Vol. 21, n.°x, pp. 127-xxx, 2022
Revista UIS Ingenierías
Página de la revista: https://revistas.uis.edu.co/index.php/revistauisingenierias

Sistema Web de Trazabilidad Para el Cultivo de Arroz Mediante Inteligencia Artificial

Traceability Web System for Rice Cultivation Using Artificial Intelligence

Ferley Medina Rojas[1], Maiker Stanley Canon Garces[2], Duban Estiben Cardozo Osorio[3], Diego Andres Castro Bahamón[4], Yeferson Mauricio Castro Vergel[5], Andres Felipe Diaz Gonzalez[6], Brayan José Liñan Garzón[7], Jhon Mauricio Mazabel Galindo[8], Nicolas Ortega Nader[9], Jhanier Johan Otavo Artunduaga[10], Daniel Ricardo Oyola Alvarez[11], Kevin Santiago Quimbaya Andrade[12], Juan Diego Rodriguez Lizcano[13]

[1]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: ferley.medina@usco.edu.co
[2]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20202193179@usco.edu.co
[3]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u2022209187@usco.edu.co
[4]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20192182708@usco.edu.co
[5]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20202191164@usco.edu.co
[6]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20202192835@usco.edu.co
[7]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20212200985@usco.edu.co
[8]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20201188628@usco.edu.co
[9]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20201186453@usco.edu.co
[10]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20201188078@usco.edu.co
[11]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20201186359@usco.edu.co
[12]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20202193482@usco.edu.co
[13]Proyecto integrador 4, Programa de Ingeniería de Software, Universidad Surcolombiana, Colombia. correo electrónico: u20212200546@usco.edu.co

Recibido: dd mes aaaa. Aceptado: dd mes aaaa. Versión final: dd mes aaaa

Resumen

ISSN impreso: 1657 - 4583. ISSN en línea: 2145 – 8456, **CC BY-ND 4.0**
Como citar: F. Medina-Rojas, M. Canon-Garces, D. Cardoso-Osorio, D. Castro-Bahamon, Y. Castro-Vergel, A. Diaz-Gonzales, B. Liñan-Garzon, J. Mazabel-Galindo, N. Ortega-Nader, J. Otavo-Artunduaga, D. Oyola-Alvarez, K. Quimbaya-Andrade, J. Rodriguez-Lizcano, "Desarrollo de Una Aplicación Para la Detección, Monitoreo y Trazabilidad De Plagas, Enfermedades, Malezas y Deficiencia de Nutrientes de los Cultivos De Arroz Mediante Inteligencia Artificial," *Rev. UIS Ing.*, vol. xx, no. x, pp. xx-xx, año. doi:

En el mundo actual la necesidad de utilizar la tecnología como herramienta para las tareas cotidianas se ha vuelto casi un requisito para optimizar recursos económicos y logísticos. Por ello uno de los sectores con la mayor demanda de implementación de tecnología es el agrícola debido a que cada día se ve obligado a producir mas con menos recursos de todo tipo, es aquí donde una aplicación de escritorio, móvil o web se debe implementar para ayudar al agricultor a llevar la trazabilidad de su cultivo y así mejorar su rendimiento. El presente proyecto se enfocó en desarrollar una aplicación cliente-servidor basada en la nube, con clientes web, que utiliza inteligencia artificial para identificar enfermedades como el añublo de la vaina, el añublo del arroz y la mancha parda; plagas como el gusano cogollero y la sogata; malezas problemáticas como el arroz rojo y la pata de gallina; y deficiencias nutricionales como nitrógeno, fósforo y potasio en cultivos de arroz. Esta aplicación permite a los agricultores realizar un monitoreo de sus cultivos, llevar un registro de costos, gastos, actividades realizadas, los problemas detectados para obtener informes de la trazabilidad de su cultivo. Se contempla el análisis detallado de requerimientos mediante Scrum y el estándar ISO/IEC/IEEE 29148:2018, el diseño del modelo de datos, la arquitectura de software y la interfaz de usuario utilizando modelos entidad-relación, modelo 4+1, wireframes y mockups. La codificación se realizará utilizando FastAPI para el backend y React para web. Se implementarán pruebas rigurosas y finalmente se desplegará en la nube utilizando RENDER.COM y Netlify que garantiza la disponibilidad para los usuarios, proporcionando a los agricultores de arroz una herramienta que les permite llevar el control de sus cultivos, identificar amenazas, tomar decisiones en base a datos históricos y mejorar su productividad.

Palabras clave: Arroz; Trazabilidad; Plagas; Enfermedades; Informe; Rentabilidad; Inteligencia Artificial; Aplicación; Requisitos; Diseño; Implementación.

Abstract

In today's world, the need to use technology as a tool for daily tasks has become almost a requirement to optimize economic and logistical resources. For this reason, one of the sectors with the greatest demand for technology implementation is agriculture because every day it is forced to produce more with fewer resources of all kinds, it is here where a desktop, mobile or web application must be implemented to help the farmer to keep track of his crop and thus improve his yield. This project focused on developing a cloud-based client-server application, with web clients, that uses artificial intelligence to identify diseases such as pod blight, rice blight and brown spot; pests such as fall armyworm and rope; problematic weeds such as red rice and crow's foot; and nutritional deficiencies such as nitrogen, phosphorus, and potassium in rice crops. This application allows farmers to monitor their crops, keep track of costs, expenses, activities carried out, problems detected to obtain traceability reports of their crop. The detailed analysis of requirements through Scrum and the ISO/IEC/IEEE 29148:2018 standard, the design of the data model, the software architecture and the user interface using entity-relationship models, 4+1 model, wireframes and mockups are contemplated. Coding will be done using FastAPI for the backend and React for web. Rigorous testing will be implemented and finally deployed in the cloud using RENDER.COM and Netlify that ensures availability for users, providing rice farmers with a tool that allows them to keep track of their crops, identify threats, make decisions based on historical data and improve their productivity.

Keywords: Rice; Traceability; Pests; Diseases; Report; Profitability; Artificial intelligence; Application; Requirements; Design; Implementation.

1. Introducción

El arroz es uno de los cultivos más importantes a nivel mundial [1]. Sin embargo, su producción se ve amenazada por diversas plagas, enfermedades, malezas y deficiencias nutricionales que pueden causar pérdidas significativas, además de la falta control de datos económicos históricos y actuales. Estos problemas se ven empeorados por el cambio climático y afectan la cantidad y calidad de la producción de arroz, además de la seguridad alimentaria de los consumidores [2].

