Gestión de Proceso de Trabajo de Grado

Universidad del Cauca

Vigilada Mineducación

Ingeniería de Software II

Sistema para gestionar trabajos de grado en la FIET. Se desarrolla en tres fases: monolítica, microservicios y finalmente una arquitectura hexagonal con seguridad. (Tercer entrega)

Presentado por:

DAVID SANTIAGO ARIAS NARVAEZ

EDIER FABIAN DORADO MAGON

Juan Fernando Portilla Collazos

JHERSON ANDRÉS CASTRO MARTÍNEZ

ELKIN ADRIAN HOYOS GOMEZ

Profesor:

Wilson Pantoja Yepez

BRAYAN DANIEL PERDOMO URBANO

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería de Sistemas

Popayán, diciembre de 2025

CONTENIDO

Introducción

El presente documento constituye el tercer y último entregable del proyecto de curso "Gestión del Proceso de Trabajo de Grado" de la Facultad de Ingeniería Electrónica y Telecomunicaciones (FIET) de la Universidad del Cauca. Este entregable evidencia la evolución del sistema a través de tres fases arquitectónicas: una base monolítica, una migración a microservicios, y finalmente, la implementación de una arquitectura hexagonal (puertos y adaptadores) junto con un esquema robusto de seguridad basado en JWT, cumpliendo con los objetivos de mayor modularidad, mantenibilidad y protección de los datos.

En esta fase, se abordó un nuevo requisito funcional (la delegación de evaluadores de anteproyectos por parte del jefe de departamento, con notificaciones por correo) y se realizaron dos mejoras arquitectónicas fundamentales:

1. Arquitectura Hexagonal y DDD: Se refactorizó un microservicio de alto dominio para adoptar la arquitectura hexagonal. Esto permite aislar la lógica de negocio central (dominio) de los detalles externos como frameworks, bases de datos y APIs, facilitando las pruebas, el mantenimiento y la evolución futura. El diseño se guió por los principios de Domain-Driven Design (DDD) para alinear el software con la complejidad del negocio.

2. Autenticación y Autorización Seguras: Se implementó un sistema completo de seguridad utilizando tokens JWT. Se definieron los roles de usuario (estudiante, docente, coordinador, jefe de departamento) y se protegió el acceso a los endpoints de la API, garantizando que cada actor sólo pueda realizar las acciones permitidas dentro del sistema.

El sistema continúa desarrollándose con Java y Spring Boot, utilizando Swagger para la documentación interactiva de las APIs y manteniendo la comunicación asíncrona mediante eventos. La documentación que aquí se presenta detalla esta evolución, incluyendo los nuevos diagramas de arquitectura C4, la explicación de los patrones de diseño aplicados, la estrategia de seguridad y la evidencia de la gestión del proyecto. Este documento, junto con el repositorio de código, sirve como evidencia del proceso de ingeniería de software aplicado.

Capítulo 1

1. Planteamiento del problema.

El proceso de gestión de los trabajos de grado en la Facultad de Ingeniería Electrónica y Telecomunicaciones (FIET) de la Universidad del Cauca constituye

un componente esencial dentro de la formación académica, pues representa la culminación de la trayectoria universitaria de los estudiantes y un espacio de validación de competencias profesionales e investigativas. Sin embargo, en la actualidad este proceso presenta múltiples limitaciones derivadas de su carácter manual, fragmentado y poco estandarizado, lo que justifica el desarrollo de una solución tecnológica integral.

Limitaciones del Proceso Actual:
En primer lugar, la gestión documental se realiza principalmente mediante el intercambio de correos electrónicos y el envío de formatos en PDF, lo que conlleva problemas de desorganización, pérdida de versiones y duplicación de archivos. La ausencia de un repositorio único dificulta la consulta de documentos actualizados y genera incertidumbre tanto para estudiantes como para docentes en relación con el estado de los trámites. En segundo lugar, la comunicación entre actores (estudiantes, directores, jurados, comités y secretaría) carece de un canal centralizado y depende de intercambios manuales. Esto ocasiona retrasos en revisiones y aprobaciones, aumentando la carga administrativa y retrasando los tiempos establecidos para la culminación de cada etapa del trabajo de grado.

Otro aspecto crítico es la falta de trazabilidad: actualmente no existe una herramienta que permita identificar con claridad en qué fase se encuentra cada trabajo de grado, qué documentos se han aprobado y qué evaluaciones están pendientes. Esta carencia limita la supervisión por parte del coordinador del programa y dificulta la planificación de recursos académicos (como asignación de jurados o programación de sustentaciones). De igual forma, la ausencia de un historial sistematizado de acciones reduce la transparencia y la capacidad de auditoría del proceso. No quedan registros centralizados de observaciones, evaluaciones o decisiones tomadas, lo que incrementa el riesgo de errores, incongruencias o incluso disputas entre actores.

Propuesta de Solución y su Evolución:
Para superar estas limitaciones, se ha desarrollado de manera evolutiva un Sistema de Gestión de Trabajos de Grado. Las dos primeras fases sentaron las bases funcionales y una arquitectura distribuida (monolítica → microservicios). Esta tercera y última fase se centra en consolidar una solución robusta, mantenible y segura. La implementación de la Arquitectura Hexagonal (Puertos y Adaptadores) en el núcleo del sistema garantiza que la lógica de negocio donde residen las reglas críticas de evaluación, flujo y estados esté aislada y protegida de cambios en frameworks o infraestructura, directamente abordando los problemas de trazabilidad y consistencia en las reglas de proceso. Además, la incorporación de un esquema de Autenticación y Autorización basado en JWT resuelve de manera estandarizada y segura la necesidad de roles y permisos bien definidos, estableciendo los canales de acceso confiables que el proceso manual carecía. Finalmente, la adición de nuevos requisitos funcionales, como la delegación formal de evaluadores con notificaciones automatizadas, ataca directamente los cuellos de botella en la comunicación y asignación de recursos. Este documento presenta el diseño, la implementación y la documentación de esta solución tecnológica.

2. Historias de Usuario (Primera y Segunda Iteración) y Épicas.

En concordancia con la metodología ágil adoptada para este proyecto, se construyó y evolucionó un conjunto de épicas e historias de usuario de alto valor para el cliente, alineadas con los requisitos funcionales definidos a lo largo de las tres iteraciones del proyecto (monolítica, microservicios y arquitectura hexagonal con seguridad).

Referencias:

Para garantizar la trazabilidad total del proyecto, toda la planificación ágil se gestiona en los siguientes documentos colaborativos, que han sido actualizados en cada fase:

Épica: [HE(Epicas)-SWIII].

Historias de usuario detalladas: [HU'S(Historias de Usuario)-SWIII].

Backlog y tareas: [Product Backlog-SWIII].

Burndown chart: [Burndown chart]

Backlog y tareas Jira (herramienta de mejor nivel de usabilidad): [https://unicauca-team-dag79f44.atlassian.net/jira/software/projects/KAN/boards/1?atlOrigin=eyJpIjoiOTMwZTcyZTVjYzhlNGFmY2JjYTYyZTA1M2VmYmFjY2YiLCJwIjoiaiJ9]

Estos enlaces constituyen la fuente única de verdad para el seguimiento del proyecto, permitiendo garantizar la trazabilidad entre los requisitos de alto nivel (épicas), las historias de usuario priorizadas en cada iteración y su respectiva gestión ágil en el backlog. De este modo, se evidencia que la implementación de las funcionalidades a lo largo de las tres entregas se encuentra documentada y gestionada de manera colaborativa.

2.1. Requisitos Funcionales.

De acuerdo con las historias de usuario, el sistema implementa requisitos funcionales de alto valor para el cliente:

Registro de Docente (HU-01).

- Interfaz de registro con campos básicos: nombres, apellidos, celular (opcional), programa, correo institucional y contraseña.
- Validación de correo institucional y reglas de seguridad de la contraseña.
- Servicio en la capa lógica encargado de registrar los datos.
- Persistencia en la base de datos de la información registrada.
- Validaciones en frontend para garantizar que los errores se detecten antes de guardar.
- Pruebas unitarias de servicio y persistencia para asegurar la integridad del registro.

