ESCUELA POLITÉCNICA NACIONAL

MANUAL TÉCNICO DE LA APLICACION

2026

Casillero Inteligente

AUTORES:

ALBÁN SALAZAR EMILIO FABIAN

ALCIVAR GOMEZ ALAN NAHIN

ALMEIDA REYES ANTHONNY JOEL

ALOMOTO GRANIZO ARIANA THAIS

ASTUDILLO CAMPOS JOSUE SEBASTIAN

Quito Ecuador

+593 99 623 7293

Índice

1. Introducción Técnica

1.1 Propósito del Manual Técnico

El propósito principal de este manual es documentar la estructura interna, la lógica de programación y los requisitos de despliegue del sistema de gestión de casilleros IoT.

Sus objetivos específicos son:

- Facilitar el Mantenimiento: Proveer a los desarrolladores y personal de TI la información necesaria para realizar correcciones, actualizaciones o refactorizaciones en el código fuente (Java 25) o en el firmware del microcontrolador (ESP32).
- Estandarizar el Despliegue: Definir los pasos técnicos exactos para la puesta en marcha del servidor, la base de datos y la configuración del hardware.
- Documentar la Integración: Explicar cómo interactúan los componentes distribuidos (Hardware físico, Middleware y Backend).

Este documento está dirigido exclusivamente al equipo de desarrollo o personal de soporte técnico avanzado.

1.2 Alcance del Sistema

El sistema "Casillero Inteligente" abarca la integración de hardware y software para el control de acceso físico automatizado.

El alcance incluye:

- Control de Hardware: Gestión de actuadores (cerradura) y sensores (teclado) mediante un microcontrolador ESP32.
- Lógica de Acceso: Validación de credenciales (PIN) centralizada en un servidor remoto.
- Gestión de Identidad: Administración de roles diferenciados (Estudiante y Administrador) con permisos específicos.
- Auditoría y Seguridad: Registro persistente de eventos (ingresos exitosos, intentos fallidos, bloqueos) en una base de datos relacional.
- Recuperación: Mecanismos administrativos para el desbloqueo y gestión de credenciales perdidas.

2. Descripción General del Sistema

2.1 Problema que Resuelve el Sistema

Los sistemas tradicionales de casilleros basados en llaves físicas o candados mecánicos presentan múltiples ineficiencias y riesgos de seguridad:

- Pérdida de Llaves
- Falta de Trazabilidad
- Gestión Ineficiente
- Vulnerabilidad

2.2 Solución Propuesta

Se propone un sistema de Smart Locker conectado, que desacopla la llave física de la cerradura mediante el uso de tecnología IoT y una arquitectura Cliente-Servidor.

La solución se estructura de la siguiente manera:

1. Interfaz Física (Frontend Hardware): Un módulo basado en ESP32 que actúa como portero. No toma decisiones de seguridad por sí mismo, sino que captura las credenciales (PIN) y las transmite, esperando instrucciones de apertura o bloqueo.

2. Núcleo Lógico (Backend): Una aplicación desarrollada en Java 25 que actúa como la autoridad central. Recibe las peticiones, consulta la base de datos para validar si el PIN corresponde al usuario y al casillero, y verifica el estado de la cuenta (activa/bloqueada).

3. Auditoría Activa: Cada interacción queda registrada. El sistema permite al perfil Administrador visualizar un historial detallado de accesos y gestionar incidentes (como desbloquear un usuario tras múltiples intentos fallidos) desde una interfaz de software, sin necesidad de herramientas físicas.

3. Arquitectura del Sistema

3.1 Arquitectura General

El sistema de Casillero Inteligente utiliza una arquitectura por capas, separando claramente las responsabilidades del dispositivo, la aplicación y el almacenamiento de datos. Esta separación mejora la seguridad, el mantenimiento y la escalabilidad del sistema.

El ESP32 actúa como dispositivo de entrada, enviando el PIN o Token al backend. El backend valida la información, aplica las reglas de negocio y registra los eventos en la base de datos. El dispositivo no accede directamente a la base de datos, garantizando mayor seguridad.

3.2 Estructura del Sistema

3.2.1 Storage

Database: casilleroInteligente.sqlite, almacena usuarios, casilleros, eventos y tokens.

DataFiles: archivos auxiliares de persistencia y apoyo.

Logs: AppErrors.log, registra errores y eventos relevantes.

Scripts: DDL_DML.sql, creación e inicialización de la base de datos.

3.2.2 Capa de Acceso a Datos (DataAccess)

Contiene DAOs, DTOs, helpers e interfaces encargadas del acceso controlado a la base de datos.

