**Arquitectura y diseño de software**



Ingeniería de software II

Entregables primer corte

Presentado por:

Julian Stevan Camacho Rosero

[Santiago Alejandro Hurtado Campo](mailto:santihurtado@unicauca.edu.co)

Oscar David Cabezas Moreno

Profesor:

Brayan Daniel Perdomo Urbano

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería de sistemas

Popayán, septiembre

**Contenido**

[**1. Contexto: 1**](#_heading=h.vjmb8tftb285)

[1.1. Propósito del sistema: 1](#_heading=h.r7fqc4x7pyc8)

[1.3. Alcance funcional: 1](#_heading=h.shqt6zdbvpwo)

[1.2. Situación Actual (Corte 1 - Implementado): 2](#_heading=h.4wsmma1ilxre)

[1.3. Actores y Responsabilidades: 2](#_heading=h.ogkv7zje2dhh)

[**2. Decisión Arquitectónica: 2**](#_heading=h.44s790obocco)

[2.1.Frontend Desktop: 2](#_heading=h.f18zvaqxqph4)

[2.2.Backend Monolítico Modular: 2](#_heading=h.nxg4on28fkan)

[2.3.Comunicación Frontend-Backend: 3](#_heading=h.tetes2q7pvx5)

[2.4.Patrones Implementados: 3](#_heading=h.kto6cwhzwhup)

[2.5. Evolución planificada(Corte 2): 3](#_heading=h.x0zys0131hzs)

[**3. Consecuencias: 4**](#_heading=h.mbhsb0ozlwxq)

[3.1.Positivas: 4](#_heading=h.ce4cwtv5u354)

[3.1.1. Rendimiento y experiencia: 4](#_heading=h.a0oluxdfw28b)

[3.1.2. Mantenibilidad: 4](#_heading=h.m9osnjpz52vy)

[3.1.3. Escalabilidad: 4](#_heading=h.bl3j1ls3378j)

[3.2.Negativas: 4](#_heading=h.avmowuxlydf5)

[3.3.Mitigaciones: 4](#_heading=h.r0r26va77ay3)

[**4. Tecnologías y versiones: 4**](#_heading=h.kam97j5w56ub)

[4.1. Frontend: 4](#_heading=h.uuhcd0sx62y3)

[4.2. Backend: 4](#_heading=h.rudj5uxawdo5)

[4.3. Infraestructura: 5](#_heading=h.ypurxw3uzdi6)

[**5. Diagramas: 5**](#_heading=h.wobgh4vfkedg)

[5.1. Diagrama de Contexto: 5](#_heading=h.lh5kagzbddb)

[5.2. Diagrama de Contenedores: 6](#_heading=h.f7q2vawcati7)

[5.3. Diagrama de Componentes: 6](#_heading=h.cz2sjmjitiif)

[5.4. Diagrama de Clases: 6](#_heading=h.fjfqc610d0dj)

# Contexto:

## 1.1. Propósito del sistema:

Sistema integral para la gestión del proceso completo de trabajos de grado en la Facultad de Ingeniería Electrónica y Telecomunicaciones (FIET) de la Universidad del Cauca.

## 1.3. Alcance funcional:

* Autenticación y autorización multi-rol.
* Gestión de perfiles (Estudiante, Docente, Administrador).
* Dashboard personalizados por rol .
* Gestión de proyectos de grado.
* Seguimiento académico y documentación.
* Comunicación entre actores del proceso.

