarios. Aquí tienes un ejemplo basado en las recomendaciones de Google (sin fecha):

```
<button mat-icon-button color='primary' class="menu-button"
(click)="openMenu()">
<mat-icon>menu</mat-icon>
</button>
```

Este estilo de formateo ayuda a mantener un código más legible y facilita la identificación de los elementos y su jerarquía en la estructura del documento HTML.

CSS:

CSS, conocido por sus siglas en inglés, Cascading Style Sheets (Hojas de Estilo en Cascada), es un lenguaje que se enfoca en definir y mejorar la presentación de un documento basado en HTML. A continuación, se presentan las directrices que debemos seguir al utilizar CSS:

- **Shorthand Properties**

Se recomienda utilizar abreviaturas de propiedades y declarar los campos de los elementos en la menor cantidad de líneas posible, según las pautas de Google (sin fecha). Esto aumenta la eficiencia del código y lo hace más legible. Además, se debe evitar agregar unidades después del valor cero.

Aquí tienes un ejemplo:

```
border-top: 0;
font: 100%/1.6 palatino, georgia, serif;
padding: 0 1em 0;
```

Siguiendo estas recomendaciones, se puede lograr un código CSS más conciso y fácil de entender.

- **Declaration Stops**

Es importante incluir un punto y coma al final de cada declaración en CSS, al igual que en la mayoría de los lenguajes de programación. Siguiendo las pautas de Google (sin fecha), esta práctica contribuye a mantener la coherencia en el código. A continuación, se muestra un ejemplo:

```
html {
  background: #fff;
  color: #404;
}
```

El uso consistente de puntos y comas al final de las declaraciones CSS ayuda a prevenir errores y mejora la claridad del código.

- **Property Name Stops**

Es necesario incluir un espacio entre los dos puntos que siguen al nombre de una propiedad y el valor correspondiente. Siempre se debe colocar un solo espacio después de los dos puntos, pero no antes. A continuación, se muestra un ejemplo siguiendo esta convención estándar de Google (s.f):

```
html {
  background: #fff;
  color: #404;
}
```

Mantener esta consistencia en la colocación de espacios ayuda a que el código CSS sea más legible y fácil de entender.

- **Declaration Block Separation**

Es esencial utilizar un espacio separador después del nombre de un selector de elemento y antes de la llave que inicia un bloque de declaración CSS. Además, la llave de apertura del bloque debe estar en la misma línea que el selector. Aquí tienes un ejemplo siguiendo esta convención estándar de Google (sin fecha):

```
html {
  background: #fff;
  color: #404;
}
```

El cumplimiento de estas directrices ayuda a mantener la consistencia y la legibilidad en el código CSS.

- **CSS quotation Marks**

No se deben utilizar comillas dobles (") en el código CSS; en su lugar, se permiten y deben emplearse comillas simples (') únicamente para

selectores de atributos y valores de propiedades.

Ejemplo conforme a las pautas estándar de Google (sin fecha):

```
html {  
  font-family: 'open sans', arial, sans-serif;  
}
```

Este ejemplo demuestra el uso de comillas simples para encerrar el valor del atributo `font-family` en CSS, lo cual es una práctica común y aceptada.

JavaScript

JavaScript es un lenguaje de programación que permite especificar de manera precisa las acciones que debe realizar el navegador web, incluyendo el orden de ejecución de tareas y la frecuencia con la que se deben llevar a cabo. A continuación, se presentan las pautas para el uso de JavaScript en nuestro proyecto:

- **Spaces around operators**

Es importante añadir espacios alrededor de cada operador matemático y comas que se utilicen en el código JavaScript. A continuación, se muestra un ejemplo siguiendo la convención estándar de W3Schools (sin fecha):

```
let x = y + z;  
const myArray = ['Volvo', 'Saab', 'Fiat'];
```

El uso consistente de espacios alrededor de operadores y comas mejora la legibilidad del código JavaScript.

- **Simple Statement's End**

Es fundamental que una instrucción simple finalice con un punto y coma, tal como es el caso en muchos otros lenguajes de programación. A continuación, se muestra un ejemplo que cumple con la convención estándar de W3Schools (sin fecha):

```
let x = v + 7;  
const myArray = ['Volvo', 'Saab', 'Fiat'];
```

El uso de punto y coma al final de cada instrucción ayuda a garantizar la estructura correcta del código JavaScript y a evitar posibles errores.

- **Beginning and End of Function**

Un bloque de función debe incluir una llave al final de la primera línea, de modo que el cierre de la función esté en la última línea, sin necesidad de un punto y coma. Este mismo principio se aplica a las estructuras condicionales y los bucles. A continuación, se muestra un ejemplo que cumple con la convención estándar de W3Schools (sin fecha):

```
function toCelsius(fahrenheit) {  
  return (5 / 9) * (fahrenheit - 32);  
}
```

En este ejemplo, la función `toCelsius` está formateada de acuerdo con estas pautas, con la llave de apertura en la misma línea que la declaración de la función y la llave de cierre en la última línea. Esto ayuda a mantener la estructura y la legibilidad del código JavaScript.