3.2.3 Capa de Lógica de Negocio (BusinessLogic)

En esta capa se implementan las reglas del sistema. Incluye:

- **Entities:** representan los objetos principales del dominio.
- **Interfaces:** definen contratos de comportamiento.
- **Mappers:** transforman datos entre entidades y DTOs.
- **Services:** contienen la lógica de validación, generación de tokens, bloqueo de casilleros y control de accesos.

Esta capa asegura que todas las decisiones críticas del sistema se manejen de forma centralizada.

3.2.4 Capa de Interfaz de Usuario (UI)

La capa de Interfaz de Usuario (UI) es responsable de la interacción directa con el usuario final. Su función principal es mostrar información, capturar acciones del usuario y enviar solicitudes al sistema, sin contener lógica de negocio ni acceso directo a la base de datos.

Esta capa se divide en Controllers y Forms, siguiendo un enfoque similar al patrón MVC.

Los controllers se encargan de recibir las acciones realizadas por el usuario desde la interfaz gráfica, como el ingreso de credenciales, validación de PIN o solicitudes de auditoría. Su función principal es coordinar la comunicación entre la UI y la capa de servicios, validando datos básicos y delegando toda la lógica de negocio a los services. Los controllers no acceden directamente a la base de datos ni contienen reglas del sistema, garantizando una correcta separación de responsabilidades.

Los controllers se encargan de recibir las acciones realizadas por el usuario desde la interfaz gráfica, como el ingreso de credenciales, validación de PIN o solicitudes de auditoría. Su función principal es coordinar la comunicación entre la UI y la capa de servicios, validando datos básicos y delegando toda la lógica de negocio a los services. Los controllers no acceden directamente a la base de datos ni contienen reglas del sistema, garantizando una correcta separación de responsabilidades.

3.3 Integración con el Dispositivo ESP32

El ESP32 funciona como un dispositivo cliente que captura el PIN o Token ingresado desde el teclado matricial. Estos datos se envían a la aplicación backend para su validación.

El dispositivo no almacena información sensible ni ejecuta lógica crítica, lo que reduce riesgos de seguridad y facilita su reemplazo o actualización.

3.4 Estándares de Desarrollo y Base de Datos

El sistema Casillero Inteligente adopta convenciones de desarrollo para garantizar legibilidad, mantenibilidad y coherencia del código.
Las clases e interfaces se nombran usando PascalCase, mientras que métodos y variables utilizan camelCase.
Las constantes se definen en mayúsculas para diferenciarlas claramente del resto de elementos.

Las capas del sistema siguen una nomenclatura clara: las clases de acceso a datos finalizan en DAO, los objetos de transferencia de datos en DTO, los servicios de negocio en Service y los controladores en Controller, facilitando la identificación de responsabilidades.

Se aplican principios de buena práctica como separación de responsabilidades, uso de interfaces para desacoplamiento, y centralización de la lógica de negocio en la capa de servicios, evitando su uso en la UI o en el acceso a datos.

La base de datos mantiene nombres coherentes con las entidades del sistema, uso explícito de claves primarias y foráneas, y scripts SQL separados del código fuente, permitiendo auditoría, seguridad y fácil mantenimiento del sistema.

4 Hardware del Sistema

4.1 Descripción del Dispositivo ESP32

El núcleo de control del hardware reside en el módulo ESP32 (específicamente la variante ESP32-WROOM-32). Este microcontrolador de 32 bits ha sido seleccionado por su capacidad de procesamiento dual-core y su conectividad nativa, actuando como el puente entre el mundo físico y el servidor lógico.

Especificaciones Relevantes para el Proyecto:

- Conectividad: Wi-Fi 802.11 b/g/n integrado, esencial para transmitir las solicitudes de validación de PIN hacia el servidor Java y la base de datos en tiempo real.
- Voltaje de Operación: Lógica de 3.3V/5V.
- GPIOs (Pines de Entrada/Salida): Provee suficientes puertos digitales para gestionar simultáneamente el barrido del teclado matricial y la señal de control para el actuador.
- Función en el Sistema:

1. Escanear constantemente el teclado matricial para detectar pulsaciones.
2. Empaquetar la credencial (PIN) y enviarla al backend.
3. Interpretar la respuesta del servidor (APROBADO/DENEGADO) para activar o mantener cerrada la cerradura.

4.2 Componentes Utilizados

Además del microcontrolador, el sistema integra periféricos de entrada y salida que permiten la interacción con el usuario y el control del mecanismo de seguridad.