## 1.2. Situación Actual (Corte 1 - Implementado):

* Backend Monolítico modular con autenticación
* Frontend Desktop completo en Java Swing
* Múltiples actores con interfaces específicas:
  + Estudiante: Dashboard con proyectos, postulaciones, documentos
  + Docente: Panel de supervisión y evaluación
  + Administrador: Gestión de usuarios y sistema
* Integración completa frontend-backend

## 1.3. Actores y Responsabilidades:

* Estudiante: Gestiona su proyecto de grado, postulaciones, documentos
* Docente: Supervisa proyectos, evalúa estudiantes
* Secretaría: Gestión administrativa del proceso
* Coordinador: Oversight del proceso completo
* Administrador: Gestión de usuarios y configuración del sistema

**Sistemas Externos:**

* Identity Provider (autenticación local)
* Base de Datos SQLite para usuarios, proyectos, sesiones
* Sistema de Archivos para documentación de proyectos

# Decisión Arquitectónica:

## 2.1.Frontend Desktop:

* Tecnología: Java Swing con GroupLayout
* Patrón: MVC con separación clara de responsabilidades
* Navegación: Sidebar con paneles de contenido dinámicos
* Branding: Logo institucional Universidad del Cauca

## 2.2.Backend Monolítico Modular:

* Módulo de Autenticación: Login, gestión de sesiones, tokens JWT
* Módulo de Usuarios: CRUD y gestión de perfiles multi-rol
* Módulo de Proyectos: Gestión de proyectos de grado
* Módulo Académico: Seguimiento y documentación
* Módulos del sistema de gestión:

co.unicauca.

├── presentation/ # Controllers, Views y observer

├── domain/ # Lógica de negocio

│ ├── entities/ # User, Project, Document

│ ├── enums/ # Career, Role

│ ├── exceptions/ # Domain exceptions

│ ├── repositories/ # Interfaces de persistencia

│ └── services/ # Business logic

├── infrastructure/

│ ├──dependency\_injection/ # Framework DI personalizado

│ ├── persistence/ # Implementaciones Repository

│ ├── security/ # Auth, Encryption

│ └──validation/ #UserValidation



## 2.3.Comunicación Frontend-Backend:

* Se usa el patrón cliente-servidor
* Frontend(Swing)←→Controllers←→Services←→Repositories←→DB
* Endpoints principales:
  + /auth/login, /auth/logout
  + /users/profile, /users/update
  + /projects, /projects/{id}
  + /academic/progress

## 2.4.Patrones Implementados:

* Observer: Para comunicación entre controladores
* Dependency Injection: Inyección de servicios
* Service Layer: Separación de lógica de negocio
* Repository: Acceso a datos

## 2.5. Evolución planificada(Corte 2):

* API Gateway con rate limiting y routing
* Microservicios desacoplados
* Frontend Web responsive
* Sistema de Notificaciones
* Auditoría y Reporting avanzado

# Consecuencias:

## 3.1.Positivas:

### 3.1.1. Rendimiento y experiencia:

* Tiempo de respuesta inmediato (aplicación desktop)
* Offline capabilities potenciales
* Rich UI components nativos del SO

### 3.1.2. Mantenibilidad:

* Código bien estructurado con separación clara
* Patrones establecidos facilitan extensibilidad
* Testing facilitado por inyección de dependencias

### 3.1.3. Escalabilidad:

* Módulos desacoplados listos para microservicios
* API consistente permite múltiples clientes
* Base sólida para evolución arquitectónica

## 3.2.Negativas:

* Dependencia de plataforma: Desktop application vs web multiplataforma
* Complejidad de deployment: Distribución de aplicación desktop
* Limitaciones de UI: Java Swing vs tecnologías web modernas

## 3.3.Mitigaciones:

* Arquitectura preparada para web: Backend REST listo para frontend web
* Módulos desacoplados: Facilita migración futura
* API consistente: Permite múltiples clientes