- **Object Rules**

Para la creación de un objeto, al igual que en una función, se comienza con una llave al final de la primera línea. Sin embargo, en este caso, la llave de cierre debe ir seguida de un punto y coma. Para definir las propiedades del objeto, se utilizan dos puntos y un espacio para separar el nombre de la propiedad de su valor. Si el valor es un string, se debe encerrar entre comillas dobles. A continuación, se muestra un ejemplo siguiendo la convención estándar de W3Schools (sin fecha):

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};
```

En este ejemplo, el objeto `person` está formateado de acuerdo con estas pautas, lo que mejora la legibilidad y la estructura del código JavaScript.

Gherking:

Gherkin es un Lenguaje Específico de Dominio (DSL por sus siglas en inglés) que se utiliza para resolver problemas específicos mediante la generación de casos de prueba que validan una característica en diversos escenarios. Gherkin incluye varios elementos, entre los cuales los más conocidos y utilizados son Feature, Scenario, Example, Given, When y Then. A continuación, se presentan las pautas que debemos seguir al utilizar Gherkin en nuestro código:

- **Discernible Given-When-Then Blocks**

Es importante aplicar sangría a los elementos que representan los pasos a seguir en un escenario. En el caso de "And", se debe aplicar una sangría adicional. Siguiendo la recomendación de Keiblinger (2021), este enfoque ayuda a identificar rápidamente las partes que componen un escenario. A continuación, se muestra un ejemplo:

Scenario: Ingreso de requisitos con claridad

```
Given que en el formulario de ingreso de oferta laboral
When escribo claramente los requisitos
Then se mostrará el mensaje
And mi oferta solo aparecerá a quienes cumplan con estos
And se habilita la opción
```

En este ejemplo, se ha aplicado la sangría de manera adecuada para resaltar los pasos del escenario, y se ha utilizado una sangría adicional para los pasos que comienzan con "And". Esto mejora la legibilidad y la comprensión de los escenarios escritos en Gherkin.

- **Step with Tables**

Conforme a la recomendación de Keiblinger (2021), cuando sea necesario introducir valores en partes del escenario, se debe emplear una tabla o crear un formulario que refleje esa parte del escenario. Antes de esta representación, se deben colocar dos puntos. Aquí tienes un ejemplo:

```
Then se mostrará el mensaje:
| Mensaje |
| Se completaron los requisitos adecuadamente |
```

Este enfoque permite una representación clara y estructurada de los valores relacionados con una parte específica del escenario.

- **Reducing Noise**

Para evitar la acumulación de demasiadas líneas de código en un escenario, es recomendable incluir valores por defecto dentro de los pasos para campos que no sean muy relevantes para ese escenario en particular. Los valores "estándar" que se coloquen deben estar entre comillas simples. Siguiendo el consejo de Keiblinger (2021), esta práctica contribuye significativamente a la reducción del tamaño del código. A continuación, se muestra un ejemplo:

```
When escribo claramente los requisitos 'dominio en C'
```

En este ejemplo, se ha incluido un valor por defecto ('dominio en C') entre comillas simples dentro del paso para representar un campo que no es esencial en ese escenario. Esto ayuda a mantener el escenario más conciso y legible.

- **Scenarios Separator**

Para separar dos escenarios, se debe insertar un salto de línea y, según la sugerencia de Keiblinger (2021), si es posible, agregar una línea de comentario para facilitar la visualización de estos. De esta manera, se identifica rápidamente el inicio y el fin de un escenario. A continuación, se presenta un ejemplo:

Scenario: Ingreso de requisitos con claridad

Given que en el formulario de ingreso de oferta laboral
When escribo claramente los requisitos
Then se mostrará el mensaje
And mi oferta solo aparecerá a quienes cumplan con estos
And se habilita la opción

Scenario: Otro escenario

Given que en otro contexto
When ocurre algo diferente
Then se muestra otro resultado

En este ejemplo, se ha agregado un salto de línea entre los dos escenarios y se ha incluido una línea de comentario como separador para mejorar la visualización y la identificación de cada escenario.

C#:

C# es un lenguaje de programación desarrollado por Microsoft en el año 2000 como parte de su plataforma .NET. Desde su creación, C# ha evolucionado significativamente, convirtiéndose en una herramienta esencial para el desarrollo de una amplia gama de aplicaciones, desde software de escritorio hasta aplicaciones web y móviles, así como servicios en la nube. Su diseño moderno y su integración con el ecosistema de .NET lo han consolidado como una opción preferida para muchos desarrolladores a nivel global.

A continuación, se presentan las pautas para utilizar C# en nuestro proyecto:

- **Datos de cadena**

Use **interpolación de cadenas** para concatenar cadenas cortas, como se muestra en el código siguiente.

