

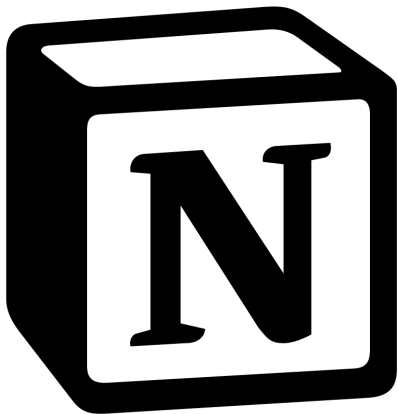
Capítulo VI: Product Implementation, Validation & Deployment.

6.1. Software Configuration Management.

6.1.1. Software Development Environment Configuration.

Project Management

Se utilizaron diversas herramientas que permitieron una comunicación en tiempo real y una edición rápida y compartida, lo que optimizó la eficiencia del trabajo colaborativo. Entre las herramientas empleadas destacan los servicios de Google (Google Drive, Notion, Google Meet).



Para la gestión de versiones y repositorios, se utilizó GitHub. Estas herramientas permiten registrar de manera ordenada cada uno de los commits, garantizando un control adecuado del desarrollo.



Product UX/UI Design

Para la creación de los segmentos objetivos y sus mapeos, así como los As-Is y To-Be Scenario Maps, se utilizó Miro y UXPressia. Estas herramientas facilitaron la construcción de escenarios que reflejan el comportamiento del usuario. En cuanto a los mockups, wireframes, wireflows y el prototipo de la aplicación, Figma fue la herramienta principal, gracias a su versatilidad y capacidad de colaboración.



Software Development

Para el desarrollo del frontend, se empleó Visual Studio Code como entorno principal, en combinación con tecnologías web como HTML, CSS, Bootstrap y JavaScript. Estas tecnologías permitieron la construcción del Landing Page y de las interfaces web de manera responsive y dinámica. HTML fue utilizado para la estructura, CSS para la presentación, Bootstrap para garantizar un diseño adaptable, y JavaScript para añadir interactividad y animaciones.



Visual Studio Code

El desarrollo de la aplicación móvil se llevó a cabo utilizando el framework Flutter, conocido por ser multiplataforma y por la flexibilidad que brinda para la creación de interfaces personalizadas. El lenguaje de programación utilizado fue Dart.



En el desarrollo de la aplicación web, se utilizó Vue.js, un framework progresivo de JavaScript que permite la creación de interfaces de usuario dinámicas e interactivas.



Por otro lado, el desarrollo del backend fue realizado en Node.js, un entorno de ejecución que permite utilizar JavaScript en el servidor.

Software Testing

Para las pruebas del Landing Page, se utilizó la extensión **LiveShare** de Visual Studio Code, que permite crear un entorno de **localhost** donde se pueden visualizar en tiempo real los cambios realizados en el código. Para las pruebas de aceptación, se empleó **Gherkin** y los resultados se subieron al repositorio en GitHub.

Software Deployment

El Landing Page fue desplegado utilizando **GitHub Pages**, que permite alojar sitios web estáticos de manera gratuita y actualizar el contenido de forma eficiente.

Software Documentation

La documentación del software se realizó mediante comentarios en el código HTML, ya que este es un lenguaje de marcado y no requiere una documentación exhaustiva como los lenguajes de programación orientados a objetos.

6.1.2. Source Code Management.

Para la gestión del código fuente y evitar conflictos, los proyectos se organizaron en diferentes repositorios dentro de una organización en GitHub, distribuidos de la siguiente manera:

- **Landing Page:** Repositorio del Landing Page
- **Aplicación Web:** Repositorio de la aplicación web
- **Aplicación Móvil:** Repositorio de la aplicación móvil
- **Backend:** Repositorio del backend
- **Aplicación Embebida:** Repositorio de la aplicación embebida

Se utilizó **GitFlow** para la administración de versiones, haciendo uso de las ramas **main**, **develop** y **feature**. Para los commits, se siguió el formato de mensajes **Conventional Commits**, lo que permite tener un registro claro y organizado de los cambios realizados en el proyecto.

Cada repositorio cuenta con las siguientes ramas:

- **main:** Contiene las versiones estables.
- **develop:** Contiene las funcionalidades en desarrollo y en constante actualización.
- **feature:** Para el desarrollo de nuevas funcionalidades.

El formato utilizado para las ramas **feature** fue **feature/<user-story/technical-story>**, y los commits siguieron la convención **<type>[optional scope]: <description>**.

6.1.3. Source Code Style Guide & Conventions.

La adherencia a las siguientes pautas, convenciones y principios es crucial para mantener la calidad estructural del software, mejorar la legibilidad del código fuente y facilitar el mantenimiento.

HTML:

- Declarar siempre el tipo de documento con **<!DOCTYPE html>**.
- Utilizar nombres de etiquetas y atributos en minúscula.
- Cerrar todas las etiquetas.