# Tecnologías y versiones:

## 4.1. Frontend:

* Java Swing JDK 8+
* Swing/GroupLayout
* Maven/Gradle

## 4.2. Backend:

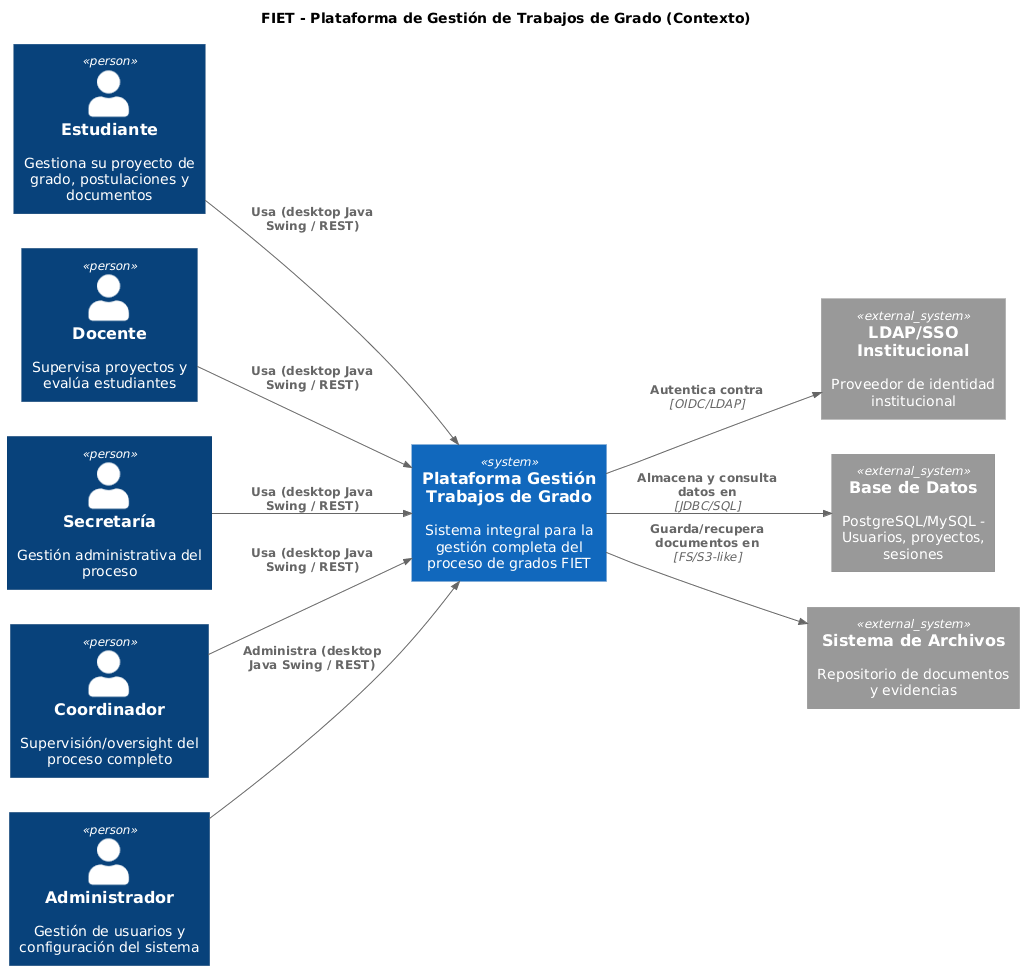
* Java SE 8+
* SLF4j 1.7+
* SQLite
* JWT para autenticación

## 4.3. Infraestructura:

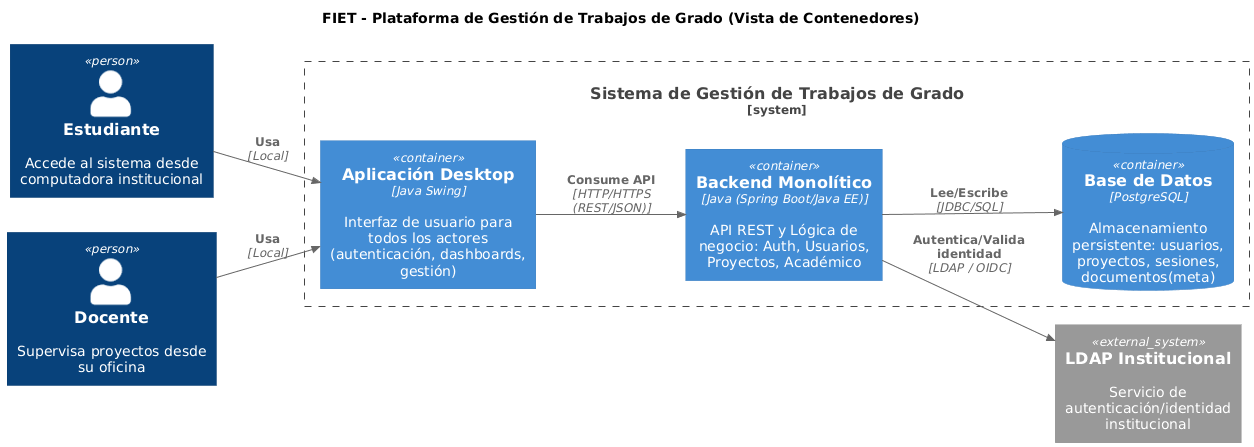
* Docker (para despliegue)
* Git (control de versiones)
* Maven (gestión de dependencias)