4.2.1 Teclado Matricial

Es el dispositivo de entrada primario (HMI - Interfaz Hombre Máquina) que permite al usuario o administrador ingresar sus credenciales numéricas.

- Principio de Funcionamiento: Utiliza una disposición de matriz de interruptores (4 filas x 4 columnas). El ESP32 realiza un "barrido" o escaneo rápido, poniendo en alto secuencialmente las filas y leyendo las columnas para determinar qué tecla ha sido presionada.
- Interfaz: Conexión de 8 hilos a los pines digitales del ESP32.
- Manejo de Rebote (Debounce): Se implementa lógica de software (o condensadores de hardware) para evitar que una sola pulsación sea interpretada como múltiples entradas debido al ruido mecánico de las teclas.

4.2.2 Cerradura Electrónica / Actuador

El mecanismo de seguridad física es una cerradura electromecánica (servomotor)que permanece en estado de bloqueo por defecto (Fail-Secure).

- Funcionamiento: El servomotor se activa tras que se haya validado el pin, en donde se vera la acción de movimiento que simula el bloqueo y desbloqueo del casillero.
- Etapa de Potencia (Driver): Dado que el ESP32 opera a 5V, no puede conectar directamente la cerradura (que usualmente requiere 12V/1A). Se utiliza un circuito intermediario compuesto por:

o Relé o Transistor MOSFET: Actúa como un interruptor controlado. El ESP32 envía una señal lógica baja/alta a la base del transistor o bobina del relé para cerrar el circuito de alta potencia que alimenta el solenoide.

o Diodo Flyback: Protección esencial instalada en paralelo al solenoide para evitar que el voltaje inverso generado al desconectar la bobina dañe el microcontrolador.

5 Comunicación ESP32 – Aplicación

La comunicación entre el ESP32 y la aplicación se realiza mediante solicitudes controladas, donde el dispositivo envía datos como PIN, Token y eventos de acceso.

El backend recibe la información, valida los datos, ejecuta la lógica correspondiente y devuelve una respuesta indicando el resultado de la operación. De esta forma, toda la validación y seguridad se centraliza en el servidor, evitando decisiones críticas en el dispositivo.

6 Modelo de Datos

El sistema utiliza una base de datos SQLite para almacenar la información necesaria para el control y auditoría de accesos.

Las principales entidades son:

- **Usuarios:** almacenan información de estudiantes y administradores.
- **Casilleros:** representan los casilleros físicos y su estado.
- **Eventos:** registran cada intento de acceso, fallos y bloqueos.
- **Tokens:** permiten accesos temporales para recuperación.

Este modelo garantiza trazabilidad completa de las acciones realizadas en el sistema.

7 Diseño UML del Sistema

El sistema se modela mediante diagramas UML para representar su comportamiento y estructura.

El diagrama de casos de uso muestra las acciones disponibles para estudiantes y administradores.
El diagrama de clases describe la estructura interna del sistema y la relación entre entidades, servicios y componentes.

Se emplean relaciones de abstracción, dependencia, composición y herencia para representar correctamente la organización del sistema.

8 Seguridad del Sistema

La seguridad del sistema *Casillero Inteligente* se basa en la protección de credenciales, control de accesos, registro de eventos y mecanismos de recuperación, evitando el uso de claves en texto plano y reduciendo el riesgo de accesos no autorizados tanto para estudiantes como para administradores.

8.1 Gestión del PIN

Cada casillero está asociado a un PIN personal de 5 dígitos, utilizado por el estudiante para validar el acceso.
El sistema valida la longitud y formato del PIN antes de procesarlo, evitando entradas inválidas.
El PIN nunca se utiliza directamente para tomar decisiones críticas sin pasar previamente por los mecanismos de validación y seguridad definidos en la lógica de negocio.

8.2 Encriptación del PIN para Seguridad

El PIN nunca se almacena ni se transmite en texto plano.

Se aplica hashing con salt antes de su almacenamiento, y la validación se realiza comparando únicamente valores encriptados, garantizando confidencialidad ante accesos no autorizados a la base de datos.

8.3 Seguridad del Modo Administrador

El administrador opera bajo un esquema de privilegios controlados. No existe una clave maestra directa; las acciones administrativas se limitan a gestión, recuperación y auditoría, y todas quedan registradas para asegurar trazabilidad.

8.4 Manejo de Tokens de Recuperación

Ante el bloqueo de un casillero, el sistema permite generar un token de recuperación temporal y encriptado.