- Siempre utilizar comillas para los valores de los atributos.
- Especificar los atributos alt, width y height para las imágenes.
- No omitir la etiqueta < title> ni los metadatos (< meta>).

```
<!DOCTYPE html>
<html>
<head>
  <title>Example Page</title>
  <meta charset="UTF-8">
  <meta name="description" content="This is an example page">
</head>
<body>
  
</body>
</html>
```

CSS:

- Usar nombres de clases generales y descriptivos.
- Utilizar nombres de clase cortos y concisos.
- Separar palabras en nombres de clase con guiones.
- Evitar los selectores de ID.
- Usar propiedades abreviadas cuando sea posible.

JavaScript:

- Usar nombres cortos y descriptivos para variables, funciones, etc.
- Evitar el uso de variables globales (var).
- Comentar líneas de código complejas para facilitar la comprensión.
- Utilizar notaciones simples y comprensibles.

```
// Declaración de variables
let nombreUsuario = "Juan";

// Definición de función
function saludar() {
  return "Hola, " + nombreUsuario + "!";
}

// Uso de función
console.log(saludar());
```

Dart:

- Usar nombres de clases, métodos y variables en minúsculas.
- Utilizar nombres de clases en singular y en mayúscula.
- Utilizar nombres de métodos y variables en minúscula.

- Utilizar nombres de métodos en camelCase.
- Utilizar nombres de variables en minúscula y separados por guiones bajos.

```
// Declaración de variable utilizando camelCase
String nombreUsuario = 'Juan';

// Definición de función utilizando camelCase
void saludar() {
    print('Hola, $nombreUsuario!');
}
```

Vue:

- Usar nombres de componentes en singular y en mayúscula.
- Utilizar nombres de métodos y variables en minúscula.
- Seguir las convenciones de Vue.js para la estructura del proyecto, como el uso de componentes y la organización del código. Por ejemplo, dividir la interfaz de usuario en componentes reutilizables y mantener una estructura de carpetas lógica para los archivos de componentes.
- Utilizar la sintaxis de Vue.js de manera consistente en los archivos de componentes. Por ejemplo, utilizar la notación v-bind para enlazar atributos y v-on para manejar eventos.

```
<!-- Ejemplo de componente Vue -->
<template>
  <div>
    <p>{{ mensaje }}</p>
    <button @click="cambiarMensaje">Cambiar Mensaje</button>
  </div>
</template>

<script>
export default {
  data() {
    return {
      mensaje: 'Hola Mundo!'
    };
  },
  methods: {
    cambiarMensaje() {
      this.mensaje = '¡Hola Vue!';
    }
  }
};
</script>
```

Node.js:

- Usar nombres de variables y funciones en minúsculas.
- Utilizar nombres descriptivos y significativos para las variables y funciones.

- Seguir las convenciones de Node.js para la estructura del proyecto, como el uso de módulos y la organización del código. Por ejemplo, dividir la lógica de la aplicación en módulos reutilizables y mantener una estructura de carpetas lógica para los archivos de módulos.
- Utilizar módulos npm y gestionar las dependencias de manera adecuada. Por ejemplo, especificar las dependencias en un archivo package.json y utilizar herramientas como npm o yarn para instalar y gestionar paquetes.

```
proyecto-node/  
├── src/  
│   ├── index.js  
│   ├── controllers/  
│   │   └── usuarioController.js  
│   ├── models/  
│   │   └── usuarioModel.js  
│   └── routes/  
│       └── usuarioRoutes.js  
├── package.json  
└── README.md
```

Spring: Se siguieron las convenciones y guías de estilo de código de la documentación oficial de Spring Boot y se destaca lo siguiente:

- **Uso de Anotaciones:** Utilizar anotaciones como @Controller, @Service, @Repository y @Component para marcar clases y componentes específicos de Spring.
- **Convención de Paquetes:** Organizar los archivos y clases en carpetas que representen la estructura lógica de la aplicación, como controladores, servicios, repositorios, etc.
- **Convención de Nombres en Bases de Datos:** Utilizar la convención de nombres en bases de datos como snake_case para nombres de tablas y columnas. Spring Boot se encargará de mapear estos nombres a objetos Java.
- **Uso de Spring Data JPA:** Emplear Spring Data JPA para simplificar la interacción con la capa de persistencia y bases de datos.
- **Uso de Inyección de Dependencias:** Aplicar la inyección de dependencias utilizando el constructor de las clases.
- **Uso de @RestController:** Usar la anotación @RestController para marcar controladores que devuelven datos en formato JSON.
- **Seguridad con Spring Security:** Implementar la seguridad en la aplicación utilizando Spring Security para autenticación y autorización.

