UNIVERSIDAD DE LOS LLANOS

Facultad de Ciencias Básicas e Ingeniería

Programa de Ingeniería de Sistemas

Manual Técnico del Proyecto de SoftPeces: análisis de carne con IA para piscicultura

Integrantes:

A. Perez Pedraza

J. Andres Ariza

C. Andres Cordoba

Y. Fabio Rey

N. Antonio Royert

O. Ignacio Galindo

Ingenieria de Software I

Universidad de los Llanos

2025

Resumen—En el presente manual técnico se describen de manera detallada las actividades realizadas durante las diferentes fases del desarrollo de SoftPeces, un sistema para gestionar estaciones, tanques, lotes y muestreos en piscicultura, integrado con un modelo de Inteligencia Artificial (ONNX) para analizar fotografías de carne de tilapia. Se describen los requerimientos, historias de usuario, arquitectura, diagramas UML y los avances logrados durante el primer sprint bajo metodología SCRUM.

Abstract--This technical manual provides a detailed description of the activities carried out during the different phases of the software development of SoftPeces, a system for managing stations, tanks, batches, and samplings in aquaculture, integrated with an Artificial Intelligence model (ONNX) for analyzing tilapia fillet images. It outlines the requirements, user stories, architecture, UML diagrams, and progress achieved during the first SCRUM sprint.

i. INTRODUCCIÓN

El presente documento describe de manera detallada el desarrollo del sistema SoftPeces, una aplicación de escritorio orientada a la gestión integral de piscicultura y al análisis automatizado de calidad de carne de tilapia mediante un modelo de Inteligencia Artificial en formato ONNX**.**
Este sistema integra módulos para administrar estaciones, tanques, lotes, muestreos y evidencias fotográficas, proporcionando una solución completa para la gestión operativa y la toma de decisiones en procesos productivos piscícolas.

El proceso de desarrollo se basó en la metodología ágil SCRUM, siguiendo fases estructuradas de análisis, diseño, implementación, pruebas y despliegue incremental.
La primera fase consistió en identificar los requisitos funcionales y no funcionales mediante entrevistas y revisión del proceso productivo, definiendo los actores: Administrador**,** Operario **e** Inspector de Calidad**.**

Posteriormente, durante la fase de diseño, se elaboraron los diagramas conceptuales y estructurales del sistema, incluidas las historias de usuario, el modelo lógico, los diagramas UML (casos de uso, clases, actividades, estados y secuencias) y la arquitectura en capas.

Actualmente, el proyecto se encuentra en una versión funcional que integra la autenticación por roles, la gestión de estaciones y tanques, la gestión de lotes, el registro de muestreos y la carga de fotografías con metadatos. Adicionalmente, se ha incorporado una primera integración con el modelo de Inteligencia Artificial basado en el algoritmo de yolo11 y exportado en ONNX para la clasificación de imágenes y el cálculo de indicadores de calidad, así como el envío de correos para recuperación de contraseña y el registro de la ubicación geográfica de las estaciones. Algunos aspectos se mantienen en evolución (por ejemplo, el ajuste fino de reglas de evaluación, la completitud de la trazabilidad por usuario y la medición formal de rendimiento), pero la arquitectura ya está preparada para su extensión en futuros sprints.

Este documento constituye una guía de referencia técnica para futuros desarrollos y mantenimiento del sistema SoftPeces.

ii. DESARROLLO

A. Análisis

Para la fase de análisis, se aplicó un cuestionario y una entrevista a operarios e inspectores del sector piscícola del Meta, con el fin de identificar sus necesidades y priorizar funcionalidades.

A partir de esta información se definieron dos categorías de requisitos:

- Requisitos Funcionales (RF)
- Requisitos No Funcionales (RNF)

También se construyeron historias de usuario y tablas de especificación.

Usuarios participantes

Se encuestaron 19 personas entre operarios, administradores e inspectores de calidad.
Los resultados indicaron que:

- Se requiere una plataforma robusta para registrar estaciones, tanques y lotes.
- El proceso de muestreo necesita evidencia fotográfica estructurada.
- El inspector requiere una herramienta de análisis automatizado con IA.
- La trazabilidad debe mantener registro de quién realizó cada acción.
- Se identificó la necesidad de procesar imágenes con calidad mínima aceptable

La siguiente parte en la fase de análisis fue la identificación de requerimientos, así como también categorizarlos en requerimientos funcionales y no funcionales

REQUERIMIENTOS FUNCIONALES	REQUERIMIENTOS NO FUNCIONALES
● **Autenticación y roles** ;El sistema debe permitir iniciar sesión y asignar permisos según rol (operario, inspector, administrador). ● **Gestión de estaciones y tanques**: Crear/editar Estación Piscícola y	● **Rendimiento**: clasificar una foto ≤ 2 s en hardware objetivo; generar reporte ≤ 5 s (100 registros). ● **Disponibilidad**: 99 % horario laboral. ● **Seguridad**: credenciales cifradas; control de acceso

Tanque; asociar tanques a una estación. ● **Gestión de lotes**: Crear/editar Lote con: especie, fecha Inicio, cantidad Peces, estado (en engorde / listo muestreo / cerrado). ● **Registrar muestreo**: El inspector registra un Muestreo de un Lote con fecha, tamaño Muestra y método ● **Capturar y almacenar fotografías**: Adjuntar **FotoMuestra** al muestreo (una o varias) con metadatos: uri/ruta, (lomo/piel/ojo/filete), timestamp. parte ● **Clasificación automática por IA**: Procesar cada **FotoMuestra** y generar **ResultadoIA** con puntajes (ojos, piel, branquias, textura), clase de calidad (A–C o 1–5) y confianza. ● **Consolidar evaluación del lote**: A partir de los Resultados IA del muestreo, calcular **Evaluación Calidad** (norma/escala, puntaje Global y decisión: apto, reproceso, descarte). ● **Trazabilidad de decisiones**: Guardar quién registró el muestreo y quién supervisó la evaluación (Usuario Evaluacion Calidad). → Muestreo / ● **Soporte y evidencia**: Visualizar la imagen original asociada a cada decisión (evidencia para auditor o cliente). ● **Centro de recursos (enlaces externos)**: Mostrar un apartado de "**Recursos del sector**" con	por rol; registro de auditoría (quién/qué/cuándo). ● **Protección de datos**: almacenamiento de imágenes con control de acceso y eliminación segura (retención configurable). ● **Escalabilidad**: soportar ≥ 10k imágenes almacenadas sin degradación perceptible en consulta. ● **Interoperabilidad API IA:** El sistema debe permitir la comunicación entre el frontend/backend desarrollado en **Java** y el módulo de **clasificación IA en Python** a través de una **API REST** documentada y segura. ● El envío de fotografías debe realizarse en formato estándar (por ejemplo, JPEG/PNG base64 o multipart). ● La respuesta de la API debe contener puntajes por criterios, clase de calidad y nivel de confianza en formato **JSON**. ● El tiempo máximo de respuesta no debe superar los 2 segundos por imagen.
enlaces a herramientas IA externas (inventario, costos, etc.) solo como link hub, sin integrarlas al sistema.	
OTROS ELEMENTOS A CONSIDERAR: REQUERIMIENTOS DE PROYECTO, RESTRICCIONES, RIESGOS….	
RESTRICCIONES: ● **Alcance**: el sistema **no** implementa inventarios, ventas ni costos; solo calidad de carne por imágenes y gestión de muestreos. ● **Datos**: tamaño de muestra de encuesta reducido (19), por lo que los hallazgos orientan diseño inicial (no inferencia estadística). ● **Conectividad**: la captura puede ocurrir offline; el procesamiento puede requerir conexión (si el modelo reside en servidor).	
RIESGOS: ● **Calidad de imágenes**: mala iluminación/encuadre reduce precisión → incluir verificador de calidad de foto antes de clasificar. ● **Resistencia al cambio**: 42,1 % respondió que el manual "sí es suficiente" → capacitación y panel de "confianza del modelo". ● **Costos de almacenamiento**: muchas fotos → políticas de retención y compresión. **Normativa**: cambios en criterios (p. ej., actualizaciones de buenas prácticas) → módulo de reglas parametrizable	

Tabla 1. Requerimientos funcionales y no funcionales

Se definieron los requerimientos para cada sprint.

- **SPRINT BACKLOG**

1). Primera entrega (Sprint 1)

En este sprint se implementaron las funcionalidades base del sistema **SoftPeces**, enfocadas en permitir la operación inicial y la gestión estructural del entorno piscícola.
Las actividades desarrolladas fueron:

- **Módulo de autenticación y roles**
  Gestiona accesos diferenciados para Administrador, Operario e Inspector de Calidad.
- **Registro y administración de estaciones piscícolas**
  Incluye nombre, ubicación y estructura productiva.
- **Gestión de tanques**
  Creación, edición y asociación con estaciones.
- **Gestión inicial de lotes**
  Registro de especie, fecha de inicio, cantidad de peces y estado productivo.

- **Registro de muestreos**
  Creación de muestreos asociados a un lote como preparación para la toma fotográfica.
- **Carga de fotografías** *(solo interfaz; análisis IA en sprint posterior)*
  Estructuración de metadatos: parte analizada, timestamp y referencia al muestreo.

Estas funcionalidades constituyen la base operativa sobre la cual se integrará el módulo de análisis por IA.