Subida del Formato A (HU-02).

- Formulario con campos obligatorios: título, modalidad, fecha, director, codirector, objetivos, archivo PDF.
- Validación de obligatoriedad de campos en frontend.
- Subida de archivo PDF válido y organización en carpeta específica.
- Validación de carta de aceptación en ambas modalidades.
- Persistencia de la información del Formato A en base de datos.
- Pruebas unitarias en frontend y persistencia para garantizar correcto funcionamiento.

Evaluación del Formato A por Coordinador (HU-03).

- Vista con listado de proyectos pendientes de evaluación.
- Lógica de negocio para aprobar, rechazar o dejar observaciones.
- Campo para observaciones textuales.
- Guardado de resultados de evaluación en base de datos.
- Notificación simulada a docentes y estudiantes mediante logger.
- Pruebas unitarias para verificar aprobación, rechazo y observaciones.

Reenvío de nuevas versiones del Formato A (HU-04).

- Adaptación del formulario para subir múltiples versiones de un mismo proyecto.
- Control del número de intentos (máximo 3).
- Rechazo definitivo del proyecto después de un tercer intento fallido.
- Pruebas unitarias para validar el conteo de intentos.

Consulta de estado por Estudiante (HU-05).

- Pantalla para visualizar el estado del proyecto.
- Lógica de negocio que consulte el estado en la base de datos.
- Estados posibles: primera, segunda, tercera evaluación, aceptado, rechazado.
- Conexión con base de datos para información actualizada.

Subida del Anteproyecto (HU-06).

- Lógica para que el docente suba el anteproyecto tras aceptación del Formato A.
- Validación de archivo PDF y campos obligatorios (incluye fecha automática).
- Persistencia del anteproyecto en la base de datos.
- Emisión de evento asíncrono para notificar al jefe de departamento.
- Pruebas unitarias para asegurar el flujo correcto y el almacenamiento del archivo.

Consulta de Anteproyectos por Jefatura (HU-07).

- Lógica que permite al jefe de departamento listar todos los anteproyectos enviados.
- Filtro por estado o fecha de envío.
- Consulta la base de datos para obtener los registros.
- Preparación para posterior asignación de evaluadores (proceso futuro).
- Pruebas unitarias para validar consulta y presentación de datos.

Delegación de Evaluadores de Anteproyecto (HU-08).

- API que permite al jefe de departamento designar dos docentes evaluadores para un anteproyecto específico.
- Interfaz que consume dicha API, presentando un listado de docentes disponibles para su selección.
- Validación de reglas de negocio básicas (selección de dos docentes por proyecto).
- Persistencia de la asignación de los evaluadores en el registro del proyecto de grado correspondiente.
- Activación de un flujo de notificación asíncrona para informar a los docentes designados, simulando el envío de un correo electrónico.
- Pruebas unitarias para garantizar la correcta asignación y el disparo del evento de notificación.

2.2. Requisitos No Funcionales.

El sistema, evolucionado a través de tres fases arquitectónicas (monolítica, microservicios y hexagonal con seguridad), debe cumplir atributos de calidad que garantizan escalabilidad, mantenibilidad, seguridad y una comunicación eficiente en una arquitectura distribuida y desacoplada.

Patrones de Diseño y Arquitectura

- Domain-Driven Design (DDD): Para delimitar los contextos de negocio y guiar la descomposición en microservicios.
- Arquitectura Hexagonal: Implementada para desacoplar el dominio de los detalles externos.
- Event-Driven Architecture: Para la comunicación asíncrona y desacoplada entre servicios mediante eventos de dominio.
- API Gateway: Patrón *Facade* para centralizar y simplificar el acceso a los microservicios desde el cliente.

Principios de Diseño SOLID

- Single Responsibility: Cada clase y servicio tiene una única responsabilidad bien definida.
- Open/Closed: El diseño permite extender el comportamiento (nuevas reglas, adaptadores) sin modificar el código existente.
- Liskov Substitution: Las jerarquías de herencia (ej., tipos de usuario) son correctamente intercambiables.
- Interface Segregation: Las interfaces son específicas y no obligan a los clientes a depender de métodos que no usan.
- Dependency Inversion: Los módulos de alto nivel (lógica de negocio) no dependen de los de bajo nivel (infraestructura), sino de abstracciones.

Patrones GoF Aplicados

- Adapter: Fundamental en la arquitectura hexagonal para adaptar el núcleo de dominio a infraestructuras específicas (APIs REST, bases de datos, clientes de mensajería).
- Strategy: Utilizado para reglas dinámicas (ej., evaluación de formatos, algoritmos de validación) y potencialmente en la lógica de autenticación/autorización.
- Observer / Event Listener: Para la publicación y suscripción de eventos en RabbitMQ.
- Facade: Implementado en el API Gateway.
- Repository: Para el acceso y encapsulación de la persistencia en cada servicio.
- Factory Method: Para la creación de objetos complejos (ej., usuarios según rol).
- DTO (Data Transfer Object): Para el transporte estructurado de datos entre capas y servicios.

Capítulo 2

1. Escenario de Calidad Escalabilidad.

Contexto: El sistema de Gestión de Trabajos de Grado es utilizado por la comunidad académica de la FIET (estudiantes, docentes, coordinadores y jefes de departamento) a través del DesktopClient JavaFX, que consume los microservicios de usuarios, proyectos, documentos, mensajería y notificaciones. Se anticipa un pico de carga significativo durante los períodos de cierre de convocatorias (p. ej., últimas dos semanas de cada semestre), cuando cientos de usuarios concurrentes intentan iniciar sesión, subir Formatos A, subir Anteproyectos, consultar estados y, en el caso del Jefe de Departamento, asignar evaluadores a los anteproyectos.

Estímulo: Se produce un aumento brusco y sostenido en la cantidad de solicitudes HTTP por minuto dirigidas principalmente a:

- los servicios de subida de documentos (Formato A, Anteproyecto)
- los casos de uso del microservicio de proyectos (creación de proyecto cambios de estado, asignación de evaluadores)
- las consultas de estado desde el DesktopClient,

debido al cierre inminente de una convocatoria oficial de trabajos de grado.

Respuesta del Sistema:

1. Escalado Horizontal (Microservicios): Los microservicios más exigidos (principalmente project-microservice y document-microservice) pueden desplegarse en múltiples réplicas/instancias de forma independiente, distribuyendo la carga entrante. Al usar JWT, los servicios se mantienen stateless (no guardan sesión en memoria), lo que facilita añadir o quitar instancias sin afectar la lógica de autenticación.

2. Colas de Mensajería (RabbitMQ): Las operaciones que disparan notificaciones (subida y evaluación de Formato A, subida de Anteproyecto, asignación de evaluadores) se manejan de forma asíncrona. El project-microservice publica eventos de dominio en RabbitMQ (formatoA.subido, anteproyecto.subido, evaluadores.asignados), responde inmediatamente al DesktopClient y deja que el notification-microservice procese los mensajes a su propio ritmo, evitando que el envío de correos/notificaciones (simulado) se convierta en cuello de botella para las operaciones críticas.

3. Balanceo de Carga: Un balanceador de carga (o el propio API Gateway en una arquitectura desplegada en nube) distribuye las solicitudes HTTP que provienen del DesktopClient entre las distintas réplicas de un mismo microservicio (por ejemplo, varias instancias de project-microservice para gestionar proyectos y anteproyectos).

4. Base de Datos (Database per Service): Cada microservicio mantiene su propia base de datos (usuarios, proyectos, documentos, notificaciones, mensajería). De esta manera, una alta concurrencia sobre la base de datos de proyectos o documentos no bloquea ni degrada la base de datos de usuarios (login/JWT) ni la de notificaciones, manteniendo el impacto aislado por bounded context.

5. Separación de Responsabilidades en el Cliente: El DesktopClient concentra la interacción de los usuarios pero delega toda la lógica de negocio y persistencia a los microservicios. Al no compartir estado entre clientes y usar solo tokens JWT, se pueden tener muchos clientes concurrentes sin que el servidor tenga que mantener sesiones por usuario.