Java:

- Seguir las convenciones de nomenclatura de Java para nombres de variables, clases, métodos y paquetes. Por ejemplo, utilizar camelCase para nombres de variables y métodos, UpperCamelCase para nombres de clases, y utilizar nombres descriptivos que reflejen el propósito de la entidad.
- Utilizar comentarios Javadoc para documentar clases y métodos públicos. Esto es fundamental para proporcionar una descripción clara de la funcionalidad de las clases y métodos, así como para generar documentación automáticamente.

- Organizar el código en paquetes lógicos y utilizar la convención de nombres de paquetes de dominio invertido para evitar conflictos de nombres. Por ejemplo, el nombre de paquete `com.ejemplo.proyecto` indica que el proyecto pertenece al dominio `ejemplo.com`.

```
package com.ejemplo.proyecto.modelo;

/**
 * Clase que representa un usuario en el sistema.
 */
public class Usuario {
    private String nombre;
    private int edad;

    /**
     * Constructor de la clase Usuario.
     * @param nombre El nombre del usuario.
     * @param edad La edad del usuario.
     */
    public Usuario(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    /**
     * Método para obtener el nombre del usuario.
     * @return El nombre del usuario.
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * Método para establecer el nombre del usuario.
     * @param nombre El nuevo nombre del usuario.
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    /**
     * Método para obtener la edad del usuario.
     * @return La edad del usuario.
     */
    public int getEdad() {
        return edad;
    }

    /**
     * Método para establecer la edad del usuario.
     * @param edad La nueva edad del usuario.
     */
    public void setEdad(int edad) {
```

```
        this.edad = edad;
    }

    /**
     * Método para imprimir los datos del usuario.
     */
    public void imprimirDatos() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
    }
}
```

6.1.4. Software Deployment Configuration.

A continuación se describe el proceso de despliegue de las aplicaciones, así como la configuración de los servidores e infraestructura necesarios para su correcto funcionamiento.

Landing Page: Para el despliegue de la Landing Page, se utilizó GitHub Pages, una plataforma gratuita que permite alojar sitios web estáticos directamente desde un repositorio de GitHub. El proceso de despliegue se realizó de la siguiente manera:

1. Crear un repositorio en GitHub con el código de la Landing Page.
 2. Acceder a la configuración del repositorio y habilitar GitHub Pages.
 3. Seleccionar la rama y la carpeta de origen del sitio web.
 4. Guardar la configuración y obtener la URL del sitio web desplegado.
 5. Finalmente, acceder a la URL del sitio web para verificar que se haya desplegado correctamente.
- Enlace de la Landing Page desplegada: XXXX

FrontEnd: Para el despliegue de la aplicación se ha usado los servicios que ofrecen netlify una plataforma de alojamiento web que ofrece integración continua y despliegue automático desde repositorios de Git. El proceso de despliegue fue el siguiente:

- Preparación del Repositorio: Asegurarse que la página web esté almacenada en un repositorio Git, como GitHub, GitLab.
- Crear una Cuenta en Netlify: Registrarse en la plataforma.
- Conectar el Repositorio: Inicia sesión en Netlify y ve al panel de control. Hacer clic en el botón "New site from Git" (Nuevo sitio desde Git). Selecciona tu proveedor de servicios de alojamiento de Git (por ejemplo, GitHub) y autoriza la conexión con tu cuenta. Seleccionar el repositorio que contiene la página web.
- Configurar las Opciones de Despliegue: Netlify detectará automáticamente la configuración de tu proyecto. Si necesitas ajustes adicionales, como la configuración del directorio de compilación, puedes establecerlos en la sección de configuración de tu sitio.
- Despliegue Automático: Activa la opción de "Deploy site" (Desplegar sitio) para habilitar el despliegue automático cada vez que realices cambios en tu repositorio.

- **Verificar el Despliegue:** Una vez que se complete el despliegue, Netlify te proporcionará una URL única para acceder a tu página web.
- BackEnd:** Para el despliegue del backend se ha utilizado los servicios de Railway, una plataforma de alojamiento de aplicaciones web que permite desplegar aplicaciones de Node.js, Python, Ruby, Java, PHP, Go y Docker. El proceso de despliegue fue el siguiente:
- **Preparación del Repositorio:** Asegurarse de que el backend esté almacenado en un repositorio Git, como GitHub, GitLab.
 - **Crear una Cuenta en Railway:** Registrarse en la plataforma.
 - **Crear un Nuevo Proyecto:** Inicia sesión en Railway y ve al panel de control. Crea un nuevo proyecto y selecciona "Backend" como tipo de proyecto. **Conectar el Repositorio:** Selecciona tu proveedor de servicios de alojamiento de Git (por ejemplo, GitHub) y autoriza la conexión con tu cuenta. Selecciona el repositorio que contiene tu backend.
 - **Configurar el Entorno:** Railway detectará automáticamente el tipo de backend que estás utilizando y configurará el entorno según sea necesario. Si tu backend necesita variables de entorno específicas, como claves API o configuraciones de base de datos, puedes establecerlas en la sección de configuración de tu proyecto.
 - **Despliegue Automático:** Activar la opción de "Auto Deploy" (Despliegue Automático) para habilitar el despliegue automático cada vez que realices cambios en tu repositorio.
 - **Verificar el Despliegue:** Una vez que se complete el despliegue, Railway te proporcionará una URL única para acceder a tu backend.