Este token se utiliza únicamente para desbloqueo o reasignación de PIN y pierde validez tras su uso.

8.5 Bloqueo por Intentos Fallidos

Después de un número máximo de intentos incorrectos, el casillero se bloquea automáticamente.

El desbloqueo solo puede realizarse mediante un proceso administrativo controlado, previniendo ataques por fuerza bruta.

8.6 Registro de Eventos de Seguridad

El sistema registra eventos como accesos, fallos, bloqueos, desbloqueos y acciones administrativas. Cada evento almacena información clave para auditoría y seguimiento, fortaleciendo el control y la confiabilidad del sistema.

9 Manejo de Errores y Excepciones

9.1 Estrategia de Manejo de Errores

El sistema sigue una arquitectura de "Defensa en Profundidad" y "Fallo Seguro" (Fail-Safe), regida por los siguientes principios:

1. Propagación Controlada: Los errores de bajo nivel son capturados en la capa de servicio de Java. Nunca se exponen detalles técnicos ("Stack Trace") al usuario final (Estudiante) ni al Administrador en la interfaz gráfica; en su lugar, se muestran mensajes amigables (ej: "Ups, Ocurrió un Error

2. Resiliencia en Comunicación: Dado que la comunicación entre el ESP32 y el servidor depende de la red Wi-Fi, se implementan tiempos de espera (Timeouts)

9.2 Excepciones Controladas

El backend en Java 25 ha sido diseñado para gestionar errores que transcurren al momento de realizar la acción.

- Bloqueo por Intentos:

o Causa: Se dispara cuando un usuario supera el límite de intentos fallidos configurado (Use Case: Bloquear acceso por intentos fallidos).

o Acción del Sistema: No es un "error" , sino una regla de negocio. Se lanza esta excepción para abortar el proceso de validación y registrar inmediatamente el evento en la tabla de auditoría con estado "BLOQUEADO".

9.3 Registro de Errores en Logs

A diferencia de los "Eventos de Auditoría" (que registran quién entra), el "Registro de Errores" (Log Técnico) almacena qué falló en el sistema.

El sistema utiliza la librería estándar de Logging de Java (java.util.logging o similar implementado en Java 25) para generar archivos de texto diarios en el servidor.

Estructura del Log: Cada entrada de error contiene la siguiente metadata para facilitar la depuración:

1. Timestamp: Fecha y hora exacta del fallo (ISO 8601).

2. Origen: Clase y método donde ocurrió la excepción (ej. AuthService.validarPin()).

3. Mensaje y StackTrace: Descripción técnica del error.

10 Inicialización del Sistema

El proceso de arranque debe seguir un orden estricto para evitar excepciones de conexión o pérdida de eventos de auditoría.

Paso 1: Inicio de la Capa de Persistencia

1. Inicie el servicio de la Base de Datos.

2. Verifique la conectividad y que las tablas relacionadas con Usuarios, PIN y Logs de Auditoría estén accesibles.

Paso 2: Ejecución del Aplicativo Java (Servidor)

1. Desde la terminal o consola de comandos, navegue al directorio del proyecto.

2. Ejecute el archivo JAR compilado o la clase principal mediante el comando:

Bash

java --enable-preview -jar CasilleroInteligente_v1.0.jar

(Nota: El flag --enable-preview puede ser necesario dependiendo de las características de Java 25 utilizadas).

3. Verificación: La consola debe mostrar el mensaje: "Sistema a la espera de conexiones. Servicio de Auditoría iniciado." (o Similar) Esto habilita los casos de uso de backend como Registrar eventos de auditoría y Consultar Eventos.

Paso 3: Energización del Hardware (ESP32)

1. Conecte el sistema del Casillero Inteligente a la fuente de energía.

2. El ESP32 iniciará su secuencia de boot.

3. Validación de Conexión:

o El dispositivo intentará conectarse a la red Wi-Fi configurada.

o Una vez conectado, establecerá comunicación con el "Tercero" (Middleware).

Paso 4: Pruebas de Arranque (Smoke Test)

Para confirmar que el despliegue fue exitoso, realice la siguiente secuencia rápida:

1. Prueba de Administrador: En la App Java, intente ejecutar Autenticar como administrador. Si el acceso es concedido, la conexión a la BD es correcta.

2. Prueba de Comunicación: En el teclado físico del locker, ingrese un PIN de prueba erróneo.

o El sistema Java debe recibir la solicitud Validar PIN.

o El sistema debe retornar la señal de Denegar acceso.

o Se debe generar un nuevo registro en la base de datos bajo el caso de uso Registrar eventos de auditoría (Acción: Denegar Acceso).