El diagnóstico temprano y preciso de estos problemas es crucial para minimizar sus impactos y garantizar una producción sostenible. Tradicionalmente, el monitoreo y la identificación de estas afecciones se han realizado mediante inspecciones visuales y análisis de laboratorio, lo cual es lento, propenso a errores y requiere experiencia. En este contexto, el uso de tecnologías de inteligencia artificial ofrece una oportunidad para desarrollar soluciones precisas, eficientes y accesibles [3].

Se desarrollo la aplicación web, que utiliza inteligencia artificial para identificar enfermedades como el añublo de la vaina y el arroz, la mancha parda, plagas como el gusano cogollero y sogata, malezas problemáticas como el arroz rojo y la pata de gallina, así como deficiencias de nutrientes como el nitrógeno, fósforo y potasio en cultivos de arroz. Permitiendo permitirá a los agricultores realizar monitoreo de sus cultivos, llevar un registro de la trazabilidad de los problemas detectados y obtener recomendaciones para su manejo.

El proyecto contemplo un análisis detallado de requerimientos de los usuarios mediante la metodología ágil Scrum, la elaboración de un modelo de datos, el diseño de interfaz y arquitectónico del software, la programación de la aplicación utilizando tecnologías modernas, la implementación de pruebas rigurosas y finalmente el despliegue en plataformas accesibles que aseguren su disponibilidad para los usuarios.

2. Metodología

2.1. Fase 1. Análisis de requerimientos mediante la metodología Scrum y el estándar de requerimientos ISO/IEC/IEEE 29148:2018

2.1.1. Metodología Scrum

- Se elaboro una Estructura Desglosada del Trabajo (EDT) para definir los entregables del proyecto.
- Se elaboro un cronograma mediante un diagrama de Gantt que definirá el tiempo requerido para cada tarea.
- Se implemento la metodología Scrum para el análisis de requerimientos y la gestión del proyecto.
- Se elaboro un plan de gestión del alcance del proyecto.
- Se elaboro un matriz de gestión de riesgo del proyecto.
- Se elaboro un plan de gestión de cambios del proyecto.
- Se registro el número de Sprint a realizar, su duración y las tareas que se abordarán en cada uno.
- Se implemento un sistema de control de cumplimiento de actividades individuales.

2.1.2. Estado de las tareas

Para llevar un monitoreo del progreso de las tareas, se utilizó un tablero Scrum siguiendo las recomendaciones del estándar ISO/IEC/IEEE 26515. El tablero incluirá las siguientes columnas para representar los diferentes estados de las tareas:

- **Not started (No empezado):** Tareas que aún no han comenzado.
- **In progress (En progreso):** Tareas que están siendo trabajadas actualmente por el equipo.
- **Under review (En revisión):** Tareas que han sido completadas por el equipo y están esperando la revisión del Product Owner.
- **In test (En pruebas):** Tareas que están siendo probadas para asegurar que cumplen con los criterios de aceptación definidos.
- **In editing (En edición):** Tareas que requieren modificaciones o ajustes después de la revisión o las pruebas.
- **Completed (Completada):** Tareas que han pasado todas las etapas anteriores y se consideran finalizadas.

2.1.3. Recolección de información

Se realizaron entrevistas con las partes interesadas para identificar sus necesidades y expectativas. Las entrevistas se grabarán en audio previo consentimiento de los participantes. Además, se tomaron notas detalladas en actas durante las entrevistas, las cuales serán transcritas y analizadas posteriormente para identificar temas clave y patrones recurrentes.

2.1.4. Especificación de requerimientos

Se utilizo el estándar ISO/IEC/IEEE 29148:2018 para la especificación de los requerimientos del proyecto. Se definieron los requerimientos funcionales que detallan los comportamientos específicos del sistema y requerimientos no funcionales que se centran en aspectos de calidad como el rendimiento, seguridad, usabilidad, interoperabilidad y mantenibilidad. Se generará el entregable SRS basado en el estándar ISO/IEC/IEEE 29148.

2.1.5. Priorización de requerimientos

Se utilizo el método MoSCoW para priorizar los requerimientos según su relevancia para el éxito del proyecto [4]. La priorización la realizará el Product Owner, considerando factores como el valor para el negocio, dependencias, restricciones y recursos disponibles. Esta técnica se centra en cuatro categorías, representadas por el acrónimo MSCW que significa:

M (Must have): Debe tener. Estos son los requerimientos no negociables y fundamentales para el éxito del proyecto. Sin estas funcionalidades el sistema no puede cumplir con su propósito principal ni satisfacer las necesidades del cliente.
S (Should have): Debería tener. son los requerimientos importantes, pero no críticos. Son funcionalidades que agregan valor al producto y mejoran la experiencia del usuario.
C (Could have): Podría tener. Estos son deseables, pero no esenciales. Son funcionalidades que mejorarían el producto, pero no son necesarios para su funcionamiento básico.
W (Won't have this time): No tendrá esta vez. Estos requerimientos se reconocen como menos críticos y no se implementarán en el sprint actual, pero podrían considerarse para versiones futuras.

Identificador único: Cada requerimiento debe tener un identificador único para facilitar la referencia y seguimiento.

Descripción del requisito: Una breve descripción del requerimiento para entender su propósito y alcance.

Impacto en el negocio: Una evaluación del impacto que tendrá el requisito en el negocio, como la mejora de la experiencia del usuario, el aumento de la eficiencia o la reducción de costos.

Complejidad/Difucultad: Una estimación de la complejidad del requisito, lo que ayudará a determinar el esfuerzo y los recursos necesarios para implementarlo.

Dependencias: Identificación de cualquier dependencia con otros requisitos o componentes del sistema.

Estado: El estado actual del requerimiento, como " Not started", " In progress", " Under review", " In test", “In editing” o “Completed”.

2.1.6. Validación de requerimientos

La validación de requerimientos implica verificar si los requerimientos definidos son correctos, completos y consistentes [5]. Por ende, el siguiente formato se centra en la estructura y gestión de los requerimientos según los lineamientos establecidos en la norma ISO/IEC/IEEE 29148:2018.

2.1.7. Definición de historias de usuarios

Basándose en la información recabada durante las reuniones, se generaron historias de usuario siguiendo el estándar ISO/IEC/IEEE 26515:2018 y la Guía SBOK (Scrum Body of Knowledge). Cada historia de usuario describio una necesidad o deseo específico del usuario final.