```
string displayName = $"{nameList[n].LastName}, {nameList[n].FirstName}";
```

Para anexas cadenas en bucles, especialmente cuando se trabaja con grandes cantidades de texto, utilice un objeto `System.Text.StringBuilder`. En este ejemplo, se han seguido las pautas para nombrar clases e interfaces de manera clara y legible.

```
var phrase = "lalalalalalalalalalalalalalalalalalalalalalalalalalalalal";  
var manyPhrases = new StringBuilder();  
for (var i = 0; i < 10000; i++)  
{  
    manyPhrases.Append(phrase);  
}  
//Console.WriteLine("tra" + manyPhrases);
```

- **Delegados**

Use **Func<>** y **Action<>** en lugar de definir tipos de delegado. En una clase, defina el método delegado.

```
Action<string> actionExample1 = x => Console.WriteLine($"x is: {x}");

Action<string, string> actionExample2 = (x, y) => Console.WriteLine($"x is: {x}, y is {y}");

Func<string, int> funcExample1 = x => Convert.ToInt32(x);

Func<int, int, int> funcExample2 = (x, y) => x + y;
```

Llame al método con la signatura definida por el delegado Func<> o Action<>.

```

        actionExample1("string for x");
        actionExample2("string for x", "string for y");
        Console.WriteLine($"The value is {funcExample1("1")}");
        Console.WriteLine($"The sum is {funcExample2(1, 2)}");
    }
}

```

- **try-catch y using en el control de excepciones**

Use **try-catch** para controlar las excepciones. Use **using** para liberar recursos no administrados.

```
try
{
    // Code that may throw an exception
}
catch (Exception ex)
{
    // Code to handle the exception
}

using (var resource = new Resource())
{
    // Code that uses the resource
}
```

- **Operadores && y ||**

Use **&&** y **||** en lugar de **&** y **|** para operaciones lógicas. Los operadores **&&** y **||** realizan una evaluación de cortocircuito, lo que significa que si la primera parte de la expresión es suficiente para determinar el resultado, la segunda parte no se evalúa.

```
if (a == 10 && b == 20)
{
    // Code to execute if both conditions are true
}

if (a == 10 || b == 20)
{
    // Code to execute if either condition is true
}
```

- **Comentarios**

Use comentarios para explicar el código y proporcionar información adicional. Los comentarios deben ser claros y concisos.

```
// This is a single-line comment

/*
This is a multi-line comment
that spans multiple lines
*/
```

- **Operador new**

Use **new** para crear instancias de clases y estructuras.

```
var person = new Person();
var point = new Point(10, 20);
```

- **Control de eventos**

Use **eventos** para notificar a los suscriptores cuando ocurre un evento.

```
public class Button
{
    public event EventHandler Click;

    protected virtual void OnClick(EventArgs e)
    {
        Click?.Invoke(this, e);
    }
}
```

- **Consultas LINQ**

Use consultas LINQ para realizar operaciones de consulta en colecciones de datos.

```
var query = from c in customers
            where c.City == "London"
            select c;
```

- **Variables locales con asignación implícita de tipos**

Use la asignación implícita de tipos para las variables locales cuando el tipo se puede inferir fácilmente.

```
var name = "John";
var age = 30;
```

Typescript

JavaScript es uno de los lenguajes más populares y ha experimentado un rápido avance y mejora en los últimos años. A continuación, se presentan las pautas para utilizar JavaScript en nuestro proyecto:

En TypeScript, se recomienda que las variables se declaren en minúsculas y se especifique el tipo de dato utilizando dos puntos después del nombre de la variable. Aquí tienes ejemplos de cómo declarar y asignar valores a variables en TypeScript:

```
// Definición e inicialización separadas
let edad: number;
edad = 20;