2). Segunda entrega (Sprint 2)

En este sprint se incorporarán los procesos de análisis inteligente y trazabilidad avanzada.
Las actividades previstas incluyen:

- **Integración del modelo ONNX para clasificación automática de imágenes**
  Evaluación de textura, ojos, piel y branquias, generando puntaje, clase y confianza.
- **Implementación de consultas y reportes avanzados**
  Filtros por fecha, estación, tanque, lote, especie y clase de calidad.
- **Generación automática de informes y evaluaciones del lote**
  Decisión: *apto*, *reproceso*, o *descarte*, según reglas configurables.
- **Trazabilidad detallada**
  Registro de quién creó, modificó o evaluó cada muestreo.
- **Histórico completo de fotografías y resultados IA**
  Visualización de evidencia para auditorías internas o clientes.

Código: **HU-01**	Usuario: Cualquier usuario (Operario, Inspector, Administrador)
Título: Iniciar sesión y acceder según rol	
Requerimiento funcional asociado (Id): RF01	
Prioridad: Alta	Tipo: Historia
Descripción: Como usuario del sistema, quiero iniciar sesión con mis credenciales para acceder únicamente a las funciones habilitadas para mi rol, manteniendo la seguridad de la información.	
Criterios de aceptación: ● Éxito: Dado usuario y contraseña válidos, cuando inicio sesión, entonces veo el panel correspondiente a mi rol y se registra mi ingreso (fecha/hora/IP). ● Alternativo: Si olvido la contraseña, puedo solicitar recuperación vía correo. ● Error: Con credenciales inválidas, el sistema rechaza el acceso y muestra un mensaje claro sin revelar si el correo existe.	

Tabla 2.Historia de usuario 1.

Código: **HU-02**	Usuario: Operario / administrador
Título: Crear y actualizar lotes	
Requerimiento funcional asociado (Id): RF03	
Prioridad: Alta	Tipo: Historia
Descripción: Como operario, quiero crear/editar lotes con especie, fecha de inicio, cantidad de peces y estado para llevar control de la producción.	
Criterios de aceptación: ● Éxito: Puedo registrar un lote con campos obligatorios; queda asociado a tanque y especie válidos. ● Alternativo: Si el tanque está lleno, el sistema sugiere otro tanque disponible. ● Error: Si falta un campo obligatorio o hay incoherencias (p. ej. cantidad negativa), se bloquea el guardado y se resalta errores..	

Tabla 3. Historia de usuario 2.

Código: **HU-03**	Usuario: Inspector / Operario
Título: Registrar muestreo de un lote	
Requerimiento funcional asociado (Id): RF04	
Prioridad: Alta	Tipo: Historia
Descripción: Como inspector, quiero registrar muestreos indicando fecha, tamaño de muestra y método para iniciar la evaluación de calidad del lote.	
Criterios de aceptación: ● Éxito: Dado un lote "listo muestreo", puedo crear un muestreo y queda vinculado al lote y al usuario que lo registra. ● Alternativo: Si el lote no está en estado "listo muestreo", el sistema permite agendar el muestreo y notifica. ● Error: Si el tamaño de muestra es 0 o inválido, se impide el registro.	

Tabla 4. Historia de usuario 3.

Código: **HU-04**	Usuario: Inspector
Título: Adjuntar fotografías al muestreo	
Requerimiento funcional asociado (Id): RF05	
Prioridad: Alta	Tipo: Historia
Descripción: Como inspector, quiero adjuntar una o varias fotografías al muestreo con metadatos (parte, timestamp) para evidenciar la evaluación.	

Criterios de aceptación: ● Éxito: Puedo subir imágenes (JPEG/PNG) y el sistema guarda uri, parte (lomo/piel/ojo/filete) y timestamp. ● Alternativo: Si la foto no cumple calidad mínima (borrosa/oscura), el sistema advierte y permite repetir. ● Error: Si el archivo no es una imagen o supera el tamaño permitido, el sistema lo rechaza con mensaje claro.

Tabla 5. Historia de usuario 4.

Código: **HU-05**	Usuario: Inspector / Operario
Título: Obtener resultado IA por fotografía	
Requerimiento funcional asociado (Id): RF-06 (y RNF-6 Interoperabilidad API IA)	
Prioridad: Critica	Tipo: Historia
Descripción: Como usuario, quiero que cada fotografía sea clasificada automáticamente (puntajes por criterios, clase y confianza) para reducir tiempo y subjetividad.	
Criterios de aceptación: ● Éxito: Dada una foto válida, al enviarla a la API, el sistema recibe JSON con {ojos, piel, branquias, textura, clase, confianza} ≤ 2 s. ● Alternativo: Si la API tarda más de 2 s, se reintenta una vez y se notifica "procesando"; se permite continuar con otras fotos. ● Error: Si la API no responde o devuelve error, el sistema registra incidente y muestra opción de reintento sin perder la foto.	

Tabla 6. Historia de usuario 5.

Código: **HU-06**	Usuario: Inspector / Administrador
Título: Generar evaluación de calidad del muestreo	
Requerimiento funcional asociado (Id): RF-07, RF-10	
Prioridad: Alta	Tipo: Historia
Descripción: Como inspector, quiero consolidar una Evaluación Calidad aplicando las reglas activas (p. ej., NTC 1443) para emitir decisión del lote (apto/reproceso/descarte).	
Criterios de aceptación: ● Éxito: Al cerrar el muestreo, el sistema calcula puntaje global y decisión según ponderaciones configuradas y guarda la versión de reglas utilizada. ● Alternativo: Un administrador puede editar ponderaciones y recalcular antes de confirmar.	
● Error: Si faltan resultados IA de fotos requeridas, el sistema impide el cierre y muestra cuáles faltan.	

Tabla 7. Historia de usuario 6.

Código: **HU-07**	Usuario: Administrador / Inspector
Título: Generar evaluación de calidad del muestreo	
Requerimiento funcional asociado (Id): RF-09	
Prioridad: Media	Tipo: Historia
Descripción: Como administrador, quiero generar reportes por fecha/especie/lote con distribución de clases y enlaces a las imágenes para auditoría y clientes.	
Criterios de aceptación: ● Éxito: Dado un filtro, el sistema genera reporte en pantalla y permite exportar PDF/CSV ≤ 5 s (100 registros). ● Alternativo: Si hay muchas imágenes, se muestran miniaturas y enlaces a descarga bajo permisos. ● Error: Sin datos para el rango, se informa "sin registros" y se permiten nuevos filtros.	

Tabla 8. Historia de usuario 7.

Código: **HU-08**	Usuario: Administrador
Título: Ver trazas de registro y supervisión	
Requerimiento funcional asociado (Id): RF-08	
Prioridad: Media	Tipo: Historia
Descripción: Como administrador, quiero ver quién registró cada muestreo y quién supervisó la evaluación, con fecha/hora, para garantizar trazabilidad.	
Criterios de aceptación: ● Éxito: En el detalle del muestreo aparecen usuarios, roles y marcas de tiempo de cada paso. ● Alternativo: Se puede exportar la traza junto al reporte del lote. ● Error: Si el usuario no tiene permiso, no puede acceder a la traza.	

Tabla 9. Historia de usuario 8.

Código: **HU-09**	Usuario Administrador / Operario
Título: Gestión de estaciones y tanques	
Requerimiento funcional asociado (Id): RF-02	
Prioridad: Media	Tipo: Historia

Descripción: Como administrador u operario, quiero poder crear, editar y mantener el registro de estaciones piscícolas y sus tanques asociados, de manera que el sistema refleje la estructura real de producción y permita organizar correctamente los lotes de peces.
Criterios de aceptación: ● Éxito: Dado un usuario con rol válido, cuando crea una estación, entonces debe registrar nombre y ubicación; cuando crea un tanque, debe asignarlo a una estación existente y establecer su capacidad. ● Alternativo: Si un tanque ya tiene lotes activos, al intentar editar capacidad se muestra advertencia y se permite solo aumento, no reducción. ● Error: Si se intenta crear un tanque sin estación asociada o dejar campos obligatorios vacíos, el sistema debe rechazar la acción con mensaje claro.

Tabla 10. Historia de usuario 9.

Identificación del Requerimiento:	RF01
Nombre del Requerimiento:	Autenticación y roles
Descripción del Requerimiento:	El sistema debe permitir iniciar sesión con credenciales y asignar permisos según el rol del usuario (operario, inspector, administrador). Cada rol tendrá acceso únicamente a las funcionalidades autorizadas.
Requerimiento No Funcional:	● RNF-02 (Disponibilidad) ● RNF-03 (Seguridad).
Prioridad del Requerimiento: Alta	

Tabla 11. Requerimiento funcional 1.

Identificación del Requerimiento:	RF02
Nombre del Requerimiento:	Gestión de estaciones y tanques
Descripción del Requerimiento:	El sistema permitirá crear y editar estaciones piscícolas y tanques, asegurando que cada tanque esté asociado a una estación y registre su capacidad.
Requerimiento No Funcional:	● RNF-03 (Seguridad) ● RNF-05 (Escalabilidad).
Prioridad del Requerimiento: Media	

Tabla 12. Requerimiento funcional 2.

Identificación del Requerimiento:	RF03
Nombre del Requerimiento:	Gestión de lotes
Descripción del Requerimiento:	El sistema permitirá crear y actualizar lotes con información sobre especie, fecha de inicio, cantidad de peces y estado (en engorde, listo muestreo, cerrado).
Requerimiento No Funcional:	● RNF-03 (Seguridad) ● RNF-05 (Escalabilidad)
Prioridad del Requerimiento: Alta	

Tabla 13. Requerimiento funcional 3.