6.2. Landing Page, Services & Applications Implementation.

6.2.1. Sprint 1

6.2.1.1. Sprint Planning 1.

Sprint Planning Meeting	
Sprint #	Sprint 1
Sprint Planning Background	
Date	2024-09-21
Time	10:00 AM
Location	Google Meet

Sprint Planning Meeting	
Prepared By	Benedetti Rivas. Lucas Sebastian
Attendees	Benedetti Rivas. Lucas Sebastian, Curi Montero. Jonatan Omar, Valentin Ricaldi. Willy David, Valverde Salazar. Clara Angie
Sprint n – 1 Review Summary	Se completó la funcionalidad básica de la aplicación web, incluye el login, creacion de perfil, asi como poder verificar los campos del usuario. El Product Owner sugirió mejoras en la usabilidad y claridad de los testimonios de usuarios (US 3). y implementar los cambios respectivos a la autenticacion en el siguiente Sprint
Sprint n – 1 Retrospective Summary	El equipo trabajó de manera colaborativa y cumplió con los tiempos planificados. El diseño inicial de la landing page fue bien recibido. Hubo problemas de comunicación durante las fases iniciales. Se decidió mejorar la sincronización entre diseñadores y desarrolladores.
Sprint Goal & User Stories	
Sprint 1 Goal	Nuestro enfoque es entregar una landing page completamente funcional, con un sistema de registro de usuarios operativo. Creemos que esto proporcionará claridad y confianza a los usuarios potenciales. Esto se confirmará cuando los usuarios completen el proceso de registro sin problemas.
Sprint 1 Velocity	El equipo ha estimado que puede completar 20 Story Points durante este Sprint. Se trabajará principalmente en la construcción de la página principal y en los procesos de registro e inicio de sesión.
Sum of Story Points	La suma de los Story Points para este Sprint es 20 Story Points.

6.2.1.2. Sprint Backlog 1.

El objetivo principal de este Sprint es completar la funcionalidad clave de la aplicación, enfocándonos en la implementación de la landing page y los procesos de registro e inicio de sesión. Se utilizará Trello como herramienta de control para gestionar el progreso de las tareas asignadas. A continuación, se presenta un screenshot del Board en Notion, junto con el enlace público para acceder al mismo.

Screenshot del Board



Enlace al Board en Trello: [Acceder al Board](#)

Tabla de Control de Estado del Sprint

Id		Title		Description	Estimation (Hours)	Assigned To	Status
----	--	-------	--	-------------	--------------------	-------------	--------

Id	Title	Id	Title	Description	Estimation (Hours)	Assigned To	Status
US 1	Landing Page	Task 1.1	Diseño inicial de la landing page	Crear el diseño visual y estructura básica de la landing page.	8	Valverde Salazar Clara Angie	Done
US 1	Landing Page	Task 1.2	Desarrollo de Frontend	Implementar el código frontend de la landing page, con todos los elementos clave.	12	Benedetti Rivas Lucas Sebastian	Done
US 2	Funcionalidades	Task 2.1	Crear resumen de funcionalidades	Desarrollar la sección que muestre un resumen claro de las funcionalidades principales de la aplicación.	6	Benedetti Rivas Lucas Sebastian	Done
US 3	Testimonios	Task 3.1	Cargar testimonios	Implementar la carga dinámica de testimonios de usuarios.	4	Curi Montero, Jonatan Omar	Done
US 4	CTA	Task 4.1	Implementar CTA	Crear una llamada a la acción visible y funcional que redirija a la página de registro.	5	Curi Montero, Jonatan Omar	Done
US 5	Registro de Usuario	Task 5.1	Implementar validación de datos	Implementar la validación de datos en el formulario de registro.	6	Benedetti Rivas Lucas Sebastian	Done

Id		Title		Description	Estimation (Hours)	Assigned To	Status
US 6	Inicio de Sesión	Task 6.1	Desarrollar sistema de autenticación	Crear la lógica de autenticación para el inicio de sesión en la aplicación.	10	Benedetti Rivas Lucas Sebastian	Done
US 14	Monitoreo de Cosechas	Task 14.1	Registro y monitoreo de cosechas	Implementar el registro de los cultivos y el monitoreo de su crecimiento basado en los datos ingresados.	8	Benedetti Rivas Lucas Sebastian	Done
US 17	Noticias Agrícolas	Task 17.1	Integrar noticias agrícolas	Implementar la visualización de una lista de noticias relevantes sobre agricultura, actualizadas en tiempo real.	3	Benedetti Rivas Lucas Sebastian	Done
-	Constraint	Task C1	Revisar seguridad y cumplimiento	Asegurarse de que las medidas de seguridad básicas están cubiertas antes de implementar el login.	5	Benedetti Rivas Lucas Sebastian	In-Process