// Definición e inicialización en la misma línea.
let edadAitor: number = 18;
```

Además, en TypeScript, se siguen las mismas convenciones que se utilizan en JavaScript.

5.1.4. Software Deployment Configuration.

Para desplegar la Landing Page desde GitHubPages hay que seguir los siguientes pasos:

1. Ubicar el repositorio que tiene guardado el código fuente y dirigirse al apartado de configuración (settings):

SV51-MetaSoft-App-Web / ElixirControl-Landing-Page

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

ElixirControl-Landing-Page Public

Edit Pins Watch 0 Fork 0

main 1 Branch 0 Tags

Go to file Add file Code

About

No description, website

Readme Activity Custom properties 0 stars 0 watching 0 forks Report repository

JanoverSaldana feat: Added the base structure of the project. 23d5ab4 · 2 hours ago 2 Commits

assets	feat: Added the base structure of the project.	2 hours ago
README.md	Initial commit	last week
index-en.html	feat: Added the base structure of the project.	2 hours ago
index.html	feat: Added the base structure of the project.	2 hours ago

1. Seleccionar la sección pages:

SV51-MetaSoft-App-Web / ElixirControl-Landing-Page

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

General

Access

Collaborators and teams

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Pages

Custom properties

General

Repository name

ElixirControl-Landing-Page Rename

☐ Template repository

Template repositories let users generate new repositories with the same directory structure and files. [Learn more about template repositories.](#)

☐ Require contributors to sign off on web-based commits

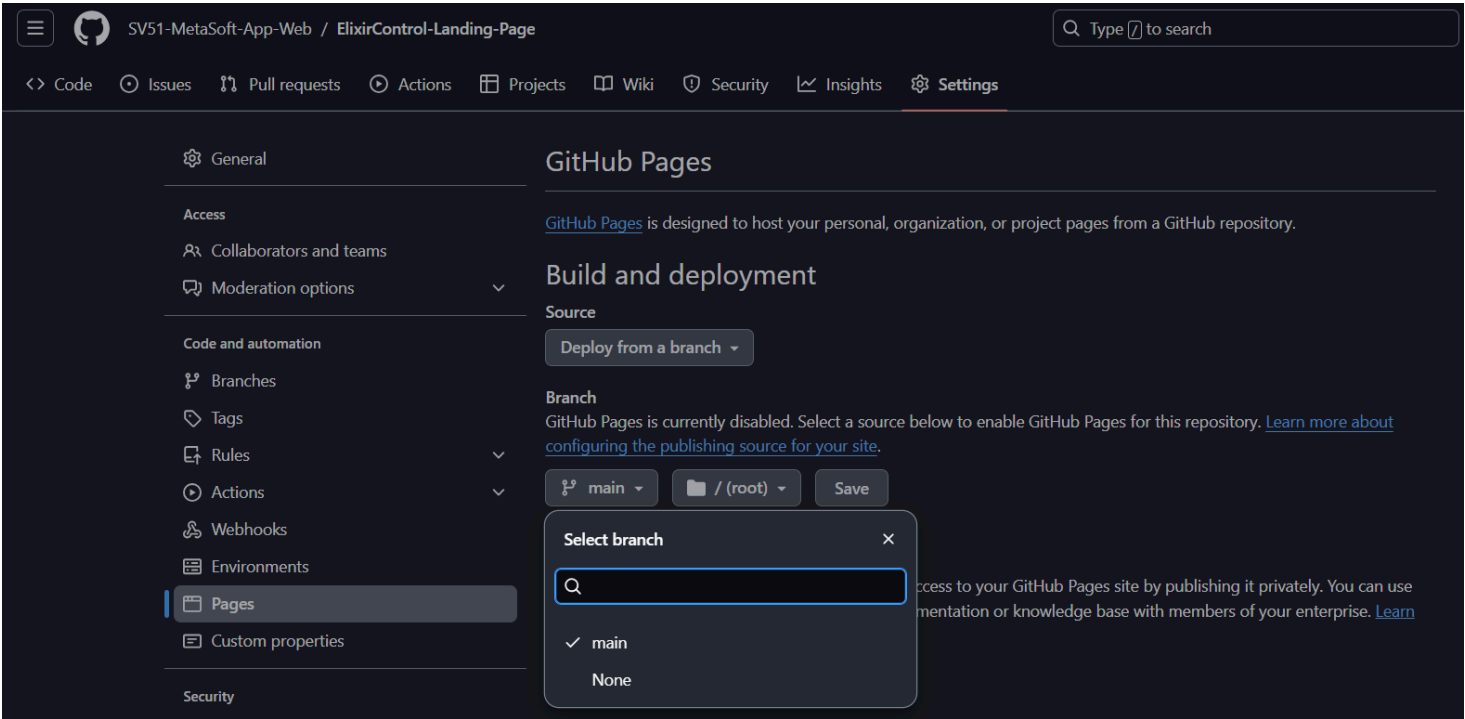
Enabling this setting will require contributors to sign off on commits made through GitHub's web interface. Signing off is a way for contributors to affirm that their commit complies with the repository's terms, commonly the [Developer Certificate of Origin \(DCO\)](#). [Learn more about signing off on commits.](#)

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

main

1. Configurar la rama que será usada para hacer deploy:



5.2. Landing Page, Services & Applications Implementation.

5.2.1. Sprint 1

En la fase inicial de nuestro proyecto, nos propusimos llevar a cabo la implementación del diseño de nuestra Landing Page utilizando WebStorm como entorno de desarrollo. Esto implica que al concluir el Sprint, todas las secciones, ya sea Home, Services, Pricing, Testimonials o About Us, deben estar completadas. A continuación, adjuntamos imágenes que ilustran cómo gestionamos las tareas en Pivotal Tracker.

Repositorio: <https://github.com/SV51-MetaSoft-App-Web/ElixirControl-Landing-Page>

Landing Page Deployed:

5.2. Landing Page, Services & Applications Implementation.

5.2.1. Sprint n

En la fase inicial de nuestro proyecto, nos propusimos llevar a cabo la implementación del diseño de nuestra Landing Page utilizando WebStorm como entorno de desarrollo. Esto implica que al concluir el Sprint, todas las secciones, ya sea Home, Services, Pricing, Testimonials o About Us, deben estar completadas. A continuación, adjuntamos imágenes que ilustran cómo gestionamos las tareas en Jira Software.

5.2.1.1. Sprint Planning 1.

Sprint #	Sprint 1
Sprint Planning Background	
Date	21-08-2024
Time	10:00 p.m
Location	Discord
Prepared By	Janover Saldaña
Attendees (to planning meeting)	Jhordi Carranza - Oscar Armas - Luis Villegas - Juan Llamccaya

Sprint #	Sprint 1
Sprint Goal & User Stories	
Sprint 1 Goal	Implementar la landing page incluyendo las distintas secciones acordadas y el requisito de cambio de idioma para la aplicación de ElixirControl
Sprint 1 Velocity	20
Sum of Story Points	20

5.2.1.2. Sprint Backlog 1.

Srpint #							Sprint 1
User Story	Work-Item / Task						
Id	Title	Id	Title	Description	Estimation (hours)	Assigned To	Status
US-001	Hipervínculos en el encabezado	T001	Definir enlaces del encabezado	Identificar y organizar los enlaces del encabezado.	1		Done
		T002	Implementar hipervínculos	Añadir hipervínculos en HTML	1		Done