Identificación del Requerimiento:	RF04
Nombre del Requerimiento:	Registrar muestreo
Descripción del Requerimiento:	El sistema permitirá al inspector registrar muestreos de un lote, indicando fecha, tamaño de la muestra y método de selección.
Requerimiento No Funcional:	● RNF-03 (Seguridad) ● RNF-05 (Escalabilidad).
Prioridad del Requerimiento: Alta	

Tabla 14. Requerimiento funcional 4.

Identificación del Requerimiento:	RF05
Nombre del Requerimiento:	Capturar y almacenar fotografías
Descripción del Requerimiento:	El sistema permitirá adjuntar una o varias fotografías a un muestreo, con metadatos como uri/ruta, parte del pez (lomo, piel, ojo, filete) y timestamp.
Requerimiento No Funcional:	● RNF-04 (Protección de datos) ● RNF-05 (Escalabilidad).
Prioridad del Requerimiento: Alta	

Tabla 15. Requerimiento funcional 5.

Identificación del Requerimiento:	RF06
Nombre del Requerimiento:	Clasificación automática por IA
Descripción del Requerimiento:	El sistema procesará cada fotografía y generará un ResultadoIA con puntajes (ojos, piel, branquias, textura), clase de calidad (A–C o 1–5) y confianza, utilizando un modelo de IA.
Requerimiento No Funcional:	● RNF-01 (Rendimiento) ● RNF-06 (Interoperabilidad API IA).
Prioridad del Requerimiento: Critica	

Tabla 16. Requerimiento funcional 6.

Identificación del Requerimiento:	RF07
Nombre del Requerimiento:	Consolidar evaluación del lote
Descripción del Requerimiento:	A partir de los resultados de IA, el sistema consolidará una Evaluación de Calidad por muestreo, aplicando reglas normativas (p. ej., NTC 1443) para determinar la decisión final (apto, reproceso, descarte).
Requerimiento No Funcional:	● RNF-01 (Rendimiento) ● RNF-03 (Seguridad).
Prioridad del Requerimiento: Alta	

Tabla 17. Requerimiento funcional 7.

Identificación del Requerimiento:	RF08
Nombre del Requerimiento:	Trazabilidad de decisiones
Descripción del Requerimiento:	El sistema guardará los registros de quién creó un muestreo y quién supervisó la evaluación de calidad, incluyendo fecha y hora.
Requerimiento No Funcional:	● RNF-03 (Seguridad) ● RNF-04 (Protección de datos).
Prioridad del Requerimiento: Media	

Tabla 18. Requerimiento funcional 8.

Identificación del Requerimiento:	RF09
Nombre del Requerimiento:	Soporte y evidencia
Descripción del Requerimiento:	El sistema permitirá visualizar la imagen original asociada a cada decisión de calidad, sirviendo como evidencia para auditorías o clientes.
Requerimiento No Funcional:	● RNF-04 (Protección de datos) ● RNF-05 (Escalabilidad).
Prioridad del Requerimiento: Media	

Tabla 19. Requerimiento funcional 9.

Identificación del Requerimiento:	RF10
Nombre del Requerimiento:	Centro de recursos (enlaces externos)
Descripción del Requerimiento:	El sistema mostrará un apartado de recursos con enlaces a herramientas externas de IA (inventario, costos, etc.), solo como referencia, sin integrarlas al software.
Requerimiento No Funcional:	● RNF-02 (Disponibilidad) ● RNF-05 (Usabilidad).
Prioridad del Requerimiento: Baja	

Tabla 20. Requerimiento funcional 10.

B. Diseño

La fase de diseño del sistema **SoftPeces** se desarrolló siguiendo los lineamientos de la Ingeniería de Software y manteniendo coherencia con los requerimientos recolectados en la etapa de análisis.
En esta sección se presentan los diferentes modelos estructurales, funcionales y de comportamiento que permiten comprender cómo operará el sistema a nivel lógico e interno.

El objetivo principal del diseño es proporcionar una visión clara, modular y extensible del sistema, facilitando la implementación en Java/JavaFX y la integración del modelo de Inteligencia Artificial.

El diseño se estructura en nueve componentes principales:

B.1. El Modelo Entidad–Relación establece la estructura de las entidades que conforman el sistema, definiendo cómo se relacionan entre sí.
En SoftPeces, el MER permite visualizar la forma en que se manejan las estaciones piscícolas, los tanques, los lotes y los muestreos, además del almacenamiento de fotografías y la asociación de resultados generados por la IA.

Este modelo define:

- Entidades principales: **Estación**, **Tanque**, **Lote**, **Muestreo**, **Foto**, **ResultadoIA**, **Usuario**.
- Relaciones clave:
  - Una estación contiene múltiples tanques.
  - Un tanque gestiona múltiples lotes.
  - Un lote puede ser muestreado varias veces.
  - Cada muestreo puede incluir múltiples fotografías.
  - Cada fotografía genera un resultado IA asociado.
- Reglas de integridad: llaves primarias, llaves foráneas, restricciones y cardinalidades.

Este diseño garantiza la trazabilidad de cada proceso productivo y su relación con los registros de calidad detectados por la IA.

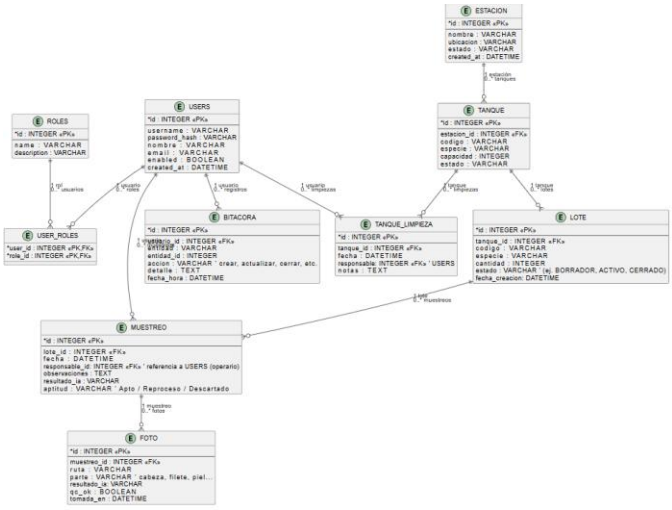


Figura 1. Modelo Entidad – Relación

B.2. El diagrama de casos de uso muestra las interacciones entre los actores (Administrador, Operario e Inspector de Calidad) y las funcionalidades del sistema.
Este modelo permitió identificar las acciones permitidas para cada rol y asegurar el cumplimiento de los requerimientos funcionales RF01 a RF10.

El diseño destaca:

- Accesos diferenciados según rol.
- Flujos principales:
  - Gestionar estaciones y tanques.
  - Gestionar lotes.
  - Registrar muestreos.
  - Cargar fotografías.
  - Ejecutar análisis IA.
  - Obtener reportes, evidencia y trazabilidad.
- Extensiones y alternativas de uso, como reintentos de IA o advertencias por fotos de baja calidad.

Este diagrama es clave para planificar los módulos de interfaz y los controladores JavaFX.

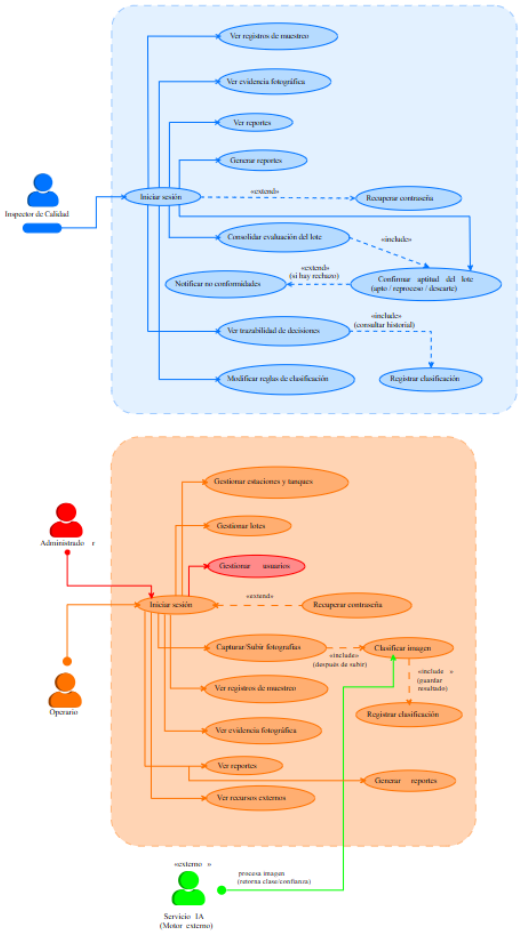


Figura 2. Diagrama de casos de uso.

B.3. El modelo de dominio representa la estructura conceptual del sistema independiente de la tecnología.
Permite visualizar las entidades como objetos del dominio piscícola y sus responsabilidades.

Incluye:

- Objetos principales: Estación, Tanque, Lote, Muestreo, Foto, ResultadoIA, ReglaClasificación, Usuario.
- Relaciones entre objetos: composición, agregación y dependencias.
- Reglas del negocio:
  - Un lote solo puede ser muestreado cuando su estado es “Listo para muestreo”.
  - No se puede editar la capacidad de un tanque si tiene un lote activo.
  - Una fotografía debe tener metadatos completos para ser analizada por IA.

El dominio garantiza coherencia con las prácticas reales de producción piscícola.