2.1.8. Gestión de riesgos

Se identificaron y evaluaron los riesgos potenciales del proyecto utilizando una matriz de probabilidad e impacto **(Tabla 1).** Para cada riesgo, se asignó una prioridad y se desarrollaron estrategias de mitigación, como planes de contingencia, acciones preventivas o transferencia del riesgo. El registro de riesgos se revisó y actualizo regularmente durante las reuniones de retrospectiva del sprint. Se realizaron simulaciones y análisis de escenarios para evaluar la efectividad de las estrategias de mitigación propuestas.

- **No:** Número secuencial del riesgo identificado.
- **Riesgo:** Se identifica y nombra el riesgo. Puede ser algo como “Retraso en el cronograma”, “Costos mayores a los esperados”, “Alcance no aceptado por el cliente”, “Fallos de calidad inaceptables”, entre otros.
- **Descripción:** Se proporciona una descripción breve y clara del riesgo.
- **Causa:** Se identifica la causa específica del riesgo.
- **Probabilidad:** Probabilidad de ocurrencia del riesgo (1. Muy Baja, 2. Baja, 3. Media, 4. Alta, 5. Muy Alta).
- **Impacto:** Impacto potencial del riesgo en el proyecto (1. Muy Bajo, 2. Bajo, 4. Moderado, 8. Alto, 16. Muy Alto).
- **Prioridad:** Prioridad del riesgo, calculada a partir de la probabilidad y el impacto.
- **Estrategias de Mitigación:** Acciones planificadas para reducir la probabilidad o el impacto del riesgo.

Tabla 1. Matriz de probabilidad e impacto

		Impacto				
		Muy Bajo	Bajo	Moderado	Alto	Muy Alto
Probabilidad		1	2	4	8	16
Muy Alta	5	5	10	20	40	80
Alta	4	4	8	16	32	64
Media	3	3	6	12	24	48
Baja	2	2	4	8	16	32
Muy Baja	1	1	2	4	8	16

Prioridad	
32-80	Riesgo Extremo
16-24	Riesgo Alto
5-12	Riesgo Tolerable
1-4	Riesgo Aceptable

Fuentes: elaboración propia

2.2. Fase 2. Diseño visual y funcional de la aplicación

2.2.1. Arquitectura de software

2.2.1.1. Modelo 4+1

Se desarrollo un Documento de Arquitectura de Software siguiendo el modelo 4+1. Este proporciona un marco de referencia que utiliza 5 vistas para describir la arquitectura. Cada vista se centra en diferentes aspectos y stakeholders del sistema, abordando desde los requerimientos del usuario hasta los detalles de implementación y despliegue. Se utilizo la Herramienta StarUML para crear los diagramas UML que ilustraron y documentaron las diferentes vistas y elementos de la arquitectura. Esto promueve una visión integral del sistema y facilitó la comunicación entre miembros y stakeholders.

- **Vista lógica:** Indica las principales entidades del sistema como clases u objetos. Es la que percibe el usuario final y refleja la funcionalidad del sistema. Esta vista utiliza los siguientes diagramas UML: Diagrama de clases

- **Vista de proceso:** Explica los procesos del sistema y cómo se comunican en tiempo de ejecución. Tiene en cuenta requerimientos no funcionales, como el rendimiento y la disponibilidad. Esta vista utiliza los siguientes diagramas UML: Diagrama de Actividades**, Diagrama de Secuencia, Diagrama de Estado, Diagrama de Comunicaciones, Diagrama Global de Interacciones.**

- **Vista de desarrollo:** Ilustra el sistema de la perspectiva del programador y está enfocado en la organización real de los módulos de software en el entorno de desarrollo. Esta vista utiliza los siguientes diagramas UML: **Diagrama de Paquetes, Diagrama de Componentes.**

- **Vista física:** Describe el sistema desde el punto de vista de un ingeniero de sistemas. Está relacionada con la topología de componentes de software en la capa física, así como las conexiones físicas entre estos componentes. Esta vista utiliza los siguientes diagramas UML: **Diagrama de Despliegue.**

- **Vista de escenarios (Casos de uso):** Describen secuencias de interacciones entre objetos, y entre procesos. Se utilizan para identificar y validar el diseño de arquitectura. También sirven como punto de partida para pruebas de un prototipo de arquitectura. [6]. Esta vista utiliza los siguientes diagramas UML: **Diagramas de Casos de Uso.**

Se utilizo la herramienta StarUML para desarrollar todos los diagramas UML y exportar el código fuente del backend.

2.2.2. Diseño de interfaz y experiencia de usuario (UX/UI)

2.2.2.1. Diseño de interfaz (UI)

El diseño de la interfaz de usuario se centró en la creación de wireframes, mockups y prototipos. Este proceso se utilizó para definir la apariencia visual de la aplicación, su estructura y cómo los usuarios interactuarán con ella.

2.2.2.2. Diseño de experiencia de usuario (UX)

Se implemento la metodología Lean UX. Esta consta de los siguientes pasos: declarar suposiciones, crear declaraciones de hipótesis, diseño colaborativo, crear MVP’s y experimentos, recopilar feedbacks y aprendizaje.

2.2.3. Base de datos

Estos son los pasos que se van a seguir en el diseño y construcción de la base de datos: modelo conceptual, modelo lógico, modelo físico.

Se utilizo la herramienta StarUML para generar el código SQL que construirá la base de datos.

La base de datos se construyó sobre el sistema gestor de bases de datos PostgreSQL versión 15. Este proporciona ventajas como la integridad de datos, la eficiencia en el manejo de grandes volúmenes de datos y la posibilidad de realizar consultas complejas. Se ejecutaron las sentencias DDL generadas en el modelo físico para crear las tablas, restricciones e índices de la base de datos.
Se poblaron las tablas con datos de prueba. Se configuraron los permisos y roles de usuario necesarios. Se realizaron pruebas de rendimiento. Se utilizo el esquema de nomenclatura snake_case para nombrar tablas y columnas siguiendo las convenciones de PostgreSQL. Se realizaron copias de seguridad periódicas utilizando la herramienta pg_dump y se monitoreó el rendimiento utilizando la herramienta pgAdmin.

2.3. Fase 3. Codificación de desarrollo

Durante esta fase del proyecto, se abordó la codificación de los componentes de la aplicación, los cuales comprenden tanto el backend como el frontend web. Este proceso se registra en la **(Figura 1)**, donde se exponen las plataformas utilizadas, frameworks y tecnologías para el desarrollo de la aplicación.