Métrica de Calidad (Medición):

- Tiempo de Respuesta (Latencia): El percentil 95 (P95) del tiempo de respuesta para las operaciones críticas:
    - subida de documentos (POST /api/v1/proyectos/{id}/formatoA, POST /api/v1/proyectos/{id}/anteproyecto),
    - y consulta de estado (GET /api/v1/proyectos/{id}/estado, GET /api/v1/proyectos/estudiante/{email}),
    - debe mantenerse por debajo de 2 segundos durante el pico de carga.

- Disponibilidad: El sistema debe mantener una disponibilidad superior al 99% durante los períodos de alta demanda, sin caídas totales de los microservicios críticos (usuarios, proyectos, documentos).

- Throughput: El conjunto de microservicios debe ser capaz de manejar al menos un 200% de incremento en la tasa de solicitudes por segundo (RPS) con respecto a la carga promedio, sin degradación severa del servicio. Las colas de RabbitMQ pueden crecer temporalmente, pero deben drenarse a un nivel normal en un tiempo acotado una vez disminuya el pico.

Resultado Esperado: El sistema absorberá el pico de carga manteniendo un rendimiento aceptable para los usuarios finales del DesktopClient. Las operaciones síncronas (login con JWT, creación de proyecto, subida de Formato A y Anteproyecto, consulta de estados, asignación de evaluadores) se completarán en tiempos razonables, mientras que las operaciones asíncronas (notificaciones a coordinador, jefe y docentes evaluadores) se encolarán y procesarán de manera confiable a través de RabbitMQ. De esta forma se garantiza la integridad del proceso académico y la continuidad del servicio durante los períodos críticos de cierre de convocatoria, sin tiempos de inactividad ni caídas globales del sistema.

2. Diagrama de Bounded Context.

El sistema, modelado bajo Domain-Driven Design (DDD), se estructura en Contextos Delimitados (Bounded Contexts) que encapsulan subdominios específicos y se materializan como microservicios independientes. La comunicación entre ellos combina APIs REST síncronas para operaciones que requieren consistencia inmediata (registro, consulta, autenticación) con mensajería asíncrona mediante RabbitMQ para orquestar procesos de larga duración y notificaciones (como el envío de un Formato A o la asignación de evaluadores). Este enfoque asegura un bajo acoplamiento, alta cohesión y mayor resiliencia del sistema. El diagrama adjunto ilustra estos contextos y sus interacciones.

Enlace(draw.io): BoundedContext tercer corte.drawio

3. Arquitectura y diseño (C4 + UML).

La aplicación se implementa con una arquitectura basada en microservicios, los cuales se comunican tanto de forma síncrona mediante APIs REST como de forma asíncrona a través de RabbitMQ para el envío de eventos y notificaciones. Cada servicio maneja su propia lógica, almacenamiento y modelo de datos, aplicando principios SOLID y patrones de diseño para garantizar escalabilidad, separación de responsabilidades y mantenibilidad. Esta estructura permite gestionar usuarios, proyectos, documentos, evaluaciones y notificaciones de manera independiente pero coordinada, y con base en ello se construyeron los respectivos diagramas de arquitectura y comunicación.

A. Diagrama de Contexto (C1).

Propósito: Representar el sistema de Gestión del Proceso de Trabajo de Grado como una plataforma distribuida basada en microservicios, junto con sus actores principales (Docente, Estudiante, Coordinador y Jefe de Departamento) y las interacciones esenciales como registro, autenticación, envío de Formato A, evaluación y consulta de estado del proyecto.

Decisiones clave.
El sistema no opera como un monolito, sino como un conjunto de microservicios independientes que colaboran entre sí. La comunicación directa entre usuarios y el sistema se realiza a través de un API Gateway, el cual coordina las solicitudes hacia los servicios correspondientes. Además, los eventos importantes del proceso académico (como envío del Formato A o resultados de evaluación) se gestionan mediante mensajería asíncrona con RabbitMQ, permitiendo notificaciones desacopladas y confiables. Las notificaciones se simulan mediante logs, dejando abierta la posibilidad futura de integrar un servicio real de correo electrónico u otro canal de mensajería.

Enlace(PlantUML Web):

https://www.plantuml.com/plantuml/uml/ZPBDRXD13CVlUGgJKolHviA9KqCQ8g4YLHIuLiwCkvrgThpCHsX0F18dRbppOjYxscJQ2C5bPNI7F__zSP4onbnQFtdXWl754jpdtANNird4h-UDo_TbNH95mo5JoEU6srddCMZEMShMk_LiyUhiMaEVtb_DMaoPeeHk5vhncCyxfK-kAIOE9vQDKEWbJ2y7QtemuKiXw28bSH2sq53AZWF2MuujPaumb-XSNdQHjsGaCZqTmVAsM8S3UxbtfhKtL4YkaIzva1Iq35RC4NZW6UPeNP07KH2BWoVluJfo4x75dhv1I_kz5zbhjFI0m6iPsmQDj42Lq7TqHvVKOSpOIgCi20qfjWuh-hDB1RtxXe1rqr80fOc0OCKXgLzKM5zaShEL-RUtRxW4Yt5xafpwABIRmOALZdNtAsZ5Zn7N-C19xLL4AmE0xnE0WJ8chrQIhcaSZpDKwBpppg3fgT82GgH6KgCqbyhQMTtceDECbQjn7QtqnxwDvMC-4G7EtyWedT7T3fPZXIpr5uZCjdYHSgfQhhXfv01JzInmTdUZG1tI1_A7IwoQ5FWeNIRcWW0G4gLUz5PxgONp4-72LMnzXpLvmI4Ld_4lLoUfJq-kpfqRGz86Ul1ko9PnFZ-x0N9yNlyWMVhFum84WnQXX55I1NwsesNOx7vYDNSEb0vx0iC1VB-dKs5TKB2bzNy0

B. Diagrama de Contenedores (C2).

El propósito de este diagrama de contenedores (C2) es mostrar la estructura interna y las interacciones del Microservicio de Gestión de Proyectos, el cual implementa la Arquitectura Hexagonal (Puertos y Adaptadores). Este nivel detalla los componentes de despliegue que conforman el servicio, su aislamiento de responsabilidades y los mecanismos de comunicación con el exterior.

El contenedor principal es la aplicación Spring Boot que encapsula el núcleo de negocio. Este se comunica de forma síncrona a través de su API REST (adaptador primario), recibiendo peticiones del API Gateway para operaciones como la creación de proyectos, consulta de estados o asignación de evaluadores. Internamente, la lógica se organiza en tres capas principales: una capa de adaptadores que maneja infraestructura (REST, persistencia), una capa de puertos que define interfaces de negocio, y un núcleo de dominio que contiene las entidades y reglas de negocio puras.

Enlace(PlantUML Web):

https://www.plantuml.com/plantuml/png/hLXHRzis47xth-2y0Kk2fGg68KE6jQXYAvcBn74jfLsVWfDqLXX8eaTIxR9rlsA_eqzxwwl_s8wKP5cM0g50yn2Jqjtt7O_7xsY_L1gaBh9qz0tFexI8aTrhlLQ_E8w43yyIhk-Bi50e8v5hpFMpI6JEEeNS-1mh7QSyT2O_7I_CezIhAoS3fL7Iexi9UG3FKJvR6_oHvZf5DfvoI2Ham8XgOa4n5X8LEsO-9rzwSubAyyr_EGiaXF0WbB6zb12ByMZqFNk-zySMtjA_cRj-vyreWLA9_20M4HBJ3yHUZiOqz8iGsOMG6MXYSDaZS-dDMef7ZEZ9-B3sfcYAc4C9u6qd1eCMe8fK0oCR2i_4sGCG2I5ZdfE1D3xDp41uxo7TV8Ow4kOsVW-uCfIlwCCWJt4DKaD62n36qrKyoOcQ00g3XaJPcnz_vWVUjTjDZ_ze4dvtBeey1IbueCeD8EnwAs9iTYF6b2qa9L2PXTQRizqRTd0nyuB3yI7xUyJep_xhYSPTNCqcxcIs-NVEtdhd7GFhjosUWvMqHH9JK1VLaBt5q2OWJueKfBD4YBHvCAFNSWL_cPWALO3aWcq-vpo2yU50J3DswGRUM_VTK3omvdS9QFm0ZyHe7SiPSzXI51I0BQwrv7d29gaeujh0xdOk2mqPDtj85QSmAIIF8Hw8xtesMTxutIBDR3AxyOT2DEVrBbD4PqR76O-aeC5x7k5ERET2sAGr-xjDcgBGe33ddL8NcPSKNvk3oIm18rtIL9ClOQhAlNFkEaInY0eg-OgfcWrIkMa64UPrfbCHGMfeGKRt_BrGR36zw53bGlCLZq1paPTqkq-6AEULRKIsg1YepQUS779Gx20LIOBoiCEOeLAG44f9jvqESTb2pp3cjiHZ914mwoIcIAHmE51DvwxI-MpgiQaR35RJD2phAGw9zvP6xBSVsVJSH646e244XRivtwqYAL8ZInt8kdmi