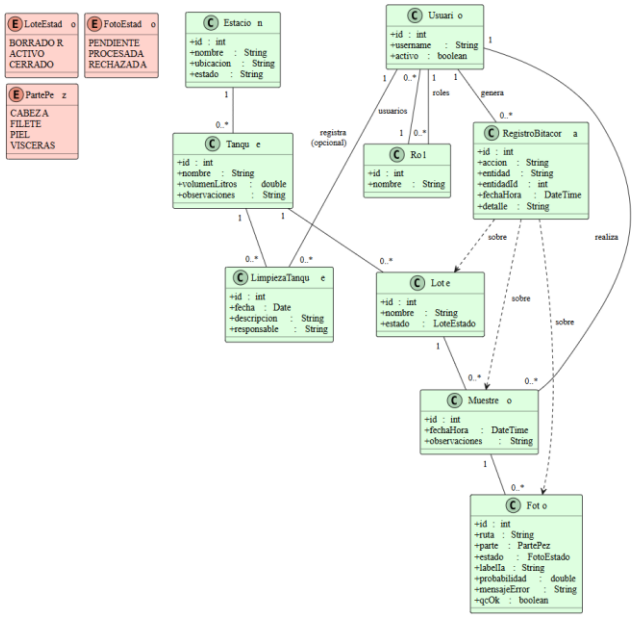


Figura 3. Modelo de Dominio

B.4. El diagrama de clases detalla la estructura interna del software a nivel orientado a objetos.
Representa las clases que formarán parte de la implementación en Java, sus atributos, métodos y las relaciones existentes entre ellas.

En SoftPeces, este diagrama incluye:

- Métodos de negocio para gestionar estaciones, tanques, lotes y muestreos.
- Clases del módulo IA para procesar imágenes, enviar datos al motor ONNX y recibir resultados.
- Repositorios y servicios para la persistencia en SQLite.
- Componentes de auditoría y control de roles.

Este diagrama sirve como guía directa para la codificación.

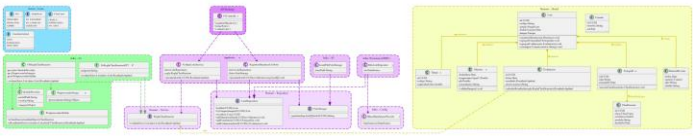


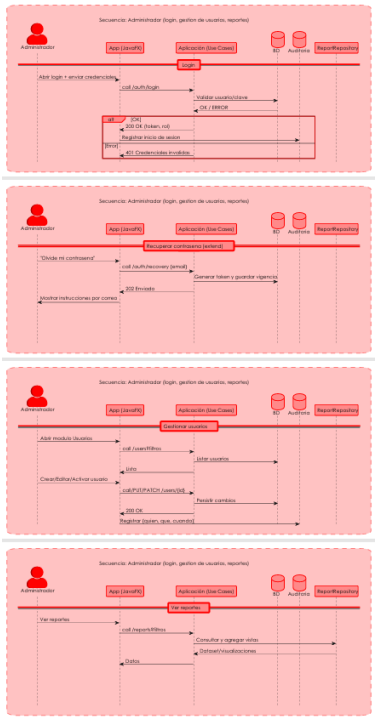
Figura 4. Diagrama de Clases

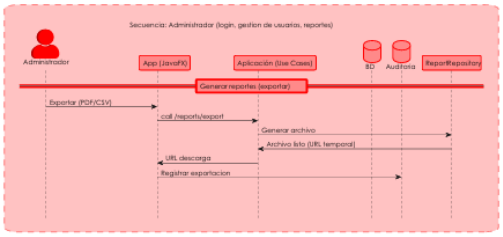
B.5. Los diagramas de secuencia representan el comportamiento dinámico del sistema, mostrando cómo se comunican los objetos y módulos en tiempo real.

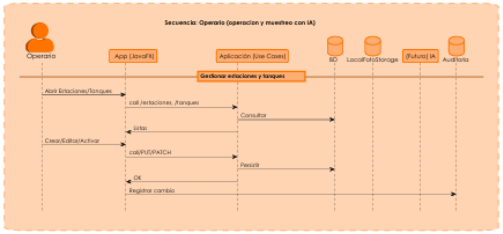
Se desarrollaron varios diagramas clave, entre ellos:

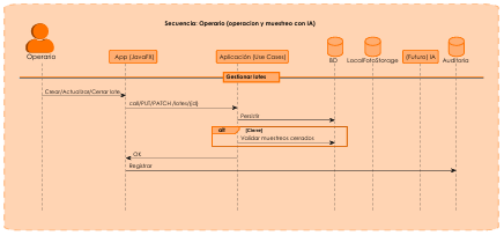
- **Inicio de sesión**
  Flujo de autenticación, verificación de credenciales y carga del panel según rol.
- **Gestión de estaciones y tanques**
  Flujo CRUD y validación de restricciones.
- **Gestión de lotes**
  Registro, edición, cierre y validación de muestreos.
- **Registro de muestreo**
  Asociar el muestreo con el lote y el usuario.
- **Carga de fotografías**
  Verificación de calidad de imagen, almacenamiento y envío a la IA.
- **Procesamiento IA**
  Envío de la foto al modelo ONNX, recepción del resultado y registro de la clasificación.
- **Evaluación del lote**
  Consolidación de resultados IA y determinación de aptitud: apto, reproceso o descarte.

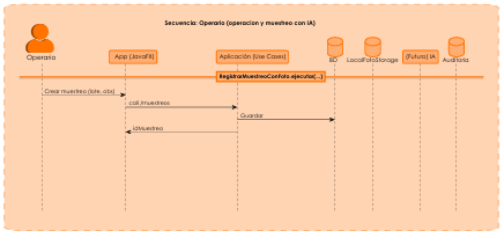
Estos diagramas permiten entender la lógica del sistema y planificar correctamente la implementación modular.

