|  |
| --- |
|  |

**Protocolo y Metodología de Desarrollo**

Proyecto PIA: OPTIMIZACIÓN DE CADENAS DE SUMINISTRO AGRÍCOLA CON MACHIN LERNING Y DEEP LESNING

Contenido

[Introducción 4](#_Toc182656393)

[Objetivos del Protocolo 5](#_Toc182656394)

[Metodología Utilizada 6](#_Toc182656395)

[Enfoque Ágil 6](#_Toc182656396)

[Desglose de Funcionalidades 6](#_Toc182656397)

[Login y Seguridad 6](#_Toc182656398)

[Dashboard 7](#_Toc182656399)

[Predicciones de Exportaciones 7](#_Toc182656400)

[Pantalla de Optimización 7](#_Toc182656401)

[Análisis de Ubicación y Registro 8](#_Toc182656402)

[Pruebas y Aseguramiento de Calidad 8](#_Toc182656403)

[Indicadores de Éxito Específicos 9](#_Toc182656404)

[Herramientas Tecnológicas 10](#_Toc182656405)

[Backend 10](#_Toc182656406)

[Frontend 10](#_Toc182656407)

[Protocolo de Desarrollo y Pruebas 10](#_Toc182656408)

[Desarrollo 10](#_Toc182656409)

[Pruebas 10](#_Toc182656410)

[Indicadores de Éxito del Proyecto 11](#_Toc182656411)

[Funcionalidad 11](#_Toc182656412)

[Rendimiento 11](#_Toc182656413)

[Usabilidad 11](#_Toc182656414)

[Escalabilidad 11](#_Toc182656415)

[Cronograma de Desarrollo 12](#_Toc182656416)

[Sprint 1-2 12](#_Toc182656417)

[Sprint 3-4 12](#_Toc182656418)

[Sprint 5-6 12](#_Toc182656419)

[Sprint 7-8 12](#_Toc182656420)

[Documentación y Soporte 13](#_Toc182656421)

# **Introducción**

El presente documento detalla la metodología y el protocolo para el desarrollo del proyecto, utilizando principios de ingeniería de software y herramientas de inteligencia artificial para optimizar las cadenas de suministro agrícola. El enfoque principal se centra en desarrollar un sistema web robusto que permita gestionar, analizar y optimizar datos relacionados con cultivos y su logística.

# **Objetivos del Protocolo**

Garantizar la alineación con los objetivos del proyecto.

Estandarizar procesos y prácticas de desarrollo.

Implementar técnicas ágiles que permitan la iteración continua y entrega temprana de valor.

Desarrollar módulos funcionales que incluyan Login, Dashboard, Predicciones de Exportaciones, Pantalla de Optimización, y análisis de ubicación.

# **Metodología Utilizada**

**Enfoque Ágil: Scrum**

Se opta por el marco Scrum debido a su flexibilidad y capacidad para manejar proyectos con alta incertidumbre y cambios constantes.

**Duración del Sprint: 2 semanas.**

**Roles y Responsabilidades:**

**Product Owner:** Define los requerimientos del sistema, prioriza las tareas y valida entregables.

**Scrum Master:** Facilita las reuniones y elimina impedimentos.

**Equipo de Desarrollo:** Desarrolla, prueba e integra el sistema.

**Artefactos Scrum:**

**Product Backlog:** Listado de funcionalidades, como predicciones, registro y análisis de ubicación.

**Sprint Backlog:** Conjunto de tareas seleccionadas para el sprint actual.

**Incremento:** Funcionalidades desarrolladas y probadas en cada sprint.

**Ceremonias Scrum:**

**Daily Standup:** Reunión diaria para sincronizar actividades.

**Sprint Planning:** Planificación de las tareas del sprint.

**Sprint Review:** Demostración del incremento al Product Owner.

**Retrospective:** Identificación de áreas de mejora.

# **Desglose de Funcionalidades**

## **Login y Seguridad**

**Objetivo:**  
Garantizar acceso seguro y controlado basado en roles de usuario.

**Características:**

**Autenticación:**

Uso de **OAuth 2.0** y JSON Web Tokens (JWT) para autenticación basada en estándares.

Opcional: Integración con servicios de terceros (Google, Facebook).

**Roles y Permisos:**

Roles: Administrador, Agricultor, Distribuidor, Minorista.

Restricciones basadas en permisos, por ejemplo: los agricultores solo pueden registrar cultivos, mientras que los administradores acceden al análisis completo.

**Implementación Técnica:**

Framework de backend: Django con Django-Security para autenticación.

Cifrado de contraseñas con bcrypt.

Base de datos para usuarios: PostgreSQL con tablas para roles y permisos.

**Pruebas:**

**Pruebas de Penetración:** Simulación de ataques como inyección SQL o XSS.

**Pruebas Funcionales:** Validación de flujos de inicio de sesión, registro y recuperación de contraseñas.

## **Dashboard**

**Objetivo:**  
Proveer un panel centralizado e intuitivo para visualizar métricas clave.

**Elementos Principales:**

Gráficos interactivos: Barras, líneas y mapas de calor.

Indicadores clave de desempeño (KPIs):

Predicciones de exportaciones.

Optimización de rutas.

Análisis de calidad del producto.

**Diseño:**

Tecnología: React.js para el frontend, con bibliotecas de gráficos como Chart.js y D3.js.

Responsividad: Diseño adaptable a dispositivos móviles.