6.2.1.3. Development Evidence for Sprint Review

Repository	Branch	Commit Id	Commit Message	Commit Message Body	Committed on (Date)
user/repositoryname	feature/news	14ca4e3	feat: news added	news view added	Sep 26, 2024
user/repositoryname	feature/news	c7b912f	feat: dashboard news component added	component news	Sep 26, 2024

Repository	Branch	Commit Id	Commit Message	Commit Message Body	Committed on (Date)
user/repositoryname	feature/dashboard	3b9c0fe	feat: dashboard added	Dashboard functionality added for project management	Sep 26, 2024
user/repositoryname	feature/login	3c67a2e	feat: login added	Login functionality integrated	Sep 26, 2024
user/repositoryname	feature/createUser	e2b1d4a	feat: create user	User creation functionality implemented	Sep 26, 2024
user/repositoryname	feature/base	7a1b4f5	feat: base project added	Initial base project structure added	Sep 26, 2024

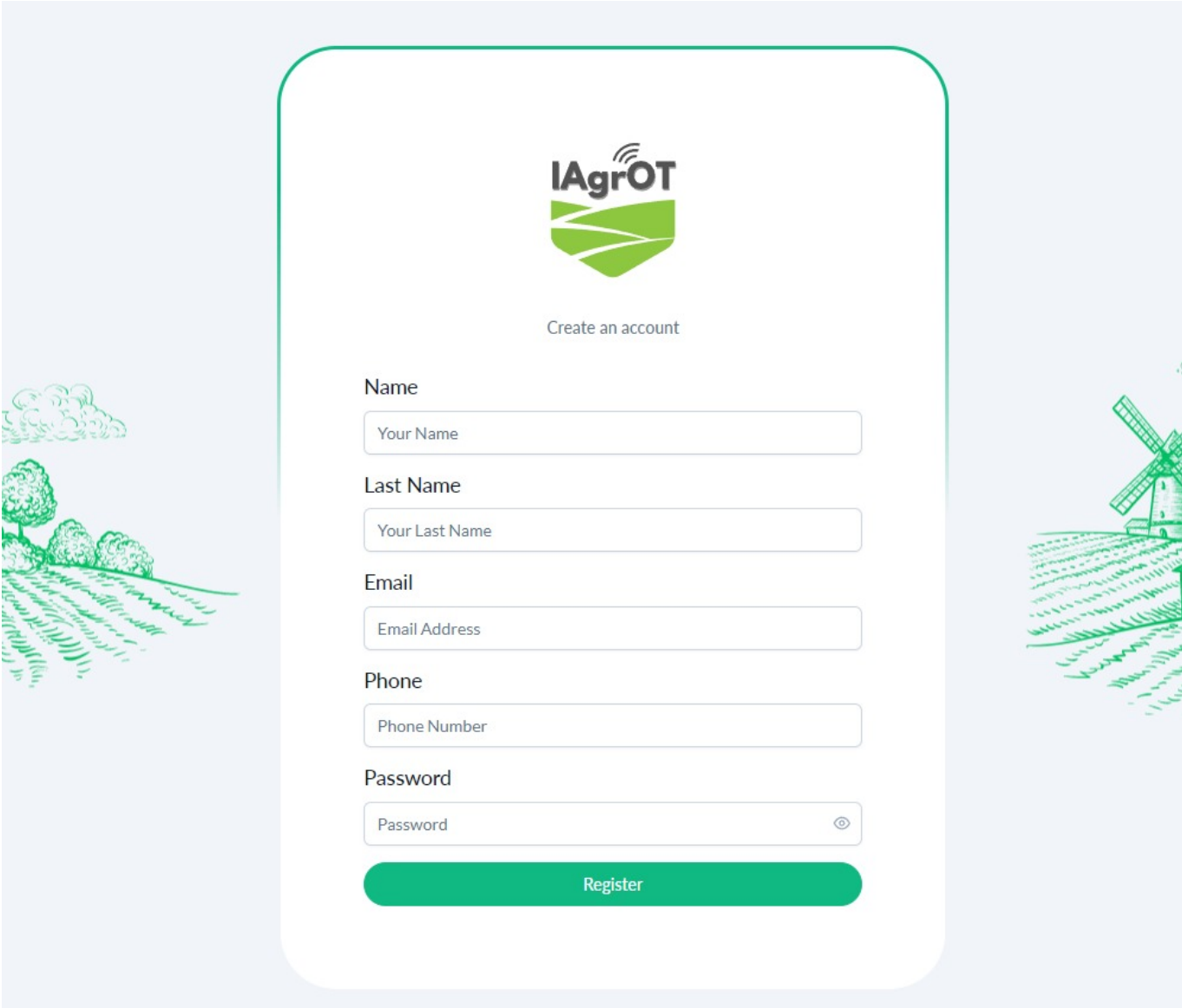
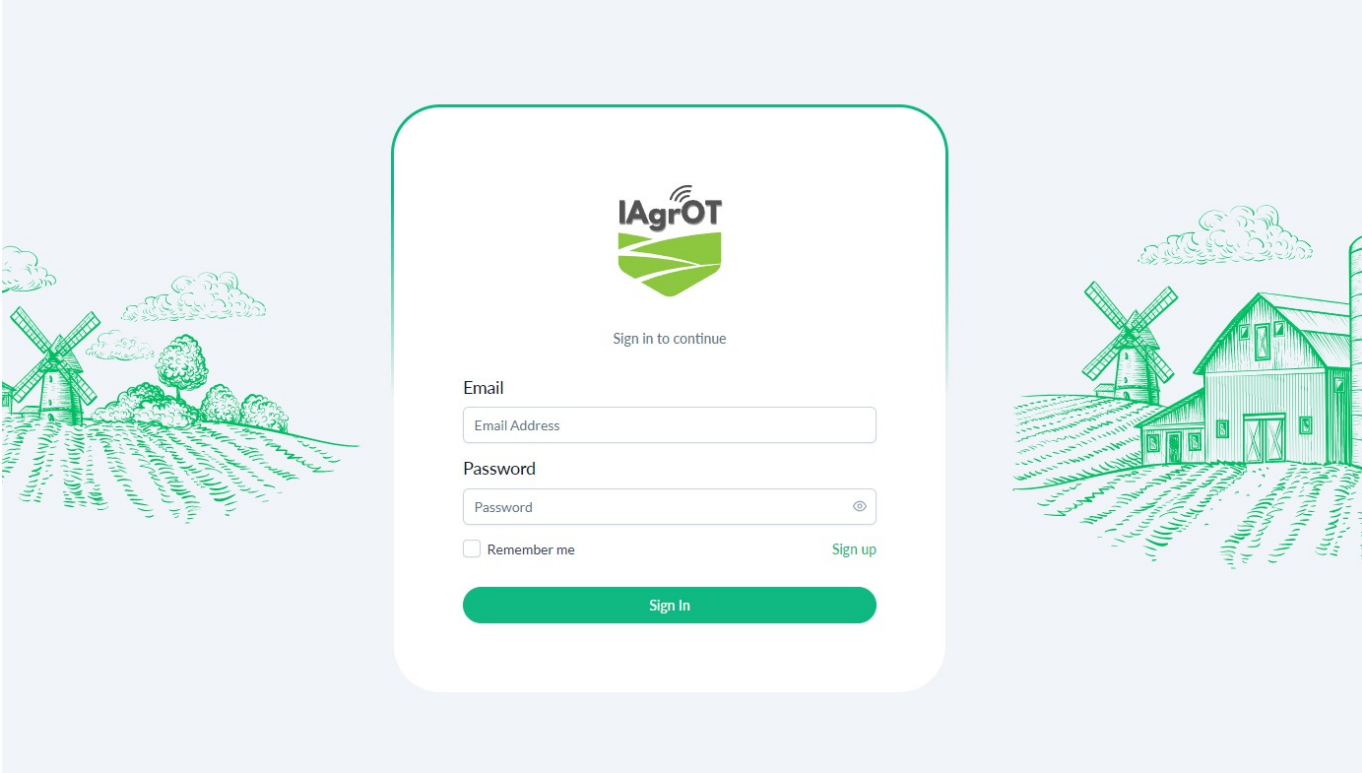
6.2.1.4. Testing Suite Evidence for Sprint Review