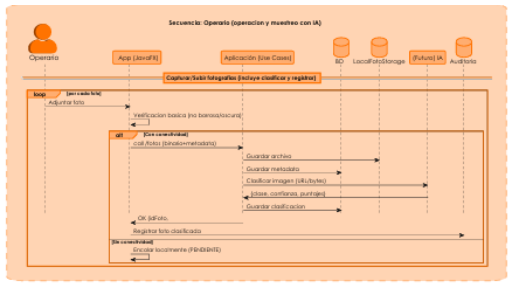


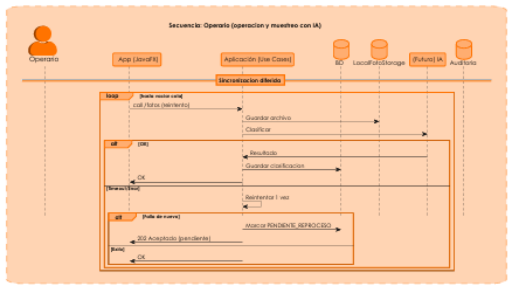


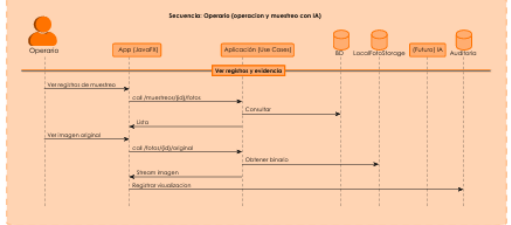


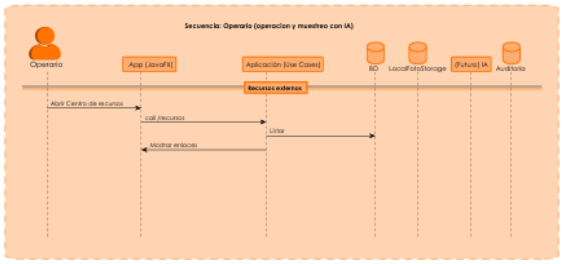


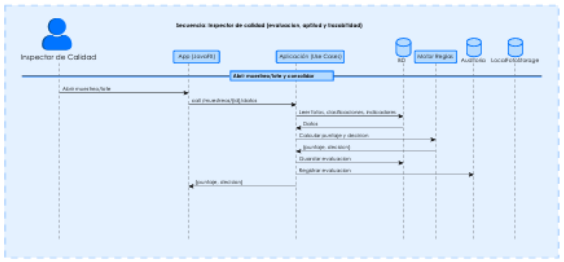


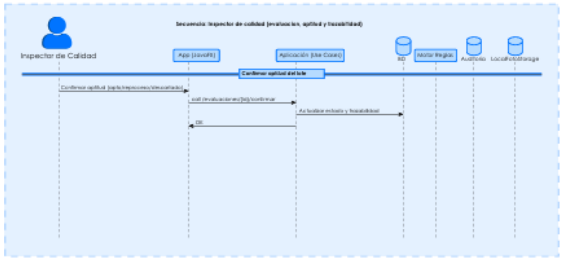


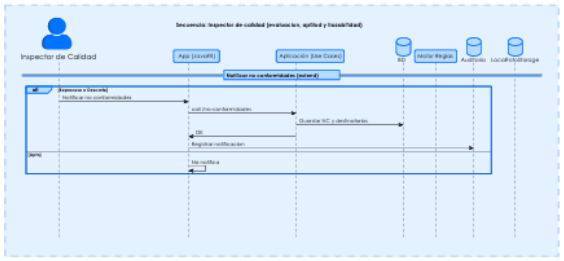


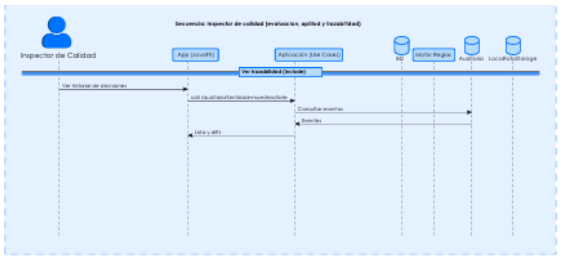












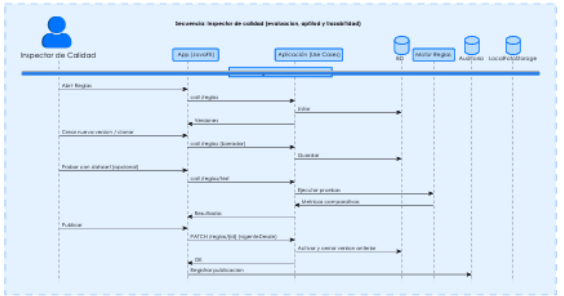


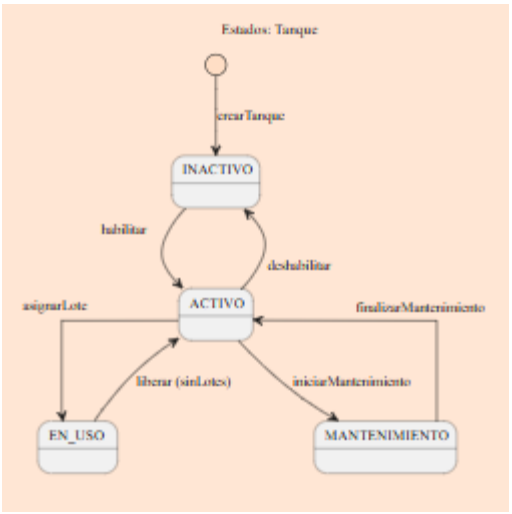
Figura 5. Diagrama de secuencia

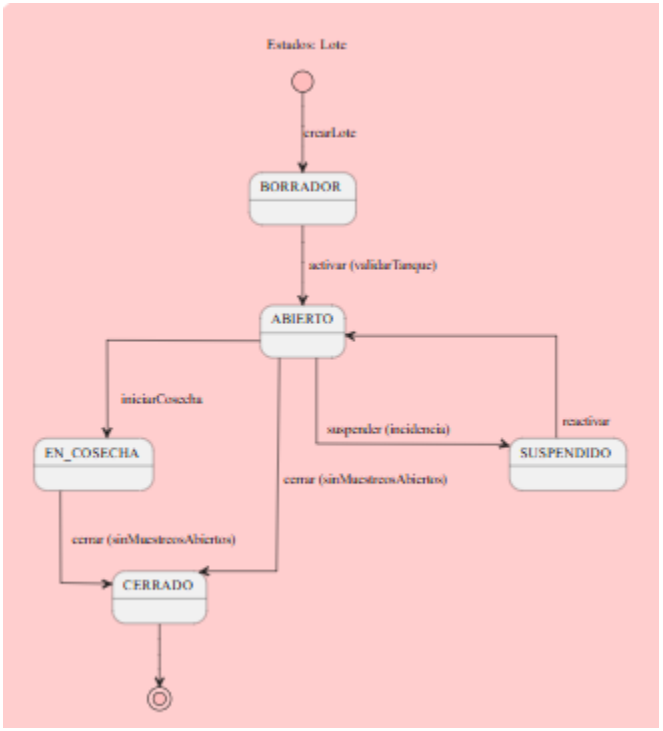
B.6. Los diagramas de estado representan los diferentes estados por los que pasan las entidades dinámicas del sistema.

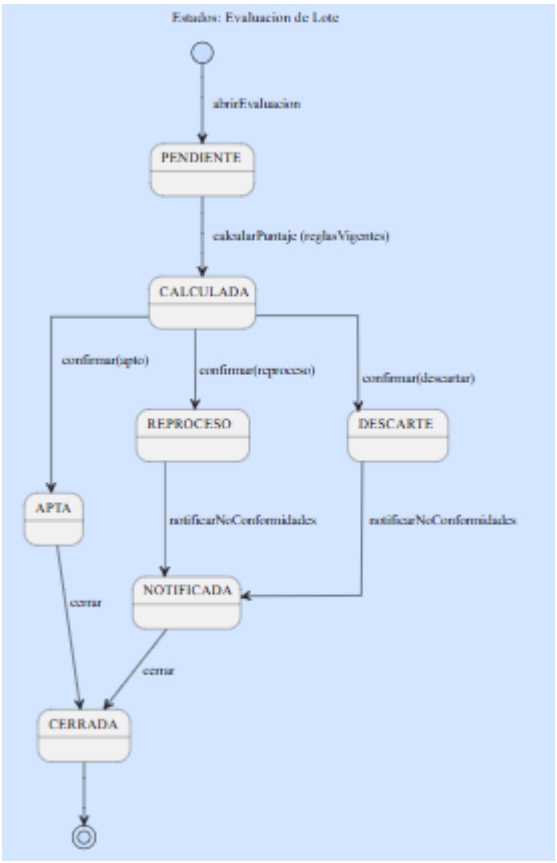
SoftPeces utiliza estados para:

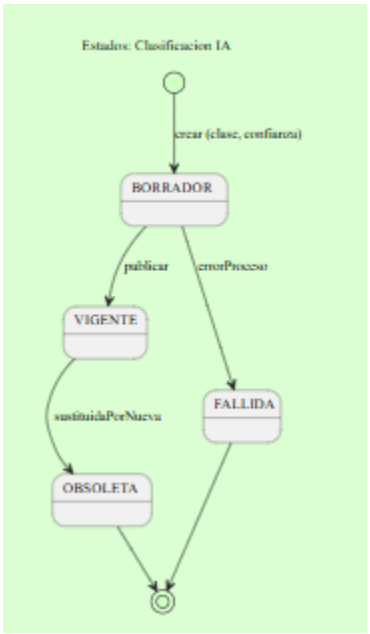
- **Lotes:** en engorde → listo muestreo → evaluado → cerrado.
- **Muestreos:** creado → en evaluación → consolidado.
- **Fotografías:** cargada → pendiente IA → clasificada → con error → reintento → lista.

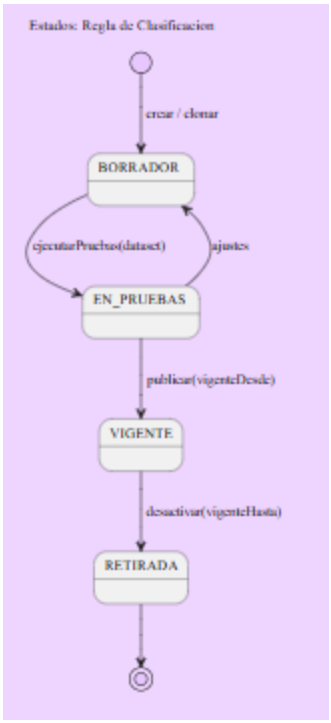
Estos modelos garantizan orden lógico, sobre todo en procesos donde interviene la IA.