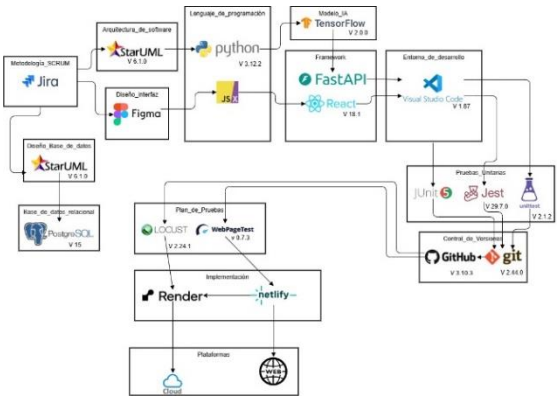

Figura 1. Arquitectura de desarrollo. Fuente: Elaboración propia.

También se estableció un flujo de desarrollo **(Figura 2)** que guio el desarrollo del proyecto garantizando que el equipo siguiera las etapas de manera continúa permitiendo el avance continuo.

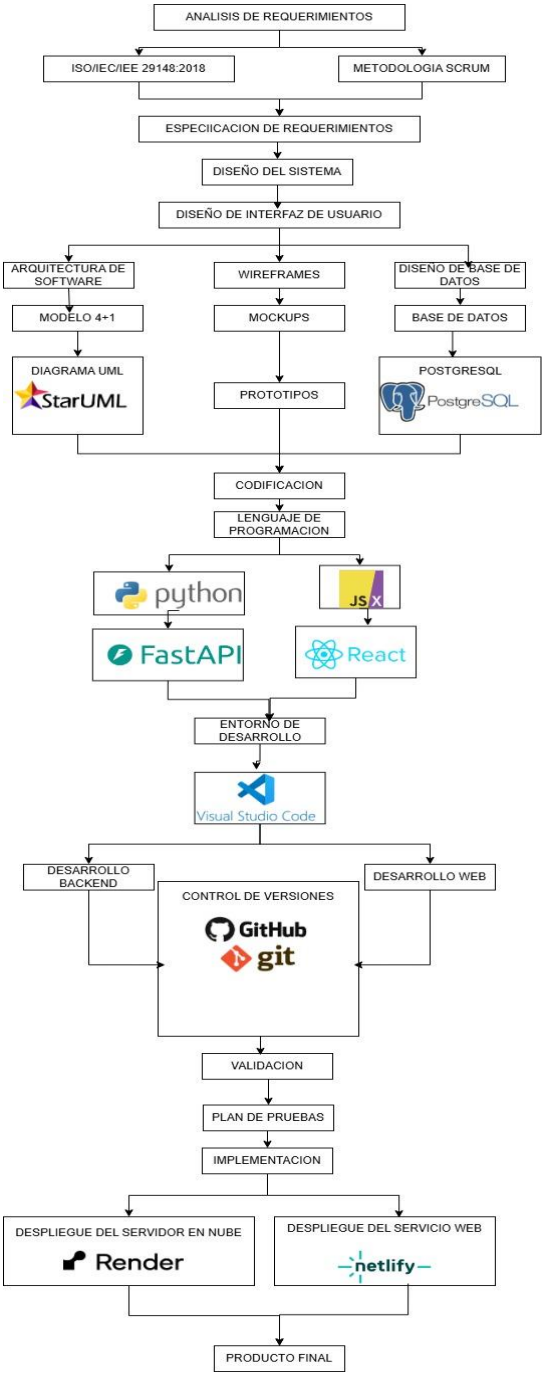

Figura 2. Flujo de desarrollo. Fuente: Elaboración propia.

2.3.1. Estándar de código

- Se utilizó nombres descriptivos y en idioma inglés para variables, funciones y constantes.

- Los nombres de variables y funciones siguieron la convención **"camelCase"**, comenzando con minúscula y utilizando mayúsculas para separar palabras.

2.3.2. Backend

El sistema tiene un backend en el lenguaje de programación Python siguiendo el paradigma de programación orientada a objetos. Python ofrece ventajas como amplio soporte de librerías de inteligencia artificial, sintaxis clara y programación multiparadigma. También permite crear entornos virtuales para cada proyecto, evitando conflictos de versiones de librerías entre distintos proyectos.
Para entrenar y ejecutar el modelo de visión se utilizó la librería TensorFlow. Esta ofrece una amplia biblioteca de funciones que facilitan el entrenamiento, validación y despliegue de modelos de inteligencia artificial en diversas plataformas.
Como Framework de desarrollo web se utilizó FastAPI. Este se caracteriza por generar documentación automática, ofrecer soporte para programación asíncrona y poseer una extensa y detallada documentación.

2.3.3. Desarrollo Web

Se utilizo React como framework de desarrollo web debido a sus ventajas, como el uso de JSX para crear componentes reutilizables y modulares, el Virtual DOM que optimiza el rendimiento al actualizar solo las partes necesarias de la interfaz, la reconciliación que minimiza las manipulaciones en el DOM real y su flexibilidad para integrarse con otras bibliotecas mediante NPM [7]. Se empleo la versión más reciente y estable de React, junto con la biblioteca de React Router para el enrutamiento de la aplicación.

2.4. Fase 4. Validación la funcionalidad de la aplicación

El equipo de QA llevo a cabo un conjunto exhaustivo de pruebas al finalizar cada sprint de codificación, antes de integrar los cambios al proyecto principal. Estas pruebas, están basadas en la norma ISO 29119 y los estándares de calidad proporcionados por la norma ISO 25010. Antes de ejecutar el plan de pruebas del proyecto, fue necesario establecer una base sólida que garantizo la efectividad de las pruebas. Esto implico revisar y documentar claramente los requerimientos, verificar que el desarrollo de los módulos a probar esté completo y funcione según lo esperado, preparar un entorno de pruebas representativo del entorno de producción, disponer de datos de prueba representativos y asegurar el acceso a todos los sistemas y recursos necesarios para llevar a cabo las pruebas.

2.4.1. Tipos de pruebas:

Se estableció un conjunto de pruebas a evaluar y otro conjunto de pruebas que no serán contempladas en esta etapa como lo muestra la **(Tabla 2)**.

Tabla 2. Pruebas cubiertas por el plan de pruebas

Cubierto	No Cubierto
Pruebas de Caja Negra	Pruebas de Caja Blanca
Pruebas de Integración	Pruebas Unitarias
Pruebas de Sistema	Pruebas de Confiabilidad y Estabilidad
Pruebas de Usabilidad	Pruebas de Confirmación
Pruebas de Seguridad	Pruebas de Regresión
Pruebas de Carga	Navegadores Legacy
Pruebas de Aceptación	Pruebas de Humo
	Criterios de Entrada y Salida
	Calidad de los Datos

Fuente: elaboración propia.