		T003	Estilizar con CSS	Aplicar estilos básicos a los enlaces del encabezado	2		Done
		T004	Pruebas de funcionamiento	Asegurarse de que los hipervínculos funcionen correctamente.	1		Done
US-002	Información sobre beneficios o servicios de la app	T005	Recopilar información de beneficios	Obtener y redactar la información sobre los beneficios de la aplicación.	2		Done
		T006	Implementar en el sitio	Incluir la información de beneficios en la página correspondiente.	2		Done
		T007	Pruebas de visualización	Asegurarse de que la información esté visible y bien organizada.	1		Done
US-003	Mostrar los planes disponibles	T008	Definir estructura de precios	Establecer la estructura de precios a seguir en la sección.	4		Done
		T009	Implementar la funcionalidad de los planes	Crear la estructura de precios en HTML.	2		Done
		T010	Estilizar con CSS	Aplicar estilos a la estructura de precios.	2		Done
US-004	Información util en el footer	T011	Recopilar opiniones	Obtener y redactar opiniones de usuarios.	2		Done
		T012	Recopilar información útil	Organizar y decidir qué información incluir en el footer.	1		Done

Srpint #							Sprint 1	
		T013	Implementar contenido en el footer	Añadir la información útil al footer.	1		Done	
		T014	Estilizar footer	Dar estilo y formato al footer utilizando CSS.	1		Done	
		T015	Pruebas de visualización	Verificar que la información sea clara y legible.	1		Done	
US-005	Información sobre el producto	T016	Definir contenido sobre el producto	Recopilar y escribir la información relevante sobre el producto.	2		Done	
		T017	Implementar el contenido en el sitio	Incluir la información en la página adecuada.	2		Done	
		T018	Estilizar y ajustar visualización	Ajustar el diseño y la visualización del contenido.	2		Done	
		T019	Pruebas de visualización	Comprobar que la información sea visible y bien organizada.	1		Done	
US-018	Implementar opción de cambio de idioma	T020	Añadir la funcionalidad para cambiar el idioma de la aplicación en Angular.	5		Done		
		T021	Definir textos traducidos	Proveer textos traducidos en los diferentes idiomas soportados.	4		Done	
		T022	Pruebas de funcionalidad de idiomas	Verificar que el cambio de idioma funcione correctamente en toda la aplicación.	3		Done	
				T026	Implementar diseño responsive utilizando CSS	5		
T027	Pruebas de visualización en diferentes dispositivos	US-018	Implementar opción de cambio de idioma	Probar el diseño en varios dispositivos (móvil, tablet, desktop).	4		Done	
T028	Ajustes finales de responsividad			Realizar ajustes finales para asegurar una experiencia responsive óptima.	2		Done	

5.2.1.3. Development Evidence for Sprint Review.

Repository	Branch	Commit Id	Commit Message	Commit Message Body	Committed on (Date)
https://github.com/SV51-MetaSoft-App-Web/ElixirControl-Landing-Page	main	ca7aff9e22b71a4a96e8d823c0e61c347240df9c	Initial commit		13/08/2024
	develop	23d5ab43ce3f6f11fe2a2fd4f9b81111ca927cd3	feat: Added the base structure of the project.		21/08/2024
	feature/nav	c6a3d0698ef363310ca9a3c96d9d1ae658118f5b	feat: Added header and navigation section.		5/09/2024
	feature/hero	456e8089d0e5e7805a2819885be2972e3d221aef	feat(hero): Added hero section.		5/09/2024
	feature/services	c3e661b693063ca547c171ba3db89fd194255fc1	feat(services): Added services section		5/09/2024
	feature/services	1462bee8ba47ca2f878468b262302d7586e0633e	feat(services): Updated the styles of the services section.		5/09/2024
	feature/testimonials	79ef0471340e7e312c05d0e1c854cd7f0e562b19	feat(testimonials): Added Testimonials section.		06/09/2024
	feature/plans	d706c364023e2d4addf47df3aa8cf0e8dcf937ed	feat(plans): Added structure of plans in Spanish		7/09/2024
	feature/plans	2d7b2b5497c5ad9d77740d17b66fcc4fd529842	feat(plans): Added structure of plans in English		7/09/2024
	feature/plans	98620905ab1a4b86b797b55133f26e0ecc69dd28	feat(plans): Added styles for the plans section.		7/09/2024
	feature/about-the-team	775e984a0b45a388f19cfcea2a4cbf42e82f29ba	feat(about-the-team): Added content and styles to the About Team section		7/09/2024