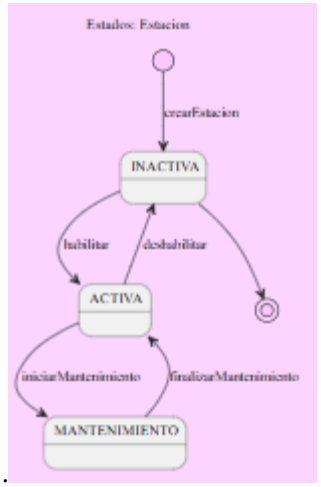


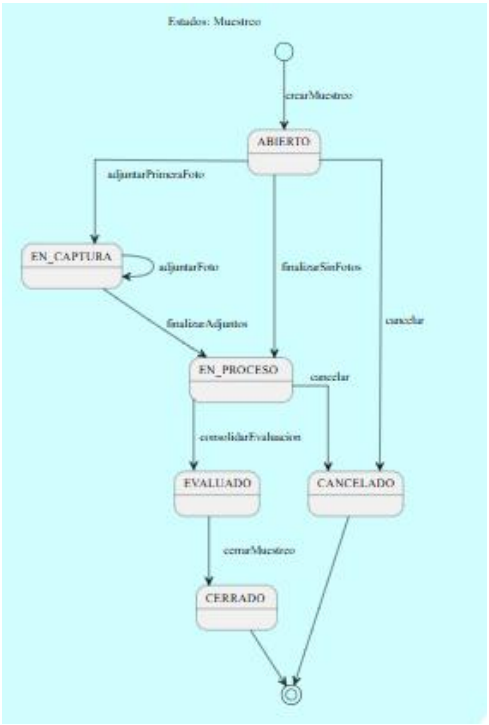


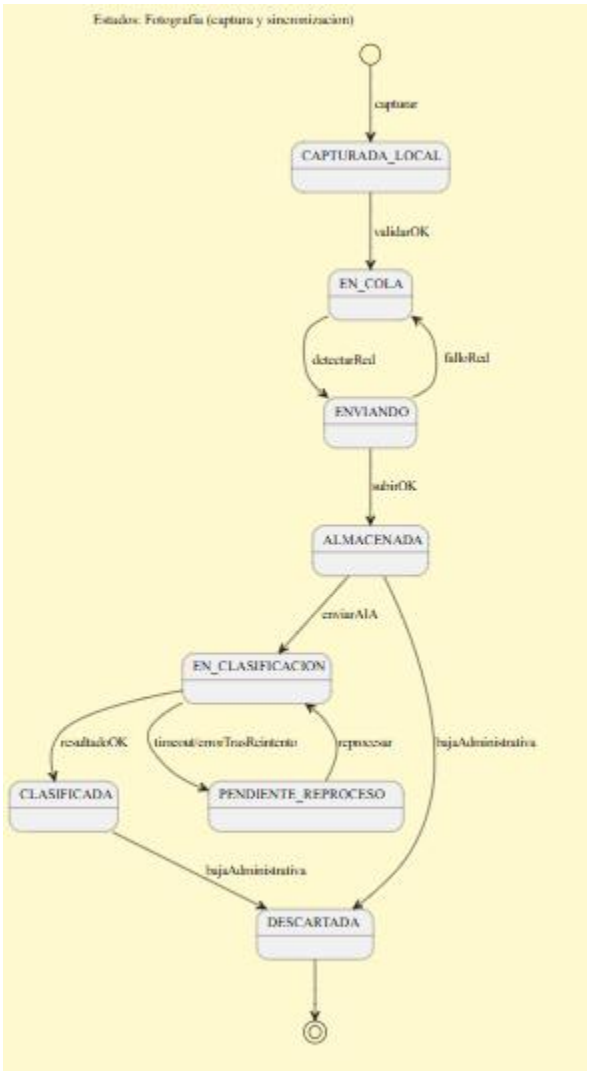


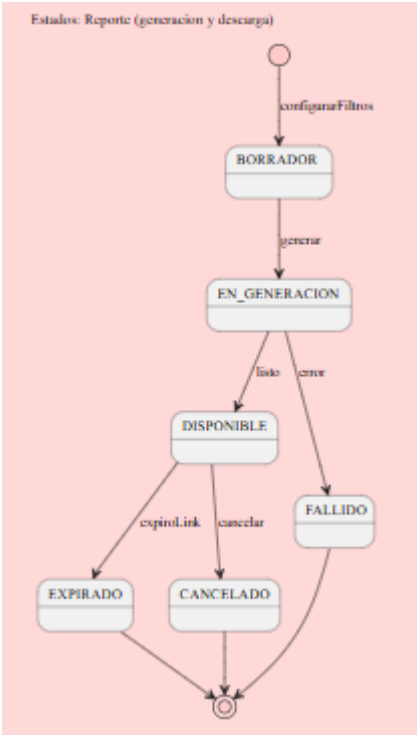


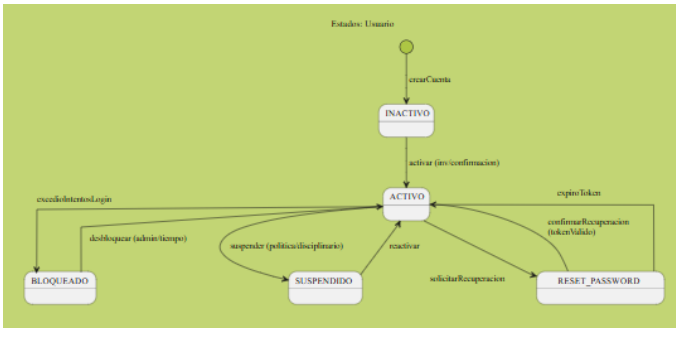

Estados: Regla de Clasificacion
crear / clonar
BORRADOR
ejecutarPruebas(dataset)
ajustes
EN_PRUEBAS
publicar(vigenteDesde)
VIGENTE
desactivar(vigenteHasta)
RETIRADA


Estados: Estacion
crearEstacion
INACTIVA
habilitar
deshabilitar
ACTIVA
iniciarMantenimiento
finalizarMantenimiento
MANTENIMIENTO


Estados: Muestreo
crearMuestreo
ABIERTO
adjuntarPrimeraFoto
EN_CAPTURA
adjuntarFoto
finalizarSinFotos
finalizarAdjuntos
cancelar
EN_PROCESO
cancelar
consolidarEvaluacion
EVALUADO
CANCELADO
cerrarMuestreo
CERRADO


Estados: Fotografia (captura y sincronizacion)
capturar
CAPTURADA_LOCAL
validarOK
EN_COLA
detectarRed
falloRed
ENVIANDO
subirOK
ALMACENADA
enviarAIA
EN_CLASIFICACION
resultadoOK
timeout/errorTrasReintento
reprocesar
bajaAdministrativa
CLASIFICADA
PENDIENTE_REPROCESO
bajaAdministrativa
DESCARTADA


Estados: Reporte (generacion y descarga)
configurarFiltros
BORRADOR
generar
EN_GENERACION
listo
error
DISPONIBLE
FALLIDO
expiroLink
cancelar
EXPIRADO
CANCELADO


Estados: Usuario
crearCuenta
INACTIVO
activar (trasConfirmacion)
ACTIVO
excedioIntentosLogin
expiroToken
desbloquear (admin/tiempo)
suspender (politica/disciplinario)
reactivar
confirmarRecuperacion (tokenValido)
BLOQUEADO
SUSPENDIDO
solicitarRecuperacion
RESET_PASSWORD

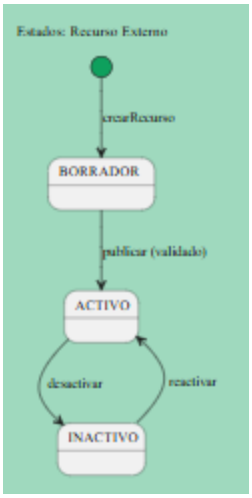


Figura 6. Diagrama de Estado

B.7. El diagrama de actividades representa el flujo de procesos dentro del sistema desde una perspectiva operativa, permitiendo visualizar la secuencia de acciones que ejecutan los usuarios y los componentes internos del software.
En **SoftPeces**, este diagrama es fundamental para describir cómo se desarrolla un muestreo con evidencia fotográfica y cómo interviene el módulo de IA dentro del flujo de trabajo.

El diseño del diagrama incluye:

- **Inicio del proceso por parte del inspector** al seleccionar un lote en estado “listo para muestreo”.
- **Registro de muestreo**, ingresando fecha, tamaño de muestra y método.
- **Toma o carga de fotografías**, verificando la calidad mínima requerida.
- **Procesamiento automático mediante IA**, enviando la fotografía al modelo ONNX y recibiendo clase y confianza.
- **Consolidación de la evaluación del lote**, calculando la aptitud final.
- **Almacenamiento de resultados y trazabilidad** para auditoría.

Este diagrama ilustra el comportamiento general del sistema y permite identificar puntos críticos en los que se aplican reglas del negocio, validaciones o procesamiento inteligente.

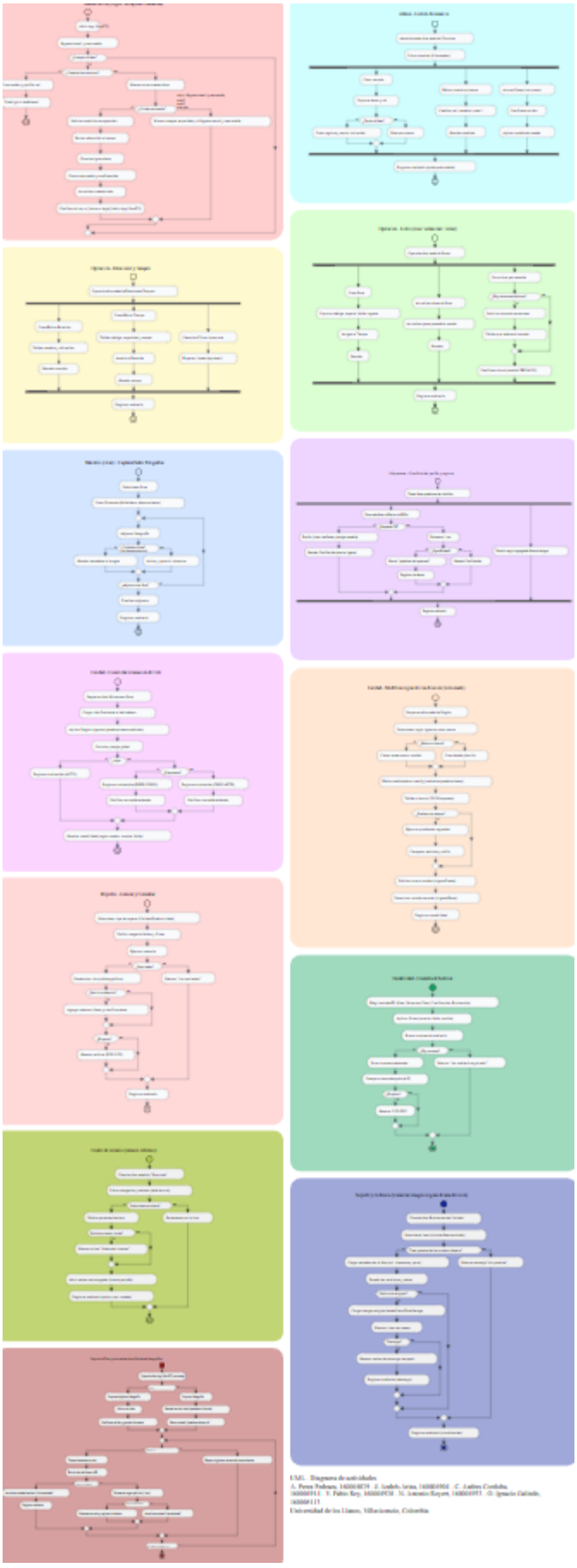


Figura 7. Diagrama de actividades

B.8. La arquitectura de SoftPeces se diseñó bajo un enfoque **multicapa**, garantizando independencia entre componentes, escalabilidad, mantenibilidad y separación clara de responsabilidades.
Este diseño facilita la futura integración del modelo de IA (ONNX Runtime), la persistencia en SQLite y la interfaz construida en JavaFX.