2.5. Fase 5. Implementación de la aplicación

En esta etapa se definieron las configuraciones y protocolos necesarias para desplegar el software de Trazabilidad de los Cultivos De Arroz Mediante Inteligencia

2.5.1. Despliegue del servidor en la Nube

Se utilizo RENDER.COM como proveedor de servicios en la nube debido a su escalabilidad, que le permite adaptar los recursos a las necesidades de la aplicación. Su amplia gama de servicios, como alojamiento de aplicaciones, modelos de inteligencia artificial, almacenamiento y análisis de datos y su enfoque en la seguridad ofreciendo una amplia gama de herramientas para proteger los datos de aplicaciones y usuarios.

Para el hosting se desplegó en la región de Oregón y se configuraron los comandos para construir y ejecutar la aplicación, con un tipo de instancia con un paquete inicial de 7 dolares mensuales que proporciona 512 mb de ram y un procesador de 0.5 Ghz.

Para la base de datos se desplegó en la región de Oregón, con una instancia que proporciona 256 mb de ram, 0.1 ghz de cpu y una giga de almacenamiento.

2.5.2. Despliegue del servicio web

Se implemento el cliente web con Netlify debido a su integración con GitHub que permite actualizaciones automáticas cuando se realizan cambios en el repositorio, sus prácticas de CI/CD que ejecuta pruebas automáticas con cada actualización, y su infraestructura optimizada, que garantiza un alto rendimiento y disponibilidad del servicio web. Se habilitará HTTPS utilizando el certificado SSL/TLS proporcionado por Netlify.

3. Resultados

3.1. Estructura de Desglose del Trabajo EDT

Se estableció las etapas del proyecto y el contenido de cada una como se muestra en la **(Figura 3)**, ya que es una estructura secuencial no permite el inicio de la siguiente etapa sin la culminación de la actual.

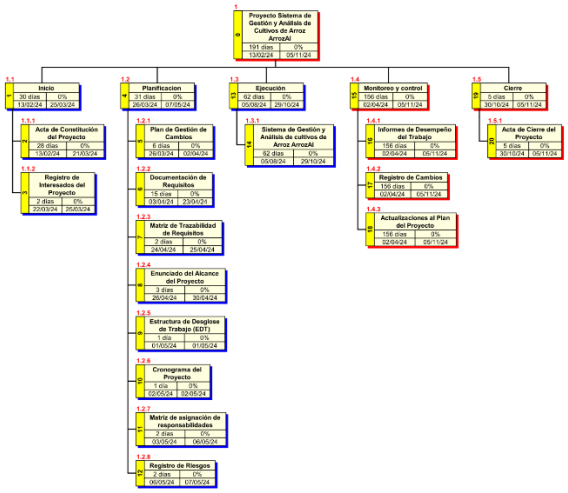

Figura 3. Estructura de desglose del trabajo. Fuente: Elaboración propia

3.2. Especificación de requerimientos (SRS)

https://drive.google.com/file/d/1jrtkXLG1VKZ5E6OiPzvID2KM0RX5XJwT/view?usp=sharing

3.3. Planificación de Sprints

Sprint 1: Implementación de la Base de datos, gestión de usuarios, roles y permisos (2024-08-05 al 2024-08-29)

- **Objetivo:** Establecer la base de datos del sistema, la gestión de usuarios, roles y permisos.
- **Requerimientos:**
  - RBD1: Implementar base de datos relacional.
  - RBD2: Garantizar la integridad referencial entre las tablas relacionadas.
  - RF62: Crear rol de acceso.
  - RF63: Consultar información de rol de acceso.
  - RF64: Modificar rol de acceso.
  - RF65: Eliminar rol de acceso.
  - RF66: Crear permiso.
  - RF67: Consultar información de permiso.
  - RF68: Modificar permiso.
  - RF69: Eliminar permiso.
  - RF82: Asignar permisos predeterminados a Agricultores y Administradores de fincas.
  - RF83: Asignar permisos predeterminados a Agrónomos.
  - RF84: Asignar permisos predeterminados a Personal de campo.
  - RF85: Iniciar sesión.
  - RF86: Recuperar contraseña.
  - RF87: Cambiar contraseña.
  - RF88: Crear cuenta de usuario.
  - RF89: Consultar información de cuenta de usuario.
  - RF90: Modificar información de cuenta de usuario.
  - RF91: Eliminar cuenta de usuario.
  - RF93: Cambiar contraseña al iniciar sesión por primera vez.
- **Artefactos entregables:**

- Modelo de datos (Diagrama Entidad-Relación) de la base de datos relacional **(Figura 4).** **https://drive.google.com/file/d/1l1TDRC9ZPX3WEfRJ3HTbTTxd4YB_t5du/view?usp=drive_link**, en link anterior se encuentra el archivo correspondiente al diagrama.

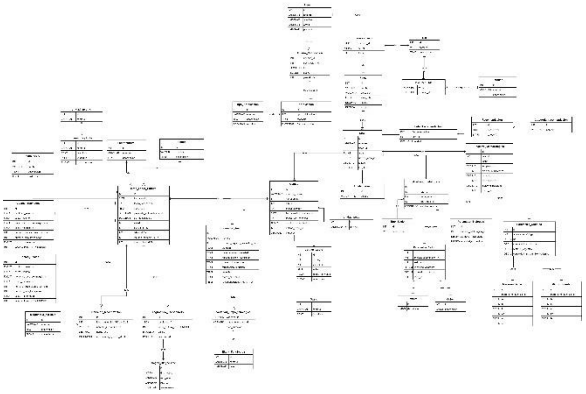

Figura 4. Diagrama entidad-relación. Fuente: Elaboración propia.

El módulo de gestión de usuarios permite al administrador crear, eliminar, actualizar usuarios **(Figura 5)**.

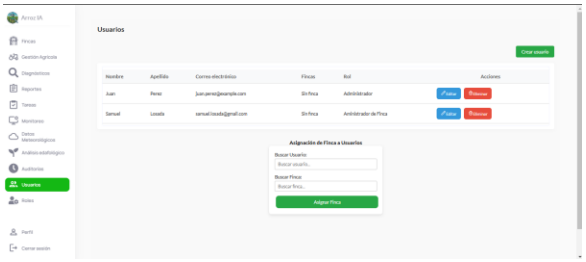

Figura 5. Modulo gestión de usuario. Fuente: Aplicación.