# Diagramas:

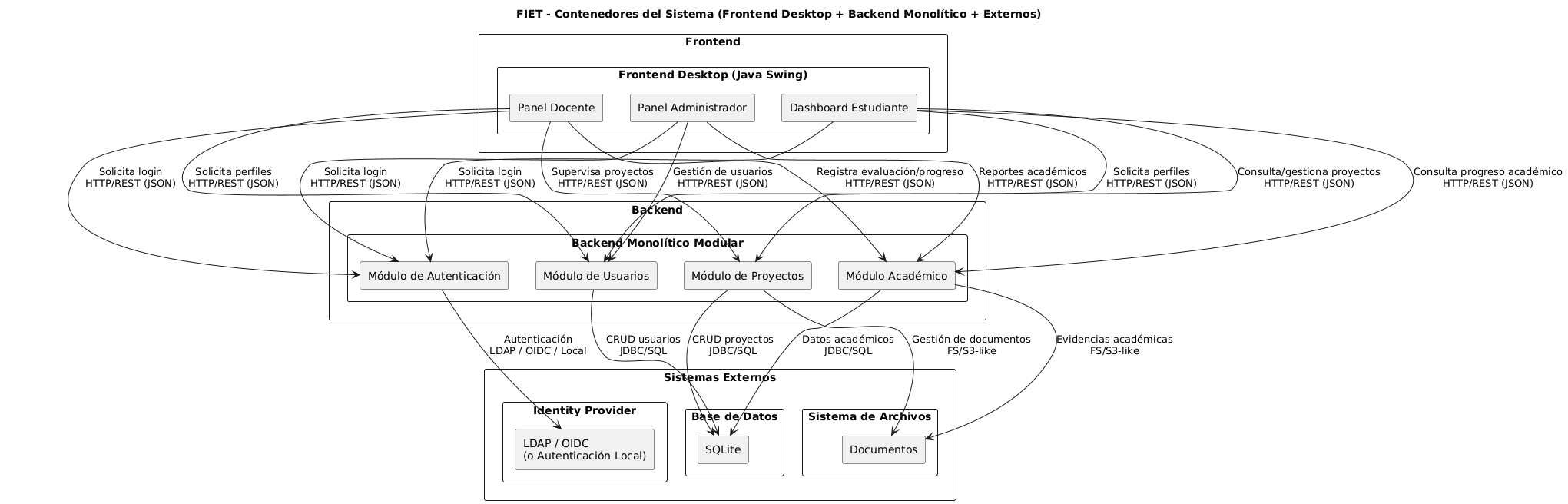
## 5.1. Diagrama de Contexto:



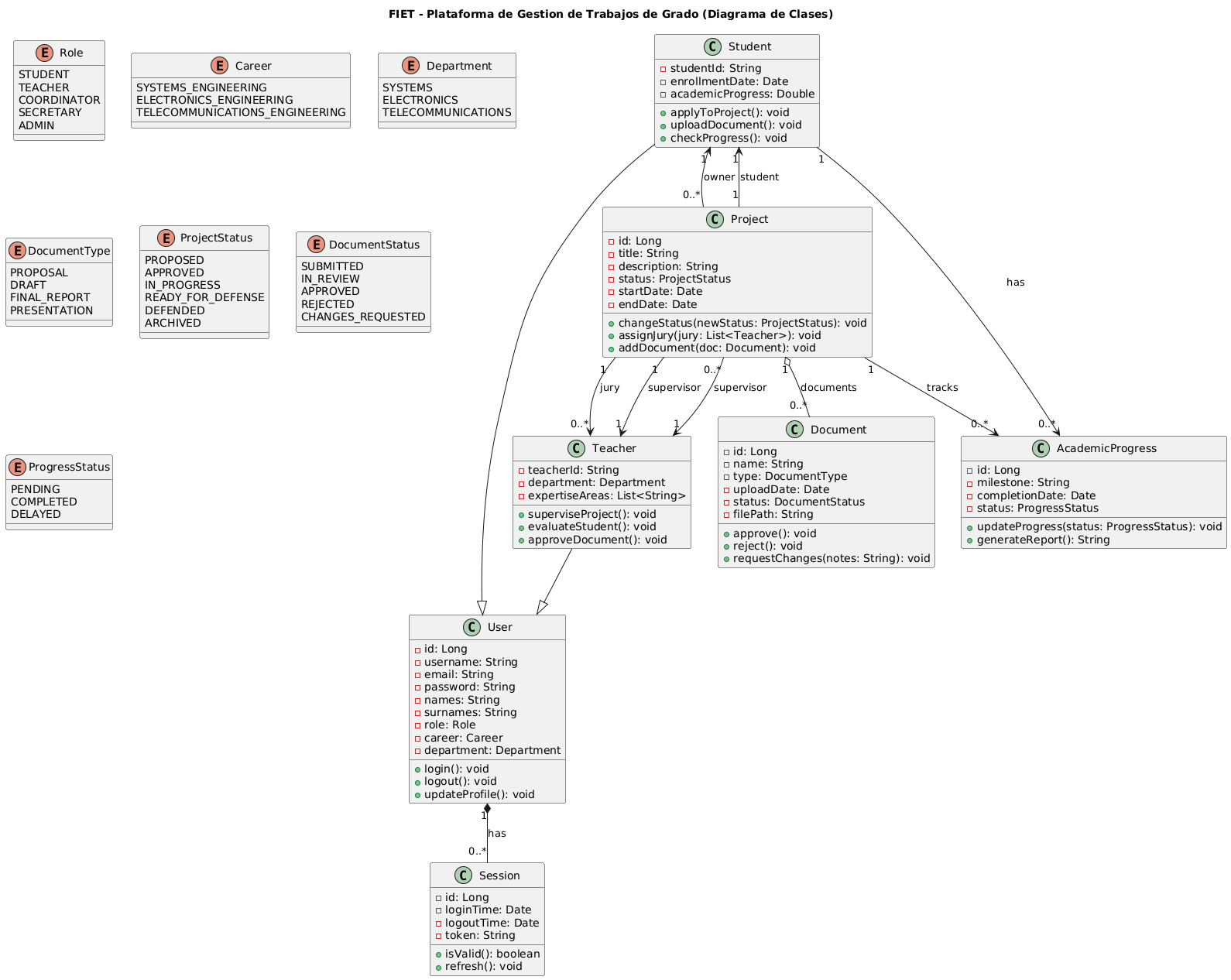
## 5.2. Diagrama de Contenedores:



## 5.3. Diagrama de Componentes:

****

## 5.4. Diagrama de Clases:



# 6. Métricas de calidad Arquitectónica:

* Acoplamiento: "Bajo (módulos independientes)"
* Cohesión: "Alta (responsabilidades claras)"
* Testeabilidad: "Media-Alta (DI facilita testing)"
* Mantenibilidad: "Alta (código bien estructurado)"
* Escalabilidad: "Media (preparado para microservicios)"