	feature/about-the-team	9b3fe2fe56a55c7ecd7e88eea28a0e5c83d7ee29	feat(about-the-team): Added content and styles to the About Team section		7/09/2024
	feature/footer	636f5902fdec78f18206aa842c9ed10eb591ac05	feat/footer): Added content		7/09/2024

Repository	Branch	Commit Id	Commit Message	Commit Message Body	Committed on (Date)
			and styles to the footer section.		
	feature/about-the-app	f5ddbc9174940b126b2dbfb87c58a8ded713beb7	feat(about-the-app): Added content and styles in about the app section.		7/09/2024

5.2.1.4. Testing Suite Evidence for Sprint Review.

Para este primer sprint no se realizaron testing.

5.2.1.5. Execution Evidence for Sprint Review.

Después de completar el Sprint 1, logramos implementar todas las secciones de nuestra Landing Page para garantizar una visualización perfecta. Además, le dimos un formato atractivo que captura la atención del usuario hacia sus diferentes componentes. También agregamos métodos de navegación en la página, como botones ubicados al principio, que te permiten moverte fácilmente de una sección a otra. A continuación, te mostraremos los avances a través de imágenes del resultado obtenido.

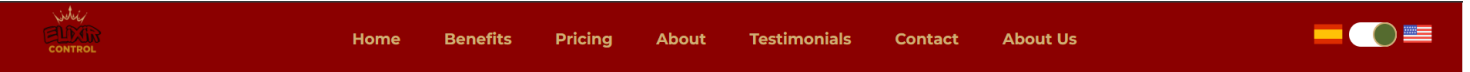
Es importante destacar que el objetivo principal de la Landing Page es convertir a los visitantes en futuros clientes o usuarios habituales de nuestro servicio. Para lograrlo, utilizamos llamados a la acción (Call To Action) que los guían hacia la aplicación web.

A continuación, te presentamos capturas de pantalla del desarrollo de la Landing Page:

Encabezado y botones de desplazamiento:

En la parte superior, se encuentra el encabezado (Header) que incluye botones de inicio (Home), beneficios (benefits), Pricing (Pricing), sobre la aplicación (about), testimonios de usuarios (testimonials), un formulario para que nos contacten (Contact), un apartado para saber sobre el equipo (About us) y un botón para cambiar el idioma entre inglés y español. Estos elementos permiten a los visitantes desplazarse fácilmente a la sección que deseen visualizar.

Imagen 01: Encabezado y botones de desplazamiento



Sección Hero:

Se presenta la sección "Hero", que incluye una breve descripción y una frase representativa de ElixirControl. Además, permite iniciar el uso del servicio web y proporciona una imagen relacionada con el mismo.

Imagen 02: Sección Hero

"From the grape to the bottle, everything under control."

ElixirControl: the tool that transforms your winery into a flawless operation.

Start now

Sección Services:

Se presenta la sección de servicio que ofrecemos para nuestros segmentos objetivos. En esta sección, se describen los beneficios y características de ElixirControl, lo que permite a los visitantes conocer más sobre el servicio.

Imagen 03: Sección Services

Services Offered by ElixirControl

Inventory Management

ElixirControl streamlines inventory management with precise control over supplies and finished products, avoiding stock shortages or surpluses and ensuring the availability of resources for continuous and profitable production.

Vinification Process Control

ElixirControl adapts to the specific needs of each producer, ensuring rigorous control at every stage of vinification to maintain consistency in the final product's quality.

Client Portfolio Management

ElixirControl centralizes client portfolio management, facilitating the tracking of interactions, orders, and needs, which helps identify sales opportunities and strengthen business relationships, preventing missed business opportunities.

Order Tracking

ElixirControl offers detailed tracking of orders from request to delivery, keeping distributors and clients informed, improving coordination, and reducing delays.

Sección Pricing:

En la sección de precios, se detallan los planes y precios de ElixirControl. Esta información es esencial para que los visitantes conozcan las opciones disponibles y puedan elegir la que mejor se adapte a sus necesidades.

Imagen 04: Sección Pricing

Our Plans

Elixir Start

This plan is aimed at small producers looking for a basic solution to optimize their production and commercial processes.