Al asignar roles el administrador también puede gestionar esos roles adicionando permisos o quitándolos **(Figura 6)**.

Figura 6. Gestión de permisos. Fuente: Aplicación.

Sprint 2: Implementación de los módulos de Gestión de Fincas, Lotes y cultivos (2024-08-30 al 2024-09-12)

- Objetivo: Implementar la gestión básica de fincas, lotes, cultivos y variedades de arroz.
- Requerimientos:
  - RF1Crear Finca.
  - RF2 Consultar información de fincas.
  - RF3 Modificar información de fincas.
  - RF4 Eliminar fincas.
  - RF5 Crear nuevos lotes.
  - RF6 Consultar información de lotes.
  - RF7 Modificar información de lotes.
  - RF8 Eliminar lotes.
  - RF9 Crear nuevos cultivos de arroz.
  - RF10 Consultar información de cultivos de arroz.
  - RF11 Modificar información de cultivos de arroz.
  - RF12 Eliminar cultivos de arroz.
  - RF13 Configurar unidad de área del lote.
  - RF74 Crear variedades de arroz
  - RF75Consultar información de variedad de arroz
  - RF76 Modificar información de variedad de arroz
  - RF77 Eliminar variedad de arroz
  - RF94 Consultar rol y finca de usuario
  - RF95 Consultar el lote de la finca
  - RF96 Consultar el cultivo del lote

Artefactos entregables:

o Diagrama de clases UML para el modelo de fincas, lotes, cultivos y variedades de arroz (Vista lógica) **(Figura 7)**.

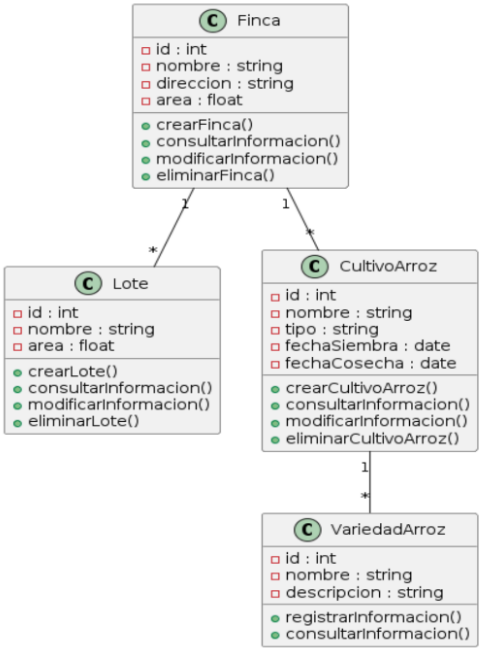

Figura 7. Diagrama de clases UML para el modelo de fincas, lotes, cultivos y variedades de arroz. Fuente: elaboración propia.

En el modulo de fincas se desarrollo un CRUD que permite al usuario crear, editar, eliminar, ver detalles de la finca **(Figura 8)**.

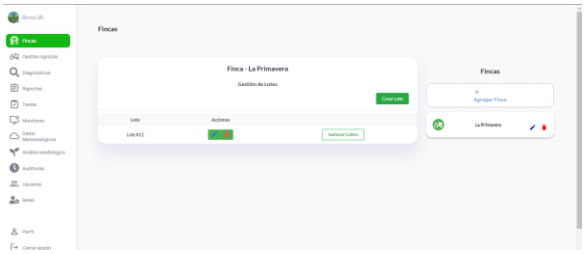

Figura 8. Módulo de fincas. Fuente: Aplicación.

Sprint 3: Desarrollo de los módulos de Planificación de cronograma del cultivo, tareas, labores culturales y mecanización (2024-09-13 al 2024-10-03)

- **Objetivo:** Habilitar la planificación de cultivos con cronogramas, registro de insumos agrícolas necesarios, labores culturales, operaciones de mecanización agrícola y asignación de tareas.
- **Requerimientos:**
  - RF14: Generar cronograma estimado del ciclo vegetativo del cultivo.
  - RF15: Generar plan de labores cultura les para el cultivo.
  - RF45: Asignar responsabilidades de ta reas del cronograma.
  - RF46: Consultar tareas asignadas.
  - RF47: Modificar tareas asignadas.
  - RF48: Eliminar tareas asignadas.
  - RF49: Marcar tareas planeadas como completadas.
  - RF76: Registrar información de insumos agrícolas
  - RF77: Consultar información de insumos agrícolas registrados
  - RF78: Modificar información de insumos agrícolas
  - RF79: Eliminar insumos agrícolas del sistema
  - RF17: Generar plan de monitoreos fitosanitarios del cultivo.
  - RF25: Crear manualmente nuevas labores y agregarlas al cronograma.
  - RF26: Consultar las labores culturales del cronograma.
  - RF27: Modificar información de las la bores culturales.
  - RF28: Eliminar labores culturales del cronograma.
  - RF29: Crear operaciones de mecaniza ción asignadas a una labor cultural.
  - RF30: Consultar información de opera ciones de mecanización.
  - RF31: Modificar datos de una operaci ón de mecanización.
  - RF32: Eliminar operaciones de mecan ización.

- **Artefactos entregables:**
    - Diagrama de clases UML para el modelo de cronogramas, insumos agrícolas, labores culturales, operaciones de mecanización y tareas. (Vista lógica) **(Figura 9)**.

Figura 9. Diagrama de clases UML para el modelo de cronogramas, insumos agrícolas, labores culturales, operaciones de mecanización y tareas. Fuente: elaboración propia.

Al desarrollar el modulo de cronograma del cultivo **(Figura 10),** se estableció la finalidad de guiar al agricultor en la organización de las labores culturales del cultivo para llevar un control de ejecución de estas de manera efectiva.

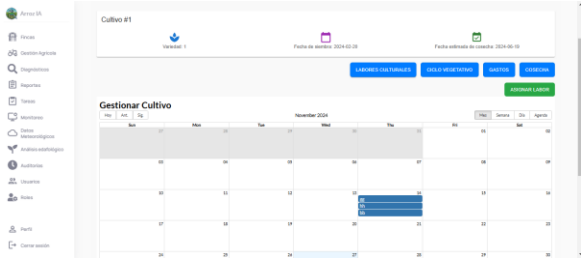

Figura 10. Módulo de cronograma del cultivo mostrando las fechas estimadas de inicio y fin de cada etapa fenológica del arroz. Fuente: Aplicación.