Durante este Sprint, se desarrollaron y ejecutaron pruebas unitarias, de integración y de aceptación para garantizar la calidad de los Web Services relacionados con las User Stories especificadas en el Sprint. Se utilizaron archivos `.feature` bajo el enfoque BDD, con escenarios para validar el comportamiento esperado de los servicios. A continuación, se presenta una relación de los tests diseñados y sus commits correspondientes.

FALTA TABLA CON COMMITS DE TESTING

6.2.1.5. Execution Evidence for Sprint Review

Durante este Sprint, se implementaron y probaron las funcionalidades principales de la aplicación web, relacionadas con las User Stories especificadas. Se desarrollaron interfaces clave, y se realizaron pruebas visuales para validar el correcto funcionamiento de las vistas y su interacción. A continuación, se presentan capturas de pantalla de la aplicación en su estado actual, mostrando las características implementadas y los avances logrados en este ciclo de desarrollo.



HOME

Dashboard

PROFILE

Edit Profile

Exit

FEATURES

Harvest

Crops

Tasks

News


DOCUMENTATION

Report Project

View Source

Harvest Data

Corn




Condition: Excellent - ☆☆☆☆

Humidity: 59%

Temperature: 32 C

Harvest Date: November

Rice



Condition: Excellent - ☆☆☆☆

Humidity: 59%

Temperature: 32 C

Harvest Date: December

Harvests

2

52% + since last week

Crops

281 (Static)

52 newly registered

Tasks

281 (Static)