- Inventory management: Control of supplies and finished products limited to 50 products.
- Winemaking process control: Basic management of up to 2 phases of the process.
- Customer portfolio management: Up to 50 clients with basic order history.
- Support: Email support.

Free

Elixir Pro

Ideal for medium producers who need a robust solution to optimize the production chain and improve customer management.

- Inventory management: Control of up to 500 products, with automatic alerts for low or excess stock.
- Winemaking process control: Comprehensive management of all phases, adaptable to different types of wines.
- Customer portfolio management: Up to 100 clients.
- Order tracking: Simultaneous monitoring of up to 50 orders.
- Support: Email and live chat support.

\$49 USD/month

Elixir Premium

Designed for large producers or companies with advanced management needs in production and sales.

- Inventory management: Unlimited product control with advanced reports and stock predictions.
- Winemaking process control: Complete and personalized management at each stage, with historical data integration for quality optimization.
- Customer portfolio management: Unlimited management with behavior analysis and advanced segmentation.
- Order tracking: Unlimited tracking, with integration to distribution systems and real-time tracking.
- Support: Priority 24/7 with personalized attention.

\$99 USD/month

Sección About the App:

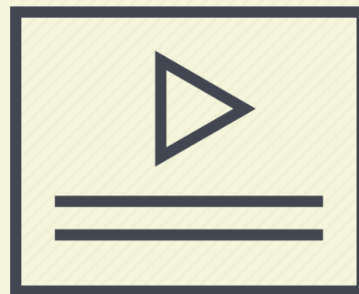
En esta sección, se presenta información detallada sobre la aplicación ElixirControl, sus características y funcionalidades. Esto permite a los visitantes conocer más sobre la aplicación y cómo puede ayudarles en su día a día.

Imagen 05: Sección About the App

About the App

ElixirControl

ElixirControl is designed to adapt to the specific needs of each user, providing key tools for effective management of the wine and pisco production process, improving efficiency and optimizing the experience for both producers and distributors.



Sección Testimonials:

En la sección de testimonios, se presentan opiniones y comentarios de usuarios reales que han utilizado ElixirControl. Esto ayuda a generar confianza en los visitantes y a mostrarles la experiencia positiva de otros usuarios.

Imagen 06: Sección Testimonials

What Do Our Clients Say?



Armando Salazar

(Elixir Start)

"ElixirControl helps me avoid stock problems. It is perfect for my small winery."



Sandra Ávila

(Elixir Start)

"Customer management is much easier. Everything is now centralized and organized."



Lesly Díaz

(Elixir Pro)

"Order tracking and winemaking control have improved our production. Highly recommended."



Pablo Medina

(Elixir Premium)

"Full control of the winemaking process has improved our quality. Great technical support."



Griselda Blanco

(Elixir Pro)

"Order tracking has streamlined our deliveries. Our customers are happier."



Susana Flores

(Elixir Premium)

"We have optimized production and delivery. Ideal for large, growing wineries."

Sección About the Team:

En la sección "About the Team", se presenta información sobre el equipo de desarrollo de ElixirControl. Esto permite a los visitantes conocer a las personas detrás del servicio y generar confianza en la calidad y profesionalismo del equipo.

About the team MetaSoft



Sección Contact:

En la sección de contacto, se presenta un formulario que permite a los visitantes enviar consultas, comentarios o solicitudes de información sobre ElixirControl. Esto facilita la comunicación con los usuarios y permite responder a sus necesidades de manera eficiente.

Contáctanos

ElixirControl

Nombres y Apellidos

E-mail

Asunto

Cuentanos: ¿Cómo podemos ayudarte?

Enviar Mensaje

Contáctanos y conoce más sobre nuestra Startup y Elixir Control



Footer:

En el pie de página (Footer), se incluyen enlaces a las redes sociales de ElixirControl, información de contacto y un botón para volver al inicio de la página. Esto permite a los visitantes acceder a más información y mantenerse conectados con el servicio.