La arquitectura contempla cuatro capas principales:

1. Capa de Presentación (UI – JavaFX)

Responsable de las pantallas, controladores gráficos y lógica de interacción con el usuario.
Se diseñó para:

- soportar diferentes roles (Administrador, Operario e Inspector),
- validar datos antes de enviarlos a la capa de aplicación,
- mostrar resultados IA y reportes.

2. Capa de Aplicación (Servicios – Use Cases)

Implementa los flujos del negocio según los casos de uso.
Coordina acciones como:

- crear estaciones, tanques y lotes,
- registrar muestreos,
- ejecutar la clasificación IA y consolidar evaluaciones,
- gestionar reportes y trazabilidad.

3. Capa de Dominio (Entidades y lógica interna)

Modelo sólido que contiene:

- entidades como Estación, Tanque, Lote, Muestreo, Foto, ResultadoIA, Evaluación, Usuario,
- reglas de negocio, validaciones y estados.

Esta capa es independiente de frameworks, bases de datos y UI.

4. Capa de Infraestructura

Encargada de:

- persistencia en SQLite,
- gestión de archivos e imágenes,
- conexión con el modelo ONNX para clasificación IA,
- auditoría, logs y servicios externos.

Esta organización garantiza que el sistema sea flexible, escalable y preparado para integraciones futuras.

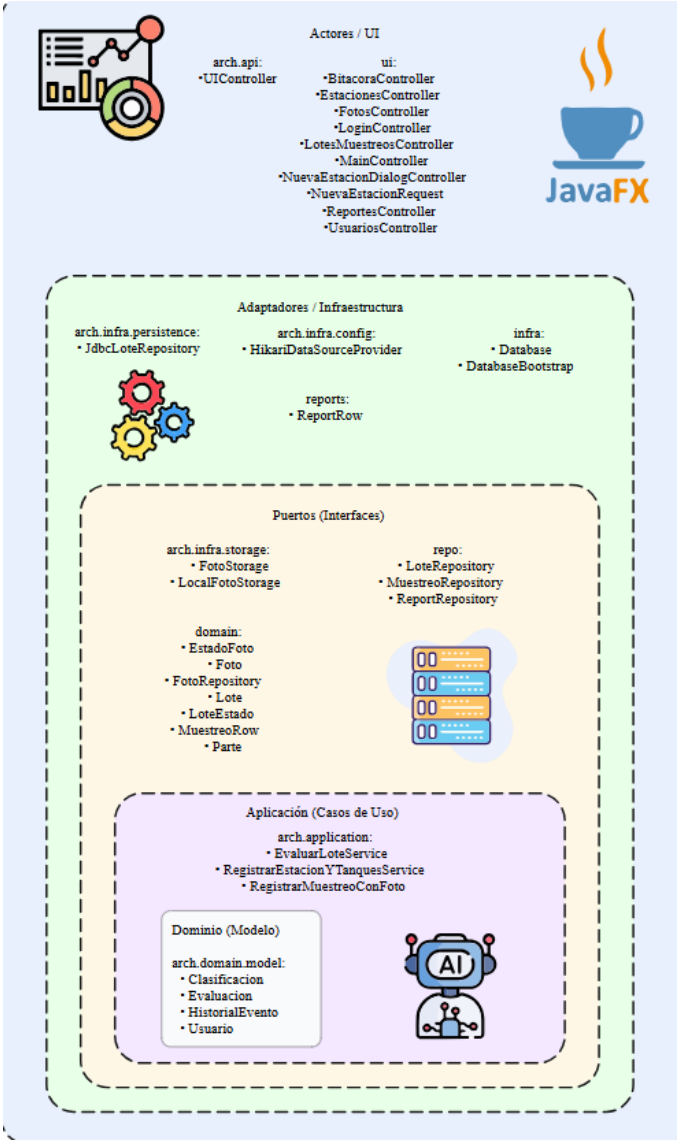


Figura 8. Diagrama de Arquitectura

B.9. Los mockups representan el diseño preliminar de las interfaces de usuario desarrolladas en JavaFX.
Su objetivo es mostrar la organización visual y la experiencia de interacción antes de ejecutar la implementación real.

El diseño de los mockups de SoftPeces considera:

A. Pantalla de Login

- Campos de usuario y contraseña.
- Botón de acceso.
- Rol asignado definido desde backend.
- Recuperación de contraseña (opcional).

B. Panel Administrador

- Menú lateral con módulos:
  - Estaciones
  - Tanques
  - Lotes

    - Usuarios
    - Reportes
    - Auditoría
- Área principal para formularios y tablas de datos.

C. Panel Operario

- Acceso a estaciones y tanques.
- Registro y actualización de lotes.
- Visualización del estado productivo.

D. Panel Inspector de Calidad

- Selección de lote “listo para muestreo”.
- Carga de fotografías con verificación visual inmediata.
- Vista previa de resultados IA.
- Panel de consolidación de evaluación final.

E. Vista de Reportes

- Filtros avanzados (fecha, especie, lote, estación).
- Resumen estadístico por clases A–C o escala 1–5.
- Miniaturas de imágenes y enlace a evidencia completa.
- Exportación a PDF o CSV.

Principios de diseño aplicados

- Interfaz limpia y orientada a tareas.
- Colores neutros para no interferir con imágenes de carne.
- Tabla de datos optimizada para grandes volúmenes.
- Contenedores JavaFX organizados mediante GridPane, VBox y HBox.
- Se prioriza la accesibilidad y claridad visual.

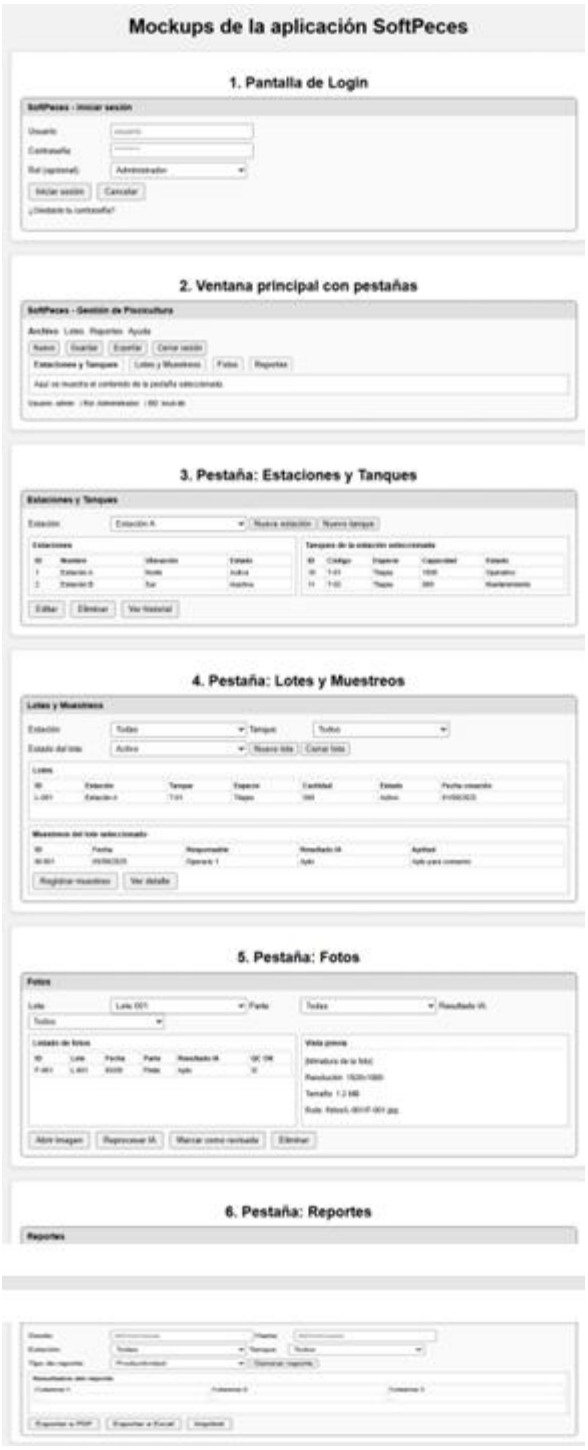


Figura 9. Mockups de la Aplicación

C. Desarrollo de Software

La fase de desarrollo del sistema **SoftPeces** se ejecutó con un enfoque incremental, aplicando buenas prácticas de ingeniería de software, arquitectura multicapa y metodología SCRUM. Se construyeron de manera progresiva los componentes que permiten gestionar la operación piscícola y preparar la integración del modelo de Inteligencia Artificial basado en ONNX para el análisis automatizado de imágenes.

El desarrollo se organizó en dos partes fundamentales:

C.1. Backend: Constituye la base lógica del sistema, implementando los **servicios, reglas de negocio, repositorios, manejo de imágenes y comunicación con el modelo IA**. Se desarrolló principalmente en Java, estructurado según la arquitectura en capas definida en el diseño.

El backend da soporte a los requerimientos funcionales RF01–RF10 y se diseñó para alinearse con los requerimientos no funcionales RNF01–RNF06. En la versión actual, algunos de ellos se encuentran parcialmente implementados (por ejemplo, el registro detallado de tamaño de muestra y método en los muestreos, la trazabilidad completa por usuario y las métricas de rendimiento y disponibilidad aún están en proceso de refinamiento).