**Pruebas:**

**Pruebas de Usabilidad:** Evaluación con usuarios finales.

**Pruebas de Rendimiento:** Carga de gráficos con grandes volúmenes de datos.

## **Predicciones de Exportaciones**

**Objetivo:**  
Predecir demandas futuras para optimizar volúmenes de producción y logística.

**Características:**

Modelos basados en Machine Learning:

LSTM para series temporales.

Random Forest para predicción de categorías.

Datos de entrada:

Historial de ventas.

Variables climáticas (Biblioteca Python-Weather).

**Implementación Técnica:**

Bibliotecas: TensorFlow/Keros para Deep Learning, scikit-learn y AgML para ML.

Entrenamiento del modelo:

Datos preprocesados y normalizados.

Validación cruzada para evitar overfitting.

**Pruebas:**

**Pruebas de Precisión:** Métricas como MAE, RMSE.

**Pruebas de Robustez:** Validación con datos históricos y nuevos.

## **Pantalla de Optimización**

**Objetivo:**  
Ofrecer soluciones óptimas para reducir costos y tiempos en la logística de distribución.

**Características:**

Algoritmos de optimización:

Algoritmo de Dijkstra para rutas más cortas.

Algoritmos genéticos para combinaciones complejas.

Factores de optimización:

Distancia, costos, tiempo, emisiones de carbono.

**Implementación Técnica:**

Integración con APIs de mapas como Google Maps o OpenStreetMap para geolocalización.

Backend con Django para manejar cálculos en tiempo real.

**Pruebas:**

**Pruebas de Escalabilidad:** Verificar tiempos de respuesta con un gran número de rutas y puntos de entrega.

**Pruebas Funcionales:** Comparación de resultados con métodos manuales.

## **Análisis de Ubicación y Registro**

**Objetivo:**  
Determinar ubicaciones óptimas para cultivos y centros de distribución.

**Características:**

Herramientas GIS:

Procesamiento de datos espaciales con GeoPandas y QGIS.

Datos considerados:

Condiciones climáticas, tipo de suelo, proximidad a mercados.

**Implementación Técnica:**

Bases de datos espaciales: PostgreSQL con extensión PostGIS.

Modelos de análisis:

Regresión logística para probabilidad de éxito en ubicaciones.

Clustering (K-Means) para agrupación de áreas similares.

**Pruebas:**

**Pruebas de Precisión:** Validación de análisis con datos históricos.

**Pruebas de Visualización:** Mapas interactivos para validar ubicaciones sugeridas.

## **Pruebas y Aseguramiento de Calidad**

**Objetivo:**  
Garantizar que el sistema cumple con los requerimientos y es fiable.

**Pruebas Implementadas:**

**Unitarias:** Cada módulo, como Login o Predicciones, probado individualmente.

**De Integración:** Validación del flujo entre Dashboard, Optimización y Registro.

**De Rendimiento:** Uso de herramientas como Apache JMeter para pruebas de carga.

**De Seguridad:** Análisis de vulnerabilidades con herramientas como OWASP ZAP.

## **Indicadores de Éxito Específicos**

**Login y Seguridad:** Tiempo de respuesta < 2s; > 95% de intentos exitosos.

**Dashboard:** Representación precisa de los datos en tiempo real.

**Predicciones:** Error < 10% en pruebas de series históricas.

**Optimización:** Reducción de costos logísticos > 15%.

**Análisis de Ubicación:** Propuestas acertadas > 80% según datos históricos.

# **Herramientas Tecnológicas**

**Backend:** Django (Python), con integraciones de bibliotecas como TensorFlow y scikit-learn.

Frontend**:** HTML, CSS, JavaScript (React.js).

**Base de Datos:** PostgreSQL para datos estructurados; MongoDB para datos no estructurados.

**APIs Externas:** OpenWeatherMap para clima; FAO para datos agrícolas.

# **Protocolo de Desarrollo y Pruebas**

## **Desarrollo**

Uso de control de versiones con Git y GitHub.

Integración continua y despliegue continuo (CI/CD) con herramientas como Jenkins.

Codificación basada en estándares de calidad como PEP-8 para Python.

## **Pruebas**

Pruebas Unitarias: Validación de módulos individuales.

Pruebas de Integración: Verificación de interacción entre módulos (e.g., Predicciones y Dashboard).

Pruebas Funcionales: Validación de funcionalidades frente a los requerimientos.

Pruebas de Usuario: Identificación de problemas de usabilidad y satisfacción del usuario final.

# **Indicadores de Éxito del Proyecto**

Funcionalidad**:** Implementación completa de módulos según requerimientos.

Rendimiento**:** Respuesta en tiempo real para consultas de datos y predicciones.

Usabilidad**:** Alta satisfacción en pruebas de usuario (>85%).

Escalabilidad**:** Capacidad de soportar aumentos en datos y usuarios.

# **Cronograma de Desarrollo**

Un cronograma de 6 meses con entregables claves por sprint:

|  |  |  |
| --- | --- | --- |
| **FASE** | **DURACIÓN** | **HITO-CLAVE** |
| Inicio-Planeación | 2 semanas | Documento de especificaciones aprobado |
| Diseño de sistema | 3 semanas | Mockups y bases de datos configuradas |
| Desarrollo Backend | 4 semanas | Servicios backend funcionando |
| Desarrollo Frontend | 4 semanas | Frontend funcionando con datos simulados |
| Integración y pruebas | 3 semanas | Sistema integrado listo |
| Implementación y despliegue | 2 semanas | Sistema en producción |
| Seguimiento y Mantenimiento | Continuo | Primer informe de retroalimentación |

# **Documentación y Soporte**

El proyecto estará acompañado por una documentación técnica detallada, manuales de usuario, y soporte post-implementación para garantizar la adopción por parte de los actores interesados.