"Quality and efficiency assured in every bottle"

[f](#) [i](#) [t](#)

Company
[Services](#)
[About the App](#)
[Pricing](#)
[Testimonials](#)

Contact
[Help/FAQ](#)

More
[News](#)
[About Us](#)

© Copyright, MetaSoft 2024. All rights reserved.

[Terms and Conditions](#)

5.2.1.6. Services Documentation Evidence for Sprint Review.

En el primer Sprint el equipo de desarrollo de MetaSoft ha diseñado, programado y puesto en funcionamiento el sitio web (Landing Page) Para presentar la aplicación Web propuesta denominada "ElixirControl". En este sitio web (Landing Page), se logrará visualizar varias secciones que ilustran en que consiste "ElixirControl", cada integrante del equipo de desarrollo de Metasoft estuvo a cargo de una sección en específico.

End Point	Funciones
https://sv51-metasoft-app-web.github.io/ElixirControl-Landing-Page/	Mostrar la Landing Page Desplegada

5.2.1.7. Software Deployment Evidence for Sprint Review.

Para la implementación de nuestro sitio web, optamos por utilizar GitHub Pages. En este proceso, creamos un repositorio en GitHub donde gestionamos el control de versiones. En la sección de Configuración, publicamos el proyecto almacenado en la rama "realease-V1.0" que previamente se encontraba en la rama release-1.0.

[Landing Page ElixirControl - https://sv51-metasoft-app-web.github.io/ElixirControl-Landing-Page/](https://sv51-metasoft-app-web.github.io/ElixirControl-Landing-Page/)

5.2.1.8. Team Collaboration Insights during Sprint.

En esta entrega, nuestra meta principal fue la implementación de la Landing Page. Para llevar a cabo este objetivo, hicimos uso de diversas herramientas como GitHub, Visual Studio Code, WebStorm, HTML, CSS y JavaScript. A continuación, vamos a presentar los diagramas de flujo que representan los commits realizados por cada miembro del equipo MetaSoft:

A continuación se muestra la cantidad de commits realizadas por cada integrante del equipo durante el desarrollo de la landing page.

Pulse
Contributors
Community
Community Standards
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
People

August 8, 2024 – September 8, 2024

Period: 1 month

Overview

0 Active pull requests

0 Active issues

0 Merged pull requests

0 Open pull requests

0 Closed issues

0 New issues

Excluding merges, 5 authors have pushed 15 commits to main and 20 commits to all branches. On main, 0 files have changed and there have been 0 additions and 0 deletions.



Los siguientes gráficos ofrecen una representación visual de las clonaciones registradas en nuestro repositorio, junto con la fecha en que cada una de estas acciones se llevó a cabo. Además, se presenta información sobre la cantidad de visitantes que ha tenido el repositorio de nuestro equipo a lo largo del tiempo.



6. Conclusiones, Bibliografía y Anexos.

Conclusiones

Bibliografia

Anexos

Sección	Características del video	Sobre el contenido	
Needfinding Interviews	Cantidad de videos: 1 Nomenclatura: upc-pre-202402-si730-sv51-metasoft-needfinding-sprint-1 Formato: .mp4	Consolida todas las entrevistas realizadas	Link: https://upcedupe-my.sharepoint.com/:v:/g/personal/u201923571_upc_edu_pe/EZw-9tmmOSJPse=Pyj62l&nav=eyJyZWZlcnJhbEluZm8iOncicmVmZXJyYWxBcHAIoiJTdHJIYW1XZWJBcHAIJCjYz
Exposición	Cantidad de videos: 1 Nomenclatura: upc-pre-202402-si730-sv51-metasoft-expo-tb1 Formato: .mp4	Consolida las exposiciones de la TB1	Link: https://upcedupe-my.sharepoint.com/:v:/g/personal/u201923571_upc_edu_pe/ERTxpx1uyvJKmnav=eyJyZWZlcnJhbEluZm8iOncicmVmZXJyYWxBcHAIoiJTdHJIYW1XZWJBcHAIJCjYzZWZlcnJhbF
Prototypes Navigation / Product Navigation	Cantidad de videos: 1 Nomenclatura: upc-pre-202402-si730-sv51-metasoft-prototype-navigation-sprint-1 Formato: .mp4	Consolida demostración del flujo de navegación de las aplicaciones, priorizando los user flows relacionados con el core business.	Link: https://upcedupe-my.sharepoint.com/:v:/g/personal/u201923571_upc_edu_pe/EQ3ShXzJBIJAllne=e6Hn8s&nav=eyJyZWZlcnJhbEluZm8iOncicmVmZXJyYWxBcHAIoiJTdHJIYW1XZWJBcHAIJCjYz
Validation Interviews			
About the Product			
About the Team			