11 Conclusiones Técnicas

11.1 Evaluación del Sistema

Tras la implementación y despliegue de la arquitectura basada en ESP32 y Java 25, se han alcanzado las siguientes conclusiones respecto al rendimiento y la estabilidad:

• Desacoplamiento Efectivo (Hardware/Software): La decisión de delegar la lógica de negocio (Validar PIN, Registrar eventos de auditoría) al servidor Java, manteniendo el ESP32 únicamente como una interfaz física de entrada/salida, ha resultado exitosa. Esto reduce la carga de procesamiento en el microcontrolador y permite actualizar las reglas de acceso (ej. bloquear usuarios) sin necesidad de reprogramar el firmware del dispositivo.

• Integridad y Seguridad de Datos: El módulo de auditoría (Registrar eventos de auditoría) cumple eficazmente su función de trazabilidad. Al centralizar los intentos de acceso (exitosos y fallidos) en la base de datos.

• Escalabilidad del Backend: El uso de Java 25 proporciona un entorno de ejecución robusto y preparado para el futuro.

• Gestión de Roles: La separación estricta entre los casos de uso del Estudiante (operativos) y el Administrador (gestión y recuperación) ha simplificado el mantenimiento del código y reforzado la seguridad.

11.2 Posibles Mejoras Futuras

Aunque el sistema es funcional y cumple con los requisitos iniciales, se identifican las siguientes áreas de oportunidad para versiones posteriores (v2.0+):

- Redundancia y Modo "Offline":

o Situación actual: El sistema depende de la conexión constante entre el ESP32 y el Servidor para validar el PIN.

o Mejora: Implementar una memoria caché local en el ESP32 (usando SPIFFS/LittleFS) que almacene temporalmente los hashes de los PINs autorizados.

- Notificaciones en Tiempo Real (Push):

o Mejora: Implementar WebSockets en el servidor Java para enviar alertas inmediatas a la App del Administrador cuando ocurra un evento crítico.

12 Anexos

12.1 Diagramas

Diagrama de Caso de Uso

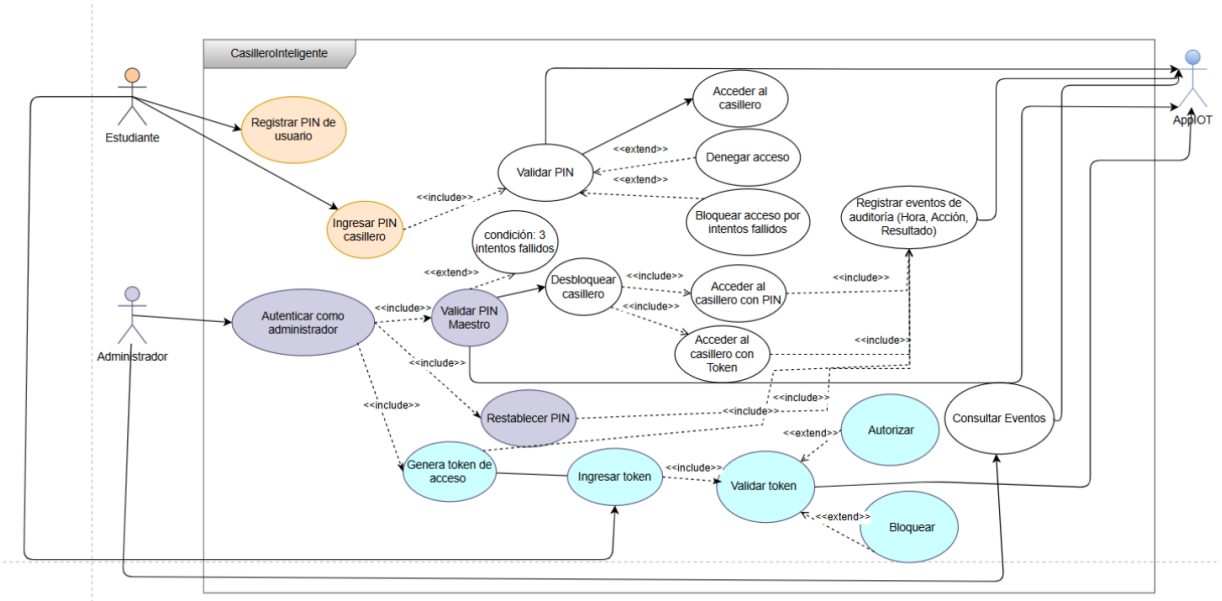