wgAS30FNHKB8bUBsO3SLOy6drMmOFKD215sscngabw5LA9PbllDac2blbKgN
eAcB4lswdWv3ruL0VPmhBII7TA0QPlEBfUlvmV9s4jmktK4zvYagspTCShd4oB1K
HspSRXtrQMj5I2UFmikPqwJOOOAceC5vueJWtHy55cYxPmfDmLla9OGXrzUlcS
DydXKfn32sISiLFA3SV8ATmqW4__GsoQLtvPAupZsVkPFWPacVno-URfrBJ9jhn
KwFkbN0C6LL9xKx5GGBdsYDHtELQ9owrmYrvzQwG3H-dRk3Okw-fxb2D2xzbu
R6PlWnitdWBNilMeQhOMdto0LgNecYuI50FzWZtMsZbFNhHlsuK1Qy1TaqEAVM
6kUeXdGi_c7NgzModAsM37koTW_QBTnXtsvJSTh7luj12MbdmEJa71IMoWIwoi
as3Or0No4wdeea3z6WlfgUJwgWTbTVZOTSMikk9eDEdVMs7Wmuj_jXCtlQpIHh
8h8YhsygZ6AKoB9M0ZiPQE-eNuHSCeTDKKKW4v16fBexsuROwzsB8amvUJV
Ar7LlkxlahK4mscYwk1NDwzS4KEbRVuAxE5PDiaOYkpWcQUqRqgvQT_U-fVcN
1SWOjjQby6VrjweBCcM-jJMdkkJQknerDuJ-1MrrbPOY9Of6laLpbJBobm5FcIerN
poLdafplnvWIus_2g9Hwl-7wXFh-Kr0ykpzRhzZ1RCtVLhjnZ554K1oC3RVzGdxEu
r_QkgiGZuVVtjoSdeQHOI_OnTNLuskxTdPoKdR5jHZ7kqRhbQdf_k6L5KTmBCp
OxUj6b0CrHgZpIUJBsMyjrS3Yn49QJ72f4R0qzNFgnKXT6pD8jgsgvFLQdNIPylp
bTmt9bCyVIBlLzuxRpxjkIUac60UCycJUprYx2EpSFGPyBLW7-sZNwkWNz2u9Vn
aBB4sgg8rHgONs5y4J2jetji0pVkPYHwLbaMa2mcLnT6HqQEZ8ngs5REkjyQGzA
BF4BgMHeMjfHlP7mQ0XLLmSHKSBRzCmcZqaluNMVe_

C. Diagrama de Componentes (C3).

El objetivo de este diagrama de componentes (C3) es detallar la estructura interna del Microservicio de Gestión de Proyectos, evidenciando la aplicación concreta de la Arquitectura Hexagonal (Puertos y Adaptadores). Este nivel desglosa los componentes de software principales, su organización en capas y las dependencias entre ellos, ilustrando la estricta separación entre el núcleo de dominio y los detalles de infraestructura.

Enlace(PlantUMLWeb):
//www.plantuml.com/plantuml/png/bLbFRnkx4R_lfo1Se2y16hd9-rCqI0rl9FbP3xQi
IdBQ0BbGkoEPoOhSabmtVdq1-bbww-6TSklLtwIVfBCSxgv8QMqt0UmbvpS_Se
QS6P8-DPPhMssBn7mIikIQRrccjgMI8Et2tXN0D6IMoqq1Ev0Srhmgx9cITigtm8vI
BNXnr24wBVQFXB6tFFkqqQgI-L0LIhFVhFzO_wybIkUWVUyVLIL_x9qJL4IGxEN
toPSaiS8MECv8y0tISvm16pRp1CD6O7bHy5on5socrKSSdLsAJ2i3-bPay44-ozG
0ycfGIP7nAkE3KgixX2dp_2X9IfmWt-009Ic_sEue7w4DEAbls9yE_6FfRC8kqlVZ-
K4nwdMkE79OVeTMFHkYxxGg2j0u0SQDuwbznRe5O4SpFyaE_K6Upi7OhiCfTy
sOOGIhQlEOUWH2ZYyFMBmOpzzDXkCdMxqWDpQMlwsCa60Ckr0RaP7n7cA
y1uIqeDSyG-L9um8FEUCPpt4zoO4yiqGmeVxO-XxbKzzsgjH98uVQPqflkLNfebfj
XRMrz_1JQEO5B2M26BRFbA91pORROziLEi8OtKCwlkL5nJEXv3cNkLjSweed6
04FJj28ZGovg4jxxLnfC8ujWWPiaTlVyKBaE0zaeq_DAeFp4ugeOXnHVN4_7zi8-r
k018_j0HS1fJ82XxiBPh6h7vEqJaXnOLREUYHgKv3Z2J6z37F8RlZFgb_T0nw9m
j7LPJgPFZa8HshBXMnYy5BbK1ZsMxQmt8BFG0JfJq8_QfxNqnvBAspWEX8T2
Y77psQyti4Ib-n7C5RSVvKCVwEA7zcXF2ZU0crg7fF6qQ5XndAcnHOqxxQwu_m
WISkDGfyI--ESrZPNVQfElgz-PRLg8EQ370m6JiTr-zvUeqRBAyEESUsHAJ8jyluS
_bQXqz27gCDGX2xdvBu8qc_WHAuLQdiE3QQiyPub0kspN5FeKgs5beUqOs6Z
_s1M5RbobYEH-wt2cRY-XvDfFyaEYkGF9T1Wn8YeGtcQ_jovaoeFCH4mPFhoI9
XFfcVpTB6SNm-Nr_EdrrnSHCsDrLLcAzsMtl7dEi9umOQ5m0duw8km-oLuggnO
YypD70pgaccp2dHZ6c58T321XXopQbK8S-FE8sdEI-iF8WxL2W-5mvolLi9U_faj
GEQkvCdR-ryvsu8q_AClKGGYH1nw-nkwjUZq3721gELuGa2iFb9PjSMl1mWYIE
IQo2GiPsknGHRwg3ItwAmTWwY_BprFrLXchY9YG7pEeBHKfDhl85UqlO_ilzbulf
WibkFf_t7Wcu4smbYGpQ6FHkvw-sjCLvhtwtJNTDevjtp53Ovt_egDV0EtvQ4ZF6tI