Sprint 4: Desarrollo de los módulos de Análisis edafológicos, monitoreos fitosanitarios y consultas meteorológicas (2024-10-04 al 2024-10-17)

- **Objetivo:** Implementar análisis edafológicos, diagnósticos fitosanitarios con IA y consultas de datos meteorológicos.
- **Requerimientos:**
    - RF20: Realizar análisis edafológico d el suelo del lote.
    - RF22: Consultar análisis edafológicos realizados en cada lote.
    - RF23: Modificar parámetros de un an álisis edafológico.
    - RF24: Eliminar análisis edafológicos r ealizados en un lote.
    - RF90: Solicitar permisos para acceder a la cámara y sistema de archivos.
    - RF33: Realizar diagnóstico fitosanitario online mediante IA.
    - RF34: Realizar diagnóstico fitosanitar io offline mediante IA.
    - RF36: Consultar resultados históricos de diagnósticos de un cultivo.
    - RF37: Reevaluación del diagnóstico c on nuevas imágenes.
    - RF38: Eliminar diagnósticos realizado s en un cultivo.
    - RF39: Registrar consulta de datos met eorológicos manual.
    - RF40: Obtener datos meteorológicos por API externa.
    - RF42: Consultar historial de datos meteorológicos registrados.
    - RF43: Modificar datos meteorológicos erróneos.
    - RF44: Eliminar consultas de datos me teorológicos.
- **Artefactos entregables:**
    - Diagrama de clases UML para el modelo de análisis edafológicos, diagnósticos fitosanitarios y consultas meteorológicas. (Vista lógica) **(Figura 11)**.

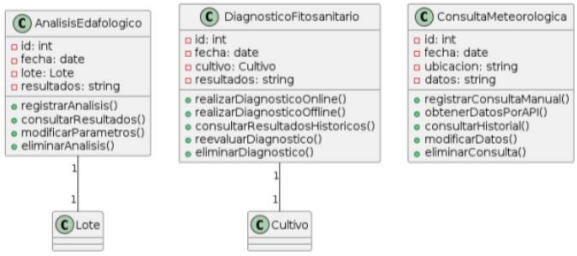

Figura 11. Diagrama de clases UML para el modelo de análisis edafológicos, diagnósticos fitosanitarios y consultas meteorológicas. Fuente: elaboración propia.

Al iniciar a cultivar el agricultor debe registrar el estado del terreno donde va a cultivar para ello se utiliza el Análisis edafológico **(Figura12)**, donde le da una guía del estado de la tierra y la forma como acondicionarle para cultivar.

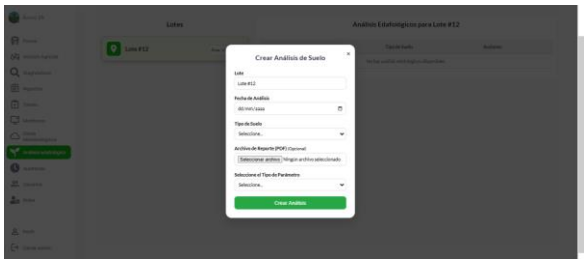

Figura 12. Módulo de análisis edafológico, Fuente: Aplicación.

Para llevar una trazabilidad eficiente es necesario realizar diagnósticos para la identificación de enfermedades presentes en el cultivo **(Figura 13)**, se sube una imagen de la planta y mediante inteligencia artificial se diagnostica si se encuentra enfermedad.

Figura 13. Módulo de diagnóstico, Fuente: Aplicación.

Sprint 5: Creación de los módulos de Presupuesto, informes y notificaciones (2024-10-18 al 2024-11-25)

- **Objetivo:** Implementar funcionalidades de generación de presupuesto estimado, presupuesto actualizado, informes finales, gráficos, notificaciones y alertas.

- **Requerimientos:**
  - RF19: Calcular presupuesto inicial estimado para la realización del cultivo.
  - RF50: Actualizar presupuesto estimado del cultivo con costos reales.
  - RF51: Generar informe final de resultados.
  - RF52: Consultar informes finales generados para cada cultivo
  - RF53: Eliminar informes finales generados.
  - RF54: Generación de gráficos y visualizaciones interactivas.
  - RF55: Generar reportes gráficos comparativos.
  - RF68: Configurar notificaciones y alertas
  - RF69: Llevar registro de notificaciones y alertas
  - RF70: Consultar el registro histórico de notificaciones y alertas
  - RF71: Almacenar notificaciones y alertas por 90 días.

- **Artefactos entregables:**

Diagrama de clases UML para el modelo de presupuestos, informes finales, gráficos y notificaciones **(Figura 14)**.

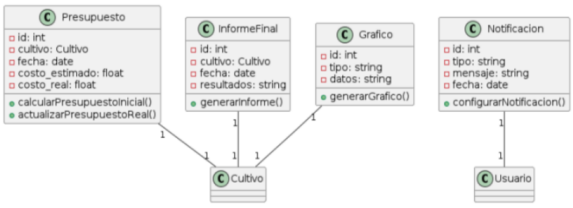

Figura 14. Diagrama de clases UML para el modelo de presupuestos, informes finales, gráficos y notificaciones. Fuente: elaboración propia.

El llevar la trazabilidad del cultivo es necesario para poder generar reportes **(Figura 15),** y así poder tomar decisiones futuras por ello al finalizar el ejercicio de cultivar el agricultor tendrá reportes Presupuesto

estimado, rentabilidad, gastos, costos, enfermedades presentes en el cultivo, malezas, deficiencias de nutrientes.

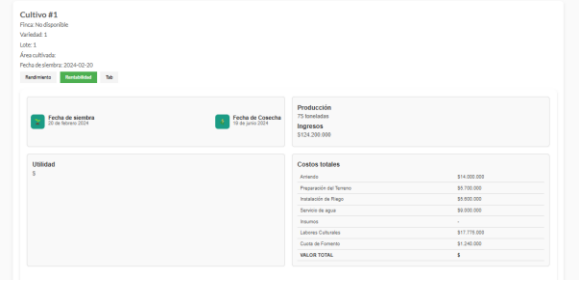

Figura 15. Modulo reportes sección de reportes. Fuente: Aplicación.

4.4 Entregable final

https://arroz-ia.netlify.app/

Usuario: juan.perez@example.com

Contraseña: Usco2024.