52 newly registered

HOME

Dashboard

PROFILE

Edit Profile

Exit

FEATURES

Harvest

Crops

Tasks

News


DOCUMENTATION

Report Project

View Source

Harvest Data

Corn




Condition: Excellent - ☆☆☆☆

Humidity: 59%

Temperature: 32 C

Harvest Date: November

Rice

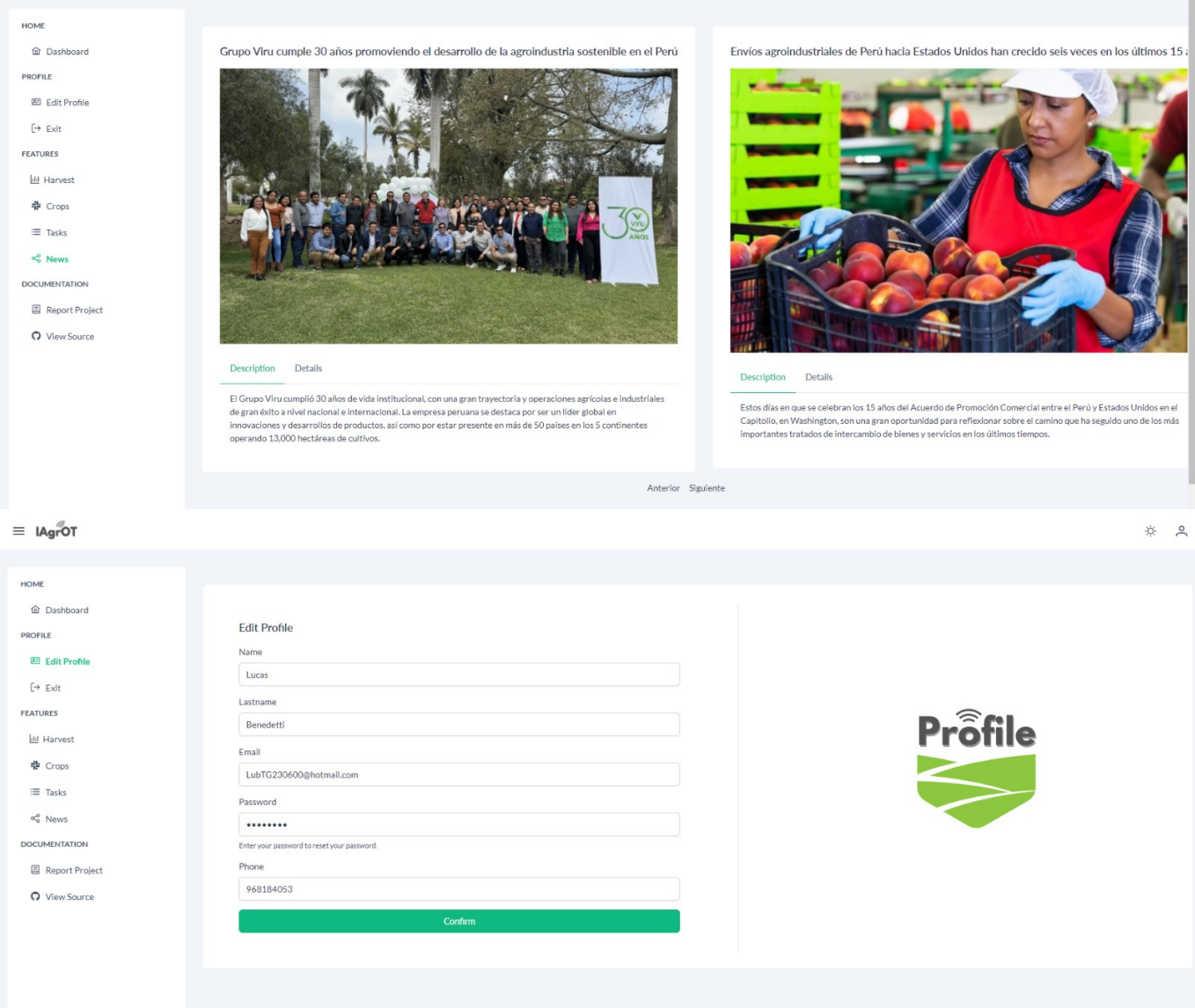


Condition: Excellent - ☆☆☆☆

Humidity: 59%

Temperature: 32 C

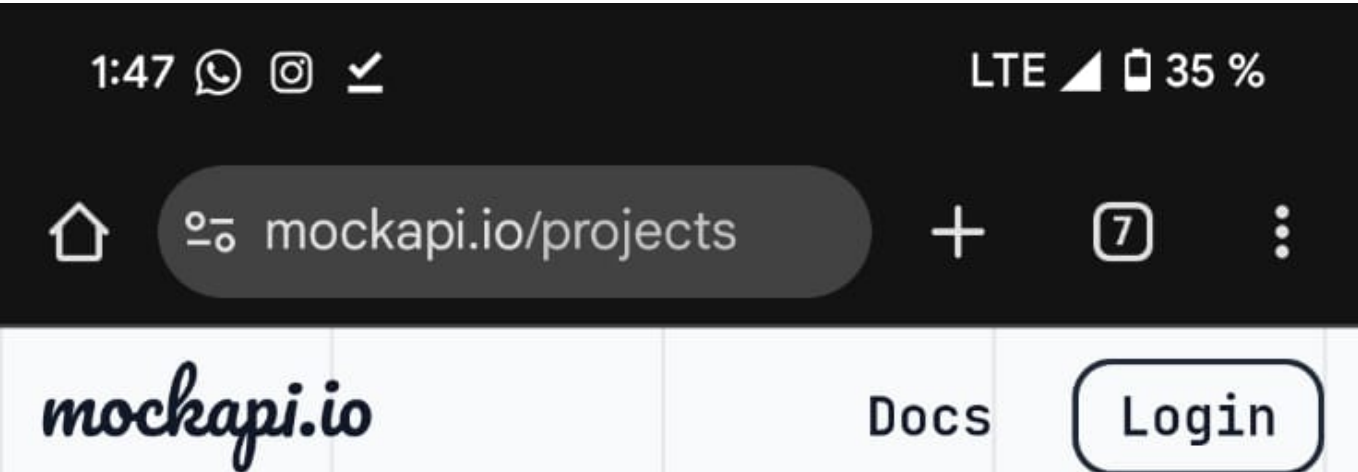
Harvest Date: December



6.2.1.6. Services Documentation Evidence for Sprint Review.

Durante este Sprint, se desarrolló la documentación de los servicios creados para el FakeAPI, proporcionando detalles sobre su estructura, funcionamiento y endpoints disponibles. Esta documentación asegura que los servicios puedan ser fácilmente comprendidos e integrados por otros desarrolladores o equipos. A continuación, se incluye la evidencia del trabajo realizado, con ejemplos de uso y las especificaciones técnicas relevantes para cada servicio del FakeAPI.

Se utilizo mockapi como fakeapi para este sprint



The easiest way to mock REST APIs

Quickly setup endpoints,
generate custom data, and
perform operations on it using
RESTful interface

[Get started](#)

Free

Projects	1
Resources	2
Custom response	<div>✖</div>

Tenemos dos instancias de mockapi debido a limitaciones con la versión gratuita

API endpoint

https://6529b2f755b137ddc83f18dc.mockapi.io/api/v1/:endpoint

New resource

Generate all

Reset all

HarvestData							
<div>50</div>							
CropData							
<div>50</div>							

User y news

API endpoint

https://66f23cbc415379191553727a.mockapi.io/api/v1/:endpoint

New resource

Generate all

Reset all

User							
<div>6</div>							
New							
<div>4</div>							

Se tienen los siguientes endpoints por cada contexto

On