yrUZju-UROVNyydo_PEEjLcbcqBOD6BBEjW3gnj2OeAV_cxJojxWjyYSpfaeh8iS-d0K2ABM0Usb-WGI7NmhNC14nmu4D3Bd6_HEZs_6VrsEvzFqWYtUuuQuN1psaLGvEgT2nmShuEC9zEN2Gt8V6p6iYng4Oa6he0iHNDuj65ENpfL6ipGjR2Fsp99oZNFMFljqHueZz5ax7r-amydLTBnWi_baEfpCqejnZytVkDlRYvFs7fDqbods0llzzOIzPfNXITDwyyjWm5g9s9O5r6RpXFf8aoXRLMgIRdHoZKXmLfv7I4hxPfTu_zRI2OFYUGWGNoa28TMvFk7-cJ_Pcue3nhaJhJFL2jXiT9R4ytkQLZmqQOH97l6tJjpULD2HpT5OG-6sZRaHPU3Cu1oC59a6VimpMo7Lppm9nQZGd4KH2kwB2IaomOi-A9qz4MurbqPG5Tq71uTEHAkwIKEG8RHkQBf9eYKa_qMNcDfxWd5_BkvHYIykfAHTWxG2oaF-0eJS_uSdCTL3O387f0rMOlSWWjWc5Q2IsnH987T159qvWXYAP3X0SBP0CBZZHnBq4enI4aBAMZzpyzsfWKaKlAJLbZWVxrs7azSL32K_pTBZSx6gWzf2pTiUx33pRolfivj9ehsD53u78ydvSZd3BBTO9d7C184KOLkx41EbKkSDgs7JUHkdrXItA2Ls5Qlsa5OMDW9NjrGwQIM4w-fJ5vksBcacQMM4ZGefmhEcrdfgd8-xSLLr2Obc55-ZK5r39bOnBhueEPnLMWPvExXH-8pGbs7xC5sunOWuq_OZzkvbjB5wozxqQfakiDJFBYR3T3bvb_QMUGjCYysDPMhDsjUtDwkJWszkRuvN9oY-qiwrTOJZeWkPYP8N3FjUe_3bWBqBKozx5cxWvuXwDM2BEg4tnuSQ5MvWmdqxOEwnGKdk4o02W-U86lNTWBdmfWn1R3rNbHh-_O1TSWaV4Qsrqc3O7MLadX4w0PcpsZT9vA4WYNb7SQlhfn0tAU-W9dLtEGXRjNF--yz_iRvNZbQyW4qbyrxn4dJMgttm4IIIKLQEXDI9EsmjfB3YjXLUtl_Rgdhw3Do8-jdptp4DGjhQBcprEIjAoUIjdbKXlxMB-1speaJ_vT2ifzzP1ruTQ3FrF0BG3kfvQg2Dqt_lH_cruLqDo9HaMbLMo0tx17UzkoIglcJu3ZaPIwkv_o3vck5t1llUFV_6OBzMhnbzXCX9k-SaA-w_RbnGc1Ao-rzhnvDFoLcuRL7XBZYDeDm7McTUNvQv6ByVJqVfuTHIm0O9o2daWXJJXh2GsKfpTWwV-KP9NdXl_CBepqVq3rkyvFHD7GxY_-Ba8NYS4rAe208_hBzThpqaBN4_aQKHPWq_K0yzd43pSiARaTRhtxSGl1SS_mLMuKXlgfD9yxo2QrgTO1-zPLjbZcTAMuD8z05v8Zd57zMs-1y0

D. Diagrama de Clases UML (C4).

El propósito de este diagrama de código (C4) es mostrar la estructura interna a nivel de clases del núcleo del microservicio de Gestión de Proyectos, detallando la implementación concreta de la Arquitectura Hexagonal. Este diagrama desciende al nivel de archivos fuente para visualizar las entidades de dominio, objetos de valor, puertos (interfaces), casos de uso, adaptadores concretos y sus relaciones de dependencia.

Enlace(PlantUML Web):

https://www.plantuml.com/plantuml/png/XLLTRXit47xdAGRR1pzvNJjjqmI9OKMM4WEMB4jAqOU-ZBYKmeHBRlbZr2ZysWFq1Zb0dte4twGdwP3SzIgoubZ0OcVc6txV_71zvZnQ7sh5t2Uf6xHOGoNM69GV6-sdM0iOM8dRSIkuHxrH0lva06-GVzfO4tGrDCfO-6xz8lvYnDXAsDRxV1L_v9rdgDN2ml7tx8unBpsTLGnDtHWjj1UE92WOg_3HmC4YhAI5iR4rUWE3WuAnXYXnGocDDHz9p64jkJLEs1I9HP5KCTYIMGnc5t29jyBc803IkA2OMNDBI3EaSgrHIjZVzDbSEDyx2a07lKadt3sYM6JsRzAqk35oh0GnjNQYoCwKkvFTbZ-WPjJI-wYHNgbyBfn9kG-Hnyc_e98LPLDlVkMI8YwbTR6STfUUtv2UsjCO9odhDdFdmDw-d9iQfNwY8MyjLeRE76alVJwl2-thnCZvX1_fcPMri3YwGHLIIGkFFZSoWt99UrLTwBL5vstWFbYnHzrx5z1AuuPAaatona9kTBOInNidR3PtKyyD3pMzFP5CaFsvK-FbMl9sGhEmKj9zI1kPRqTtGC9zhQFzT1wFAffzHohqk48de7Xt0kUjaHM-YJVp05xJ7un-NOxcqy5bifWs5IK4gd_dthLTcInQIDKsuZ6CwYVKxq44WirnjP9-SfqYa-i7rIdeSZ2yk9gE5fctlu1mU7ZQNPQNK1oNSAcmne9bNmptYnqH9sMs2TBx8wgVOeGzws0Gsl4Nh4Tq2LHW1BzRBcUaSR4is1PlTnSYueSIXbOWDErS2fQ36HWd4L4_bZ0JraddHS6YCqQf_p7sKmbIIKY5EnpTNO-

OvsMq1NJTZg9x0C7xROlmdrlux7nSi3xIufugBIzTZBye8QqcHn0taR9WEKOecc2432RNi-9XbDEhvM3HROyNu4q3Pltb9_NLwdIagyP8xTtBLqUhK_BDhXPBEEhwvuxM-JDuY366zlvpBUW0sfgWl6pGUYPq1P62PI8hDnyyKVNha9csluCT6QrHT-j1_17deSbBgpCtlZZWnjgaWfoqA1DJfJCgkBd_75_87GTE8X2S1CfeI23tagSDM0j7RAXs5IgnZWBxJnQfo-yWjUGILQVlQryy1eJqSGm3xWCgU0t7MonARAAnCjwREf-V_xFg40-hJ3eVZk6_I_w6a_HybfvbMMPuN8SC6fewa2ZYllzNWxl_HtDBHMKOZNml37iS2Sbos1azGgt-1m00

Capítulo 3

1. Patrones de diseño GoF implementados.

A lo largo de las tres iteraciones del proyecto, se han aplicado los siguientes patrones de diseño de la "Gang of Four" (GoF) para resolver problemas recurrentes de diseño, mejorar la mantenibilidad y facilitar la extensión del sistema.

Adapter
Este patrón es fundamental en la Arquitectura Hexagonal implementada en la tercera iteración. Se utiliza para adaptar el núcleo de dominio (puertos) a tecnologías específicas de infraestructura (adaptadores). Por ejemplo, un *puerto* de repositorio define cómo persisten las entidades, y un *adaptador* de Spring Data JPA implementa concretamente esa interfaz para interactuar con una base de datos MySQL. De igual forma, los controladores REST son adaptadores que traducen peticiones HTTP en llamadas a los casos de uso del dominio. Este patrón permite que el negocio sea completamente independiente de los frameworks externos.

Factory Method
En el sistema se aplica el patrón Factory Method para la creación de objetos del dominio, particularmente en la construcción de usuarios y componentes relacionados al proyecto y sus documentos. En lugar de instanciar clases directamente dentro de los controladores o servicios, el sistema delega la responsabilidad de creación a métodos especializados que retornan la entidad correspondiente según el rol o tipo requerido. Esto permite extender fácilmente nuevos tipos de usuarios o entidades sin modificar la lógica existente, garantizando mayor flexibilidad y cumplimiento del principio de abierto/cerrado.

Strategy
El patrón Strategy se utiliza para encapsular la lógica de evaluación del proyecto de grado, permitiendo sustituir dinámicamente el comportamiento asociado al proceso según las reglas académicas vigentes. Cada estrategia representa una forma diferente de validar y decidir el estado del proyecto o documento cargado por el estudiante. De esta forma, la lógica de negocio se mantiene desacoplada,

facilitando la incorporación de nuevas políticas de evaluación sin afectar los módulos principales del sistema.