4.5 Indicadores de rendimiento

Adicional a la metodología SCRUM, se implemento un sistema de verificación de cumplimiento individual (Figura 14), donde en cada incumplimiento se realizaba un llamado de atención (memorando), y si llegase a cumplir tres quedaba por fuera del equipo de trabajo en la imagen se puede observar que el 50% del equipo tiene un memorando y que el 33% tiene dos memorandos quedando solo el 17% del equipo es decir dos integrantes sin ningún llamado de atención, identificando como una herramienta de motivación efectiva para el desarrollo de las actividades.

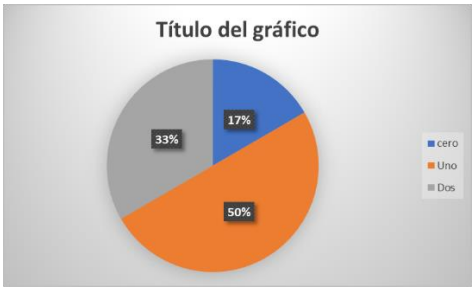

Figura 14. porcentaje de llamados de atención del equipo de trabajo. Fuente: Elaboración propia.

4. Conclusiones

El análisis de requerimientos del sistema ArrozIA, utilizando la metodología Scrum y el estándar ISO/IEC/IEEE 29148:2018, permitió identificar y definir las necesidades de los usuarios de manera detallada y estructurada. Esto creó una base sólida para el desarrollo de la aplicación, asegurando que todas las funcionalidades necesarias fueran adecuadamente consideradas desde el inicio.

El diseño de la arquitectura de software empleando el modelo 4+1, junto con la selección de tecnologías como FastAPI y React, proporcionó un marco sólido y flexible para la implementación de la aplicación. Estas elecciones consideraron las necesidades de escalabilidad, seguridad y rendimiento, permitiendo una robusta integración de todos los componentes del sistema.

El diseño de la interfaz de usuario, utilizando wireframes, mockups y metodologías como Lean UX, buscó garantizar una experiencia de usuario intuitiva y eficiente. Este enfoque facilitó la interacción con las funcionalidades del sistema, asegurando que los usuarios pudieran utilizar la aplicación de manera sencilla y efectiva.

La integración con servicios externos, como la API de OpenWeather, fue planificada estratégicamente, teniendo en cuenta la disponibilidad, seguridad y calidad de los datos. Esta integración es crucial para el buen funcionamiento del sistema, proporcionando información precisa y oportuna para los usuarios.

La identificación y análisis de los riesgos potenciales del proyecto, junto con la definición de estrategias de mitigación, proporcionaron un plan de acción para abordar los desafíos y asegurar el éxito del desarrollo de la aplicación. La gestión de riesgos es esencial para anticipar y manejar posibles problemas que puedan surgir durante el ciclo de vida del proyecto.

El desarrollo de la aplicación permitió la automatización de la planeación y ejecución del cultivo arroz.

En la ejecución de un proyecto la reestructuración de la etapa anterior (planeación), es necesaria para compensar los imprevistos como lo es cambio de personal, tecnología, presupuesto y así poder tener un producto de buena calidad.

El desarrollo de aplicaciones web debe cumplir los requerimientos del mundo digital, exigiendo así que el software sea adaptable y escalable, por lo cual la mayoría de las aplicaciones desarrolladas deben el mínimo de las

tecnologías para que se puedan desempeñar de manera correcta con las necesidades del usuario.

La implementación de inteligencia artificial a las labores agrícolas permite una detección temprana de enfermedades en el cultivo, facilitando su diagnóstico y disminuyendo el tiempo de reacción para tratar dichas enfermedades.

5. Recomendaciones

Definición de la idea de negocio: es necesario tener definida y acotada la idea de negocio para evitar cambios en medio del desarrollo del proyecto por nuevas funcionalidades o funcionalidades que no se tuvieron en cuenta lo cual aumenta la carga de trabajo.

Para un correcto funcionamiento del aplicativo web se recomienda utilizarlo en el navegador web Google Chrome.

Refuerzo de la comprensión del dominio: Es crucial profundizar la comprensión del dominio agrícola para asegurar la precisión de las reglas de negocio, la información de variedades de arroz y la validez de las recomendaciones.

Mejorar la planeación: la planeación de un proyecto es la parte más importante de un proyecto ya que si se hace correctamente se estaría mitigando los cambios fortuitos que desencadenan en cambios estructurales o extensión del tiempo.

Evaluación del equipo de trabajo: una acertada evaluación y caracterización del equipo de trabajo minimiza el riesgo de retraso del proyecto.

Métricas de cumplimiento: aunque se implementen metodologías agiles como Scrum, es necesario implementar métricas que garanticen el cumplimiento de todo el equipo del trabajo para evitar retrasos en la producción.

Definición de métricas de éxito: Se deben establecer métricas claras para evaluar el éxito del proyecto, no solo en términos de funcionalidades implementadas, sino también en relación con el impacto en la productividad y rentabilidad de los usuarios.

6. Conflicto de interés

Los autores declaran que no tienen ningún conflicto de interés

7. Referencias

[1] S. Sen, R. Chakraborty, y P. Kalita, "Rice - not just a staple food: A comprehensive review on its phytochemicals and therapeutic potential," Trends in Food Science & Technology, vol. 97, pp. 265–285, 2020.

[2] D. Argelia y R. Córdoba Cantero, "Capacidad de resiliencia de pequeños productores de arroz en Colombia y sus implicaciones para la soberanía alimentaria en el contexto pandémico," Redes, vol. 27, 2022, doi: 10.17058/redes.v27i1.17946.

[3] K. Petchiammal, B. Kiruba, M. Murugan y P. Arjunan, "Paddy doctor: A visual image dataset for automated paddy disease classification and benchmarking," en CODS-COMAD 2023: 6th Joint International Conference on Data Science & Management of Data (10th ACM IKDD CODS and 28th COMAD), 2023, pp. 203–207, doi: 10.1145/3570991.3570994.

[4] A. Hudaib, R. Masadeh, M. H. Qasem y A. Alzaqebah, "Requirements Prioritization Techniques Comparison," Modern Applied Science, vol. 12, no. 2, p. 62, 2018, doi: 10.5539/mas.v12n2p62.

[5] R. Lai y N. Ali, "A method of requirements change management for global software development," Information and Software Technology, vol. 70, pp. 49–67, 2016, doi: 10.1016/j.infsof.2015.09.005.

[6] I. Sommerville, "Software Engineering," Pearson, 2018.

[7] Meta Platforms, Inc., "React - Una biblioteca de JavaScript para construir interfaces de usuario," Es.legacy.reactjs.org, 2024. [Online]. Available: https://es.legacy.reactjs.org/