GET /New

\$mockData

On

GET /New/:id

\$mockData

On

POST /New

\$mockData

On

PUT /New/:id

\$mockData

On


DELETE /New/:id

\$mockData

Close

Update



Durante este Sprint, se completó el despliegue de la landing page en Netlify, asegurando que la página esté accesible y funcionando correctamente. Se realizaron pruebas básicas de accesibilidad y rendimiento para garantizar su estabilidad. A continuación, se presentan capturas de pantalla que evidencian el estado del despliegue y el acceso a la landing page.



iagrot
Deploys from GitHub

Owned by G-Pulenta

Published on Sep 19 (8 days ago)



Deploys from github.com/Grupo01-Soluciones-IoT/Landing-Page.

Published `main@HEAD`

Auto publishing is on. Deploys from `main` are published automatically.

Runtime:	Not set
Base directory:	/
Package directory:	Not set
Build command:	Not set
Publish directory:	Not set
Functions directory:	netlify/functions
Deploy log visibility:	Logs are public
Build status:	Active

[Learn more about configuring builds in the docs ↗](#)

Functions region

Configure how functions are deployed for your site.

Region:

US East (Ohio) - us-east-2

[Learn more about functions region in the docs ↗](#)

6.2.1.8. Team Collaboration Insights during Sprint

Durante este Sprint, la colaboración del equipo fue clave para el avance en las tareas de desarrollo. Se realizaron diversos commits tanto en el desarrollo del frontend como en la creación del reporte en formato .md, lo que permitió mantener la documentación actualizada y el código alineado con los objetivos del sprint.

A continuación, se presenta una lista de los commits realizados, mostrando las contribuciones en ambas áreas y reflejando el trabajo colaborativo del equipo.

Reporte



Frontend App Web