Observer
El sistema implementa el patrón Observer mediante el uso de mensajería asíncrona con RabbitMQ. Cuando se registra un proyecto, se sube un documento o se aprueba una evaluación, se publica un evento que es escuchado por el microservicio de notificaciones, el cual envía los mensajes correspondientes. Esto permite que los servicios permanezcan independientes entre sí, reaccionando a los cambios sin crear dependencias directas, lo que aumenta la extensibilidad del sistema y favorece una arquitectura desacoplada orientada a eventos.

State
El patrón State está presente en la gestión del ciclo de vida del proyecto de grado. Cada proyecto transita por estados como registrado, en evaluación, aprobado o finalizado, y su comportamiento varía según el estado en el que se encuentra. La lógica transicional permite controlar las acciones disponibles en cada fase, evitando condiciones complejas y manteniendo la claridad de flujo. Este enfoque facilita el mantenimiento y la futura extensión del proceso académico sin necesidad de reescribir reglas.

Facade
El sistema aplica el patrón Facade al centralizar operaciones complejas dentro de un servicio que actúa como punto de acceso para los diferentes módulos funcionales. En lugar de que los controladores o microservicios interactúen directamente con múltiples componentes internos, el façade expone métodos simples que encapsulan detalles de orquestación y manejan el flujo coordinado entre los servicios y repositorios. Esto reduce el acoplamiento, mejora la claridad arquitectónica y facilita la evolución del sistema.

Singleton (implícito en servicios compartidos)
El patrón Singleton se observa de manera práctica en la configuración compartida y en los componentes que deben existir como una única instancia en el sistema, como los administradores de conexión, los publishers de RabbitMQ y los servicios de logging. Aunque los frameworks modernos controlan la instancia única, la intención del patrón se mantiene: garantizar que ciertos objetos críticos existan solo una vez y sean accesibles globalmente sin duplicarse.

2. **Arquitectura Hexagonal.**

En el microservicio de proyectos se aplicó arquitectura hexagonal para separar claramente el núcleo de negocio de los mecanismos de entrada/salida. En el centro se encuentra el dominio de proyectos, representado por la entidad ProyectoGrado y el manejo de estados mediante el patrón State (clases de

models.estados). Sobre este dominio se construyen los casos de uso en la fachada IProyectoServiceFacade / ProyectoServiceFacade, que encapsula operaciones como crear proyectos, subir Formato A, subir anteproyecto y asignar evaluadores, sin depender directamente de HTTP, base de datos o mensajería. Los controladores REST (ProyectoController) actúan como adaptadores de entrada: reciben las peticiones HTTP y delegan en la fachada, funcionando como puertos de entrada (input ports). A su vez, el dominio se comunica con el exterior mediante puertos de salida: por ejemplo, ProyectoRepository para persistencia (implementado por Spring Data JPA), los clientes UsuariosClient y DocumentosClient para integrarse con otros microservicios, y INotificacionesClient/NotificacionesPublisher para publicar eventos en RabbitMQ hacia el microservicio de notificaciones. De esta manera, el núcleo de negocio se mantiene independiente de los detalles tecnológicos, lo que facilita el mantenimiento, las pruebas y la evolución del sistema.

3. **Autenticación y Autorización JWT.**

El sistema implementa autenticación y autorización basada en JWT (JSON Web Tokens) para asegurar la comunicación entre el DesktopClient y los microservicios. El proceso inicia en el login: el cliente envía el correo y la contraseña al microservicio de usuarios, donde se valida la credencial contra la base de datos usando contraseñas encriptadas con BCrypt. Si las credenciales son correctas, el servidor genera un JWT firmado que incluye como claims el email del usuario, su rol (ESTUDIANTE, DOCENTE, COORDINADOR o JEFE_DEPARTAMENTO) y la fecha de expiración. Este token se devuelve al DesktopClient y se almacena en la sesión de la aplicación; a partir de ahí, todas las llamadas a los microservicios se realizan incluyendo el encabezado Authorization: Bearer <token>. En el lado del servidor, un filtro de seguridad comprueba en cada petición la validez del token (firma y expiración) y reconstruye la identidad del usuario con sus roles. Finalmente, la autorización se aplica con anotaciones como @PreAuthorize("hasRole('DOCENTE')") en los controladores de cada microservicio, de manera que solo los usuarios con el rol adecuado pueden invocar ciertos endpoints, manteniendo una arquitectura stateless y coherente con el enfoque de microservicios.

4. **Gestión del proyecto (Git) + Presentación de Trabajo.**

Ramas:

master(dev1, dev2, dev3, dev4, dev5): Estable (entregables).

dev1: Juan Fernando Portilla Collazos

dev2: EDIER FABIAN DORADO MAGON

dev3: DAVID SANTIAGO ARIAS NARVAEZ

dev4: JHERSON ANDRÉS CASTRO MARTÍNEZ

dev5: ELKIN ADRIAN HOYOS GOMEZ

Guía de contribución:

- Uso de estándares de codificación consistentes, siguiendo convenciones de nombres, organización de paquetes y documentación mediante comentarios claros y Javadoc cuando aplique.
- Evitar lógica de negocio en controladores; los controladores deben limitarse a manejar solicitudes HTTP y delegar la ejecución al nivel de servicios.
- Aplicación del principio de inversión de dependencias, utilizando inyección a través de interfaces para asegurar desac acoplamiento entre componentes y evitar dependencias directas entre capas o servicios.
- Mantener toda regla de negocio en el dominio y servicios correspondientes, evitando colocarlas en controladores o componentes de presentación.
- Uso de commits pequeños, descriptivos y coherentes con las tareas realizadas, facilitando trazabilidad y revisión de cambios.

Evidencias repositorio:

RepositorioGit [https://github.com/Jfernando014/ProyectoFinalconHexagonal.git].

RepositorioGit [https://github.com/JhersonCastro/DesktopClient.git]

Anexos

[1] Ingeniería de Software HE[excel]. 2025. Anexo: "HE(Epicas)-SWII(excel)" . Disponible: HE(Epicas)-SWIII

[2] Ingeniería de Software HU[excel]. 2025. Anexo: "HU'S(Historias de Usuario)-SWII(excel)" . Disponible: HU'S(Historias de Usuario)-SWIII

[3] Ingeniería de Software PB[excel]. 2025. Anexo: "Product Backlog-SWII(excel)" . Disponible: Product Backlog-SWIII

[4] Ingeniería de Software Gráfico[excel]. 2025. Anexo: “Burndown chart”. Disponible: Burndown chart

[5] Ingeniería de Software PB[Jira]. 2025. Anexo: "Trabajo de Grado(Jira)" . Disponible: https://unicauca-team-dag79f44.atlassian.net/jira/software/projects/KAN/boards/1?atlOrigin=eyJpIjoiOTMwZTcyZTVjYzhlNGFmY2JjYTYyZTA1M2VmYmFjY2YiLCJwIjoiaiJ9

[6] Ingeniería de Software Bounded Context[Draw.io]. 2025. Anexo: "BoundedContext tercer corte(Draw.io)" Disponible: BoundedContext tercer corte.drawio

[7] Ingeniería de Software ModelosC4[PlantUML]. 2025. Anexo: "Diagrama de Contexto (PlantUML)" Disponible:

https://www.plantuml.com/plantuml/uml/ZPBDRXD13CVlUGgJKolHviA9KqCQ8g4YLHIuLiwCkvrgThpCHsX0F18dRbppOjYxscJQ2C5bPNI7F__zSP4onbnQFtdXWl754jpdtANNird4h-UDo_TbNH95mo5JoEU6srddCMZEMShMk_LiyUhiMaEVtb_DMaoPeeHk5vhncCyxfK-kAIOE9vQDKEWbJ2y7QtemuKiXw28bSH2sq53AZWF2MuujPaumb-XSNdQHjsGaCZqTmVAsM8S3UxbtfhKtL4YkaIzva1Iq35RC4NZW6UPeNP07KH2BWoVluJfo4x75dhv1I_kz5zbhjFI0m6iPsmQDj42Lq7TqHvVKOSpOIgCi20qfjWuh-hDB1RtxXe1rqr80fOc0OCKXgLzKM5zaShEL-RUtRxW4Yt5xafpwABIRmOALZdNtAsZ5Zn7N-C19xLL4AmE0xnE0WJ8chrQIhcaSZpDKwBpppg3fgT82GgH6KgCqbyhQMTtceDECbQjn7QtqnxwDvMC-4G7EtyWedT7T3fPZXIpr5uZCjdYHSgfQhhXfv01JzInmTdUZG1tI1_A7IwoQ5FWeNIRcWW0G4gLUz5PxgONp4-72LMnzXpLvmI4Ld_4lLoUfJq-kpfqRGz86Ul1ko9PnFZ-x0N9yNlyWMVhFum84WnQXX55l1NwsesNOx7vYDNSEb0vx0iC1VB-dKs5TKB2bzNy0

[8] Ingeniería de Software ModelosC4[PlantUML]. 2025. Anexo: "Diagrama de Contenedores (PlantUML)" Disponible:

https://www.plantuml.com/plantuml/png/hLXHRzis47xth-2y0Kk2fGg68KE6jQXYAvcBn74jfLsVWfDqLXX8eaTIxR9rlsA_eqzxwwl_s8wKP5cM0g50yn2Jqjtt7O_7xsY_L1gaBh9qz0tFexI8aTrhlLQ_E8w43yyIhk-Bi50e8v5hpFMpI6JEEeNS-1mh7QSyT2O_7I_CezIhAoS3fL7Iexi9UG3FKJvR6_oHvZf5DfvoI2Ham8XgOa4n5X8LEsO-9rzwSubAyyr_EGiaXF0WbB6zb12ByMZqFNk-zySMtjA_cRj-vyreWLA9_20M4HBJ3yHUZiOqz8iGsOMG6MXYSDaZS-dDMef7ZEZ9-B3sfcYAc4C9u6qd1eCMe8fK0oCR2i_4sGCG2I5ZdfE1D3xDp41uxo7TV8Ow4kOsVW-uCfIlwCCWJt4DKaD62n36qrKyoOcQ00g3XaJPcnz_vWVUjTjDZ_ze4dvtBeey1IbueCeD8EnwAs9iTYF6b2qa9L2PXTQRizqRTd0nyuB3yI7xUyJep_xhYSPTNCqcxcIs-NVEtdhd7GFhjosUWvMqHH9JK1VLaBt5q2OWJueKfBD4YBHvCAFNSWL_cPWALO3aWcq-vpo2yU50J3

DswGRUM_VTK3omvdS9QFm0ZyHe7SiPSzXI51I0BQwrv7d29gaeujh0xdOk2m
qPDtj85QSmAIIF8Hw8xtesMTxutIBDR3AxyOT2DEVrBbD4PqR76O-aeC5x7k5E
RET2sAGr-xjDcgBGe33ddL8NcPSKNvk3oIm18rtIL9ClOQhAlNFkEaInY0eg-Ogfc
WrlkMa64UPrfbCHGMfeGKRt_BrGR36zw53bGlCLZq1paPTqkq-6AEULRKlsg1Y
epQUS779Gx20LIOBoiCEOeLAG44f9jvqESTb2pp3cjiHZ914mwoIcIAHmE51Dv
wxl-MpgiQaR35RJD2phAGw9zvP6xBSVsVJSH646e244XRivtwqYAL8ZInt8kdmi
wgAS30FNHKB8bUBsO3SLOy6drMmOFKD215sscngabw5LA9PbllDac2blbKgN
eAcB4lswdWv3ruL0VPmhBII7TA0QPlEBfUlvmV9s4jmktK4zvYagspTCShd4oB1K
HspSRXtrQMj5I2UFmikPqwJOOOAceC5vueJWtHy55cYxPmfDmLla9OGXrzUlcS
DydXKfn32sISiLFA3SV8ATmqW4__GsoQLtvPAupZsVkPFWPacVno-URfrBJ9jhn
KwFkbN0C6LL9xKx5GGBdsYDHtELQ9owrmYrvzQwG3H-dRk3Okw-fxb2D2xzbu
R6PIWnitdWBNiIMeQhOMdto0LgNecYuI50FzWZtMsZbFNhHlsuK1Qy1TaqEAVM
6kUeXdGi_c7NgzModAsM37koTW_QBTnXtsvJSTh7luj12MbdmEJa71IMoWIwoi
as3Or0No4wdeea3z6WlfgUJwgWTbTVZOTSMikk9eDEdVMs7Wmuj_jXCtlQpIHh
8h8YhsygZ6AKoB9M0ZiPQE-eNuHSCeTDKKKW4v16fBexsuROwzsB8amvUJV
Ar7LlkxlahK4mscYwk1NDwzS4KEbRVuAxE5PDiaOYkpWcQUqRqgvQT_U-fVcN
1SWOjjQby6VrjweBCcM-jJMdkkJQknerDuJ-1MrrbPOY9Of6laLpbJBobm5FcIerN
poLdafplnvWIus_2g9Hwl-7wXFh-Kr0ykpzRhzZ1RCtVLhjnZ554K1oC3RVzGdxEu
r_QkgiGZuVVtjoSdeQHOI_OnTNLuskxTdPoKdR5jHZ7kqRhbQdf_k6L5KTmBCp
OxUj6b0CrHgZpIUJBsMyjrS3Yn49QJ72f4R0qzNFgnKXT6pD8jgsgvFLQdNIPylp
bTmt9bCyVIBILzuxRpxjkIUac60UCycJUprYx2EpSFGPyBLW7-sZNwkWNz2u9Vn
aBB4sgg8rHgONs5y4J2jetji0pVkPYHwLbaMa2mcLnT6HqQEZ8ngs5REkjyQGzA
BF4BgMHeMjfHIP7mQ0XLLmSHKSBRzCmcZqaluNMVe_

[9] Ingeniería de Software ModelosC4[PlantUML]. 2025. Anexo: "Diagrama de Componentes(PlantUML)" Disponible:

http://www.plantuml.com/plantuml/png/bLbFRnkx4R_lfo1Se2y16hd9-rCqI0rl9FbP
3xQiIdBQ0BbGkoEPoOhSabmtVdq1-bbww-6TSklLtwIVfBCSxgv8QMqt0UmbvpS
_SeQS6P8-DPPhMssBn7mIikIQRrccjgMI8Et2tXN0D6IMoqq1Ev0Srhmgx9cITigt
m8vIBNXnr24wBVQFXB6tFFkqqQgI-L0LIhFVhFzO_wybIkUWVUyVLIL_x9qJL4I
GxENtoPSaiS8MECv8y0tISvm16pRp1CD6O7bHy5on5socrKSSdLsAJ2i3-bPay4
4-ozG0ycfGIP7nAkE3KgixX2dp_2X9IfmWt-009Ic_sEue7w4DEAbls9yE_6FfRC8k
qlVZ-K4nwdMkE79OVeTMFHkYxxGg2j0u0SQDuwbznRe5O4SpFyaE_K6Upi7O
hiCfTysOOGIhQIEOUWH2ZYyFMBmOpzzDXkCdMxqWDpQMlwsCa60Ckr0RaP
7n7cAy1uIqeDSyG-L9um8FEUCPpt4zoO4yiqGmeVxO-XxbKzzsgjH98uVQPqflkL
NfebfjXRMrz_1JQEO5B2M26BRFbA91pORROziLEi8OtKCwlkL5nJEXv3cNkLjS
weed604FJj28ZGovg4jxxLnfC8ujWWPiaTIVyKBaE0zaeq_DAeFp4ugeOXnHVN4
_7zi8-rk018_j0HS1fJ82XxiBPh6h7vEqJaXnOLREUYHgKv3Z2J6z37F8RlZFgb_T
0nw9mj7LPJgPFZa8HshBXMnYy5BbK1ZsMxQmt8BFG0JfJq8_QfxNqnvBAspW
EX8T2Y77psQyti4Ib-n7C5RSVvKCVwEA7zcXF2ZU0crg7fF6qQ5XndAcnHOqxx
Qwu_mWISkDGfyl--ESrZPNVQfElgz-PRLg8EQ370m6JiTr-zvUeqRBAyEESUsH
AJ8jyluS_bQXqz27gCDGX2xdvBu8qc_WHAuLQdiE3QQiyPub0kspN5FeKgs5be
UqOs6Z_s1M5RbobYEH-wt2cRY-XvDfFyaEYkGF9T1Wn8YeGtcQ_jovaoeFCH4
mPFhoI9XFfcVpTB6SNm-Nr_EdrrnSHCsDrLLcAzsMtl7dEi9umOQ5m0duw8km-
oLuggnOYypD70pgaccp2dHZ6c58T321XXopQbK8S-FE8sdEI-iF8WxL2W-5mvol
Li9U_fajGEQkvCdR-ryvsu8q_AClKGGYH1nw-nkwjUZq3721gELuGa2iFb9PjSMl
1mWYIEIQo2GiPsknGHRwg3ItwAmTWwY_BprFrLXchY9YG7pEeBHKfDhI85UqI
O_ilzbulfWibkFf_t7Wcu4smbYGpQ6FHkvw-sjCLvhtwtJNTDevjtp53Ovt_egDV0Et
vQ4ZF6tIyrUZju-UROVNyydo_PEEjLcbcqBOD6BBEjW3gnj2OeAV_cxJojxWjyYS

pfaeh8iS-d0K2ABM0Usb-WGI7NmhNC14nmu4D3Bd6_HEZs_6VrsEvzFqWYtUu
uQuN1psaLGvEgT2nmShuEC9zEN2Gt8V6p6iYng4Oa6he0iHNDuj65ENpfL6ipGj
R2Fsp99oZNFMFljqHueZz5ax7r-amydLTBnWi_baEfpCqejnZytVkDlRYvFs7fDqb
ods0llzzOIzPfNXITDwyyjWm5g9s9O5r6RpXFf8aoXRLMgIRdHoZKXmLfv7l4hxPf
Tu_zRI2OFYUGWGNoa28TMvFk7-cJ_Pcue3nhaJhJFL2jXiT9R4ytkQLZmqQOH
97l6tJjpULD2HpT5OG-6sZRaHPU3Cu1oC59a6VimpMo7Lppm9nQZGd4KH2kw
B2IaomOi-A9qz4MurbqPG5Tq71uTEHAkwIKEG8RHkQBf9eYKa_qMNcDfxWd5_
BkvHYIykfAHTWxG2oaF-0eJS_uSdCTL3O387f0rMOlSWWjWc5Q2IsnH987T159
qvWXYAP3X0SBP0CBZZHnBq4enI4aBAMZzpyzsfWKaKlAJLbZWVxrs7azSL32
K_pTBZSx6gWzf2pTiUx33pRolfivj9ehsD53u78ydvSZd3BBTO9d7C184KOLkx41
EbKkSDgs7JUHkdrXItA2Ls5Qlsa5OMDW9NjrGwQIM4w-fJ5vksBcacQMM4ZGef
mhEcrdfgd8-xSLLr2Obc55-ZK5r39bOnBhueEPnLMWPvExXH-8pGbs7xC5sunO
Wuq_OZzkvbjB5wozxqQfakiDJFBYR3T3bvb_QMUGjCYysDPMhDsjUtDwkJWsz
kRuvN9oY-qiwrTOJZeWkPYP8N3FjUe_3bWBqBKozx5cxWvuXwDM2BEg4tnuS
Q5MvWmdqxOEwnGKdk4o02W-U86lNTWBdmfWn1R3rNbHh-_O1TSWaV4Qsrq
c3O7MLadX4w0PcpsZT9vA4WYNb7SQlhfn0tAU-W9dLtEGXRjNF--yz_iRvNZbQ
yW4qbyrxn4dJMgttm4IIIKLQEXDI9EsmjfB3YjXLUtl_Rgdhw3Do8-jdptp4DGjhQBc
prEljAoUIjdbKXlxMB-1speaJ_vT2ifzzP1ruTQ3FrF0BG3kfvQg2Dqt_lH_cruLqDo9
HaMbLMo0tx17UzkoIglcJu3ZaPIwkv_o3vck5t1IlUFV_6OBzMhnbzXCX9k-SaA-w
_RbnGc1Ao-rzhnvDFoLcuRL7XBZYDeDm7McTUNvQv6ByVJqVfuTHIm0O9o2d
aWXJJXh2GsKfpTWwV-KP9NdXl_CBepqVq3rkyvFHD7GxY_-Ba8NYS4rAe208_
hBzThpqaBN4_aQKHPWq_K0yzd43pSiARaTRhtxSGI1SS_mLMuKXlgfD9yxo2Q
rgTO1-zPLjbZcTAMuD8z05v8Zd57zMs-1y0

[10] Ingeniería de Software ModelosC4+UML[PlantUML]. 2025. Anexo: "Diagrama de Clases(PlantUML)" Disponible: https://www.plantuml.com/plantuml/png/XLLTRXit47xdAGRR1pzvNJjjqmI9OKMM4WEMB4jAqOU-ZBYKmeHBRlbZr2ZysWFq1Zb0dte4twGdwP3SzIgoubZ0OcVc6txV_71zvZnQ7sh5t2Uf6xHOGoNM69GV6-sdM0iOM8dRSIkuHxrH0lva06-GVzfO4tGrDCfO-6xz8lvYnDXAsDRxV1L_v9rdgDN2ml7tx8unBpsTLGnDtHWjj1UE92WOg_3HmC4YhAI5iR4rUWE3WuAnXYXnGocDDHz9p64jkJLEs1l9HP5KCTYIMGnc5t29jyBc803lkA2OMNDBI3EaSgrHIjZVzDbSEDyx2a07lKadt3sYM6JsRzAqk35oh0GnjNQYoCwKkvFTbZ-WPjJI-wYHNgbyBfn9kG-Hnyc_e98LPLDIVkMI8YwbTR6STfUUtv2UsjCO9odhDdFdmDw-d9iQfNwY8MyjLeRE76alVJwl2-thnCZvX1_fcPMri3YwGHLIIGkFFZSoWt99UrLTwBL5vstWFbYnHzrx5z1AuuPAaatona9kTBOInNidR3PtKyyD3pMzFP5CaFsvK-FbMl9sGhEmKj9zI1kPRqTtGC9zhQFzT1wFAffzHohqk48de7Xt0kUjaHM-YJVp05xJ7un-NOxcqy5bifWs5IK4gd_dthLTcInQIDKsuZ6CwYVKxq44WirnjP9-SfqYa-i7rIdeSZ2yk9gE5fctlu1mU7ZQNPQNK1oNSAcmne9bNmptYnqH9sMs2TBx8wgVOeGzws0Gsl4Nh4Tq2LHW1BzRBcUaSR4is1PlTnSYueSIXbOWDErS2fQ36HWd4L4_bZ0JraddHS6YCqQf_p7sKmbIlKY5EnpTNO-OvsMq1NJTZg9x0C7xROlmdrlux7nSi3xIufugBIzTZBye8QqcHn0taR9WEKOecc2432RNi-9XbDEhvM3HROyNu4q3Pltb9_NLwdIagyP8xTtBLqUhK_BDhXPBEEhwvuxM-JDuY366zlvpBUW0sfgWl6pGUYPq1P62PI8hDnyyKVNha9csluCT6QrHT-j1_17deSbBgpCtIZZWnjgaWfoqA1DJfJCgkBd_75_87GTE8X2S1CfeI23tagSDM0j7RAXs5IgnZWBxJnQfo-yWjUGILQVlQryy1eJqSGm3xWCgU0t7MonARAAnCjwREf-V_xFg40-hJ3eVZk6_I_w6a_HybfvbMMPuN8SC6fewa2ZYllzNWxl_HtDBHMKOZNml37iS2Sbos1azGgt-1m00

[11] Ingeniería de Software Repositorio[GitHub]. 2025. Anexo: "ProyectoFinalconHexagonal(GitHub)" Disponible: https://github.com/Jfernando014/ProyectoFinalconHexagonal.git

[12] Ingeniería de Software Repositorio[GitHub]. 2025. Anexo: "DesktopClient(GitHub)" Disponible: https://github.com/JhersonCastro/DesktopClient.git