1. Estructura del backend

El desarrollo se organizó en los siguientes paquetes:

- **domain/** – Entidades del dominio (Estación, Tanque, Lote, Muestreo, Foto, ResultadoIA, Usuario).
- **application/** – Casos de uso (UseCases) para gestionar operaciones de negocio.
- **infrastructure/**
    - Persistencia en SQLite.
    - Controladores de almacenamiento de fotografías.
    - Conector con IA (interfaz ONNX Runtime o API externa).
    - Auditoría y seguridad.
- **shared/** – Utilidades comunes (fechas, validaciones, excepciones).

2. Implementación de reglas de negocio

Se desarrollaron validaciones clave tales como:

- Validar capacidad de tanques al registrar lotes.
- Evitar editar tanques con lotes activos.
- Validar tamaño de muestra en muestreos.
- Solo permitir muestrear lotes en estado *"Listo para muestreo"*.
- Evitar registro de fotos sin metadatos completos.
- Estructurar datos para procesamiento IA.

Estas reglas garantizan la integridad del proceso piscícola.

3. Persistencia y modelo de datos

Se implementaron repositorios para:

- Estaciones
- Tanques
- Lotes
- Muestreos
- Fotografías
- Resultados IA
- Usuarios y roles
- Auditoría

Cada repositorio incluye operaciones CRUD y operaciones especializadas según el caso de uso.

La persistencia se implementó sobre **SQLite**, garantizando:

- manejo de transacciones,
- integridad referencial,
- consultas eficientes,
- almacenamiento seguro de imágenes mediante rutas locales.

4. Módulo IA (preparado para Sprint 2)

Aunque el análisis IA se integra completamente en el Sprint 2, se desarrollaron:

- Interfaces para enviar bytes o rutas del archivo al modelo ONNX.
- Adaptadores para recibir puntajes, clase y confianza.
- Manejo de tiempos de espera y reintentos.
- Registro de errores y fallos IA.

Esto permite conectar fácilmente el motor IA sin alterar la lógica general del sistema.

5. Servicios para reportes y auditoría

Se desarrollaron servicios que permiten:

- Consultar históricos de muestreos.
- Visualizar evidencia fotográfica.
- Exportar datos (a implementar en Sprint 2).
- Mantener trazabilidad completa de decisiones del inspector.

C.2. Frontend: Fue implementado utilizando **JavaFX**, siguiendo un diseño modular y orientado a la experiencia del usuario. Este módulo se encarga de mostrar las pantallas, validar datos e interactuar con la capa de aplicación para ejecutar cada caso de uso.

El frontend cumple roles críticos como:

- Interacción directa con el usuario.
- Presentación de información en tiempo real.
- Validación previa a la persistencia.
- Manejo claro de errores y estados.
- Visualización de fotografías y resultados IA.

1. Estructura general del Frontend (JavaFX)

Se diseñaron las siguientes pantallas principales:

- **Pantalla de Login** – Ingreso por rol, validación de credenciales.
- **Panel Administrador** – Gestión de estaciones, tanques, usuarios y reportes.

- **Panel Operario** – Gestión de lotes y registro de información productiva.
- **Panel Inspector de Calidad** – Registro de muestreos, carga de fotos y evaluación del lote.
- **Vista de Reportes** – Filtros por fechas, especie, lote, estación y calidad.

Cada pantalla se construyó empleando:

- **FXML** para estructura visual.
- **Controladores JavaFX** para manejar eventos.
- **Servicios** para interacción con la capa de aplicación.

---

2. Validaciones en interfaz

El frontend incluye validaciones locales antes de enviar datos al backend:

- Campos obligatorios.
- Rangos numéricos válidos.
- Formato de imagen compatible.
- Calidad visual mínima (foto borrosa/no apta).
- Estados válidos del lote para permitir procesos.

Esto evita errores y mejora la experiencia del usuario.

---

3. Visualización de fotografías y resultados IA

La interfaz del Inspector permite:

- Mostrar miniaturas de fotos cargadas.
- Indicar si la foto ya fue procesada por IA o está pendiente.
- Resaltar errores o advertencias.
- Permitir reintentos para fotos no procesadas.

La visualización final de IA se presenta en paneles con:

- Puntaje por criterio.
- Clase asignada.
- Nivel de confianza.
- Evidencia original ampliada.

---

4. Navegación y experiencia de usuario

Se utilizan componentes JavaFX como:

- **VBox, HBox, GridPane, TableView, ListView, TabPane**
- Botones estilizados y tarjetas informativas.
- Menú lateral persistente para navegación por módulos.
- Colores neutros para evitar interferencias con imágenes de carne.

La interfaz prioriza claridad, rapidez y facilidad de uso para operarios y técnicos del sector piscícola.

D.Pruebas

El proceso de pruebas del sistema **SoftPeces** se realizó siguiendo una estrategia incremental orientada a validar la funcionalidad implementada durante el Sprint 1 y preparar la infraestructura para las pruebas especializadas del módulo de Inteligencia Artificial en el Sprint 2.

Las pruebas se diseñaron para cubrir cuatro dimensiones fundamentales:

1. **Pruebas unitarias** sobre la lógica interna de cada módulo.
2. **Pruebas de integración**, verificando la comunicación entre backend, base de datos y vista en JavaFX.
3. **Pruebas funcionales**, asegurando el cumplimiento de los requerimientos RF01–RF10.
4. **Pruebas de calidad para imágenes**, evaluando restricciones, metadatos y preparación de datos para IA.

La estrategia de pruebas reproducida en esta sección se basó en la misma línea metodológica presentada en el *Primer Informe del Sprint* del repositorio original del proyecto.

E. Guía de instalación y ejecución

Esta sección describe los pasos necesarios para instalar y ejecutar el sistema SoftPeces en un entorno local de desarrollo o pruebas.

E.1 Requisitos del sistema

• Sistema operativo: Windows 10 o superior.

• Java Development Kit (JDK): versión 21 o superior.

• Herramienta de construcción: Gradle (incluido mediante el wrapper del proyecto).

• Librerías JavaFX: versión 21.x (configuradas en build.gradle).

• Conexión a Internet (recomendada) para la descarga de dependencias y, en caso de usar IA remota, para el consumo de la API externa.

• Cliente de correo electrónico o cuenta Gmail para el envío de correos de recuperación de contraseña (configurable en app.properties).

E.2 Estructura de carpetas relevante

• /src/main/java: código fuente principal (controladores JavaFX, casos de uso, repositorios, servicios).

• /src/main/resources: archivos FXML, recursos gráficos y plantillas.

• /data:

– softpeces.db: base de datos SQLite utilizada por la aplicación.

– app.properties: archivo de configuración (parámetros de IA, correo, rutas).

– models/: carpeta sugerida para almacenar el modelo ONNX y archivos de clases de IA.

E.3 Configuración de app.properties

El archivo data/app.properties centraliza la configuración del sistema. Entre los parámetros más importantes se encuentran:

• Parámetros de IA:

– ai.onnx.path: ruta al archivo del modelo ONNX.

– ai.classes.path: ruta al archivo de clases/etiquetas.

– ai.img_size, ai.confidence, ai.iou: parámetros de inferencia.

• Parámetros de correo:

– mail.enabled: true/false para activar o desactivar el envío de correos.

– mail.smtp.host, mail.smtp.port: host y puerto del servidor SMTP.

– mail.smtp.username, mail.smtp.password: credenciales de la cuenta de correo.

– mail.smtp.starttls.enable, mail.smtp.ssl.enable, mail.smtp.auth: parámetros de seguridad del protocolo.

– mail.from: dirección de correo que aparecerá como remitente en los mensajes de recuperación de contraseña.

Es importante mantener este archivo fuera del control de versiones público (por ejemplo, agregándolo al .gitignore) para evitar exponer credenciales sensibles.

E.4 Pasos para ejecutar el sistema

1. Clonar el repositorio del proyecto SoftPeces en el equipo local.

2. Verificar que se cuenta con JDK 21 instalado y configurado en la variable de entorno PATH.

3. Revisar y ajustar el archivo data/app.properties según la configuración local:

• Rutas al modelo ONNX y archivo de clases.

• Configuración SMTP (en caso de habilitar recuperación de contraseña por correo).

4. Abrir una terminal en la carpeta raíz del proyecto y ejecutar:

• En Windows:

./gradlew run

Esto descargará dependencias, compilará el proyecto y levantará la aplicación JavaFX.

5. Iniciar sesión con un usuario de prueba creado automáticamente (según los datos base cargados en la base de datos) y navegar por los módulos de estaciones, tanques, lotes, muestreos y análisis IA.

E.5 Consideraciones de despliegue

Para entornos productivos o demostraciones formales se recomienda:

• Configurar una cuenta de correo exclusiva para la funcionalidad de recuperación de contraseña.

• Proteger el archivo app.properties con permisos adecuados.

• Mantener copias de seguridad periódicas del archivo data/softpeces.db.

• Documentar los modelos de IA utilizados (versión, fecha de entrenamiento, métricas de precisión) para garantizar trazabilidad en auditorías.

REFERENCIAS

[1] Sommerville, I. (2011). Ingeniería del software (9.ª ed.). Pearson Educación.

[2] Nobody-clkafmG. (2024). *Código fuente del proyecto SoftPeces (Java, JavaFX, SQLite, ONNX Runtime)* [Código fuente]. GitHub. https://github.com/Nobody-clkafmG/Proyecto_de_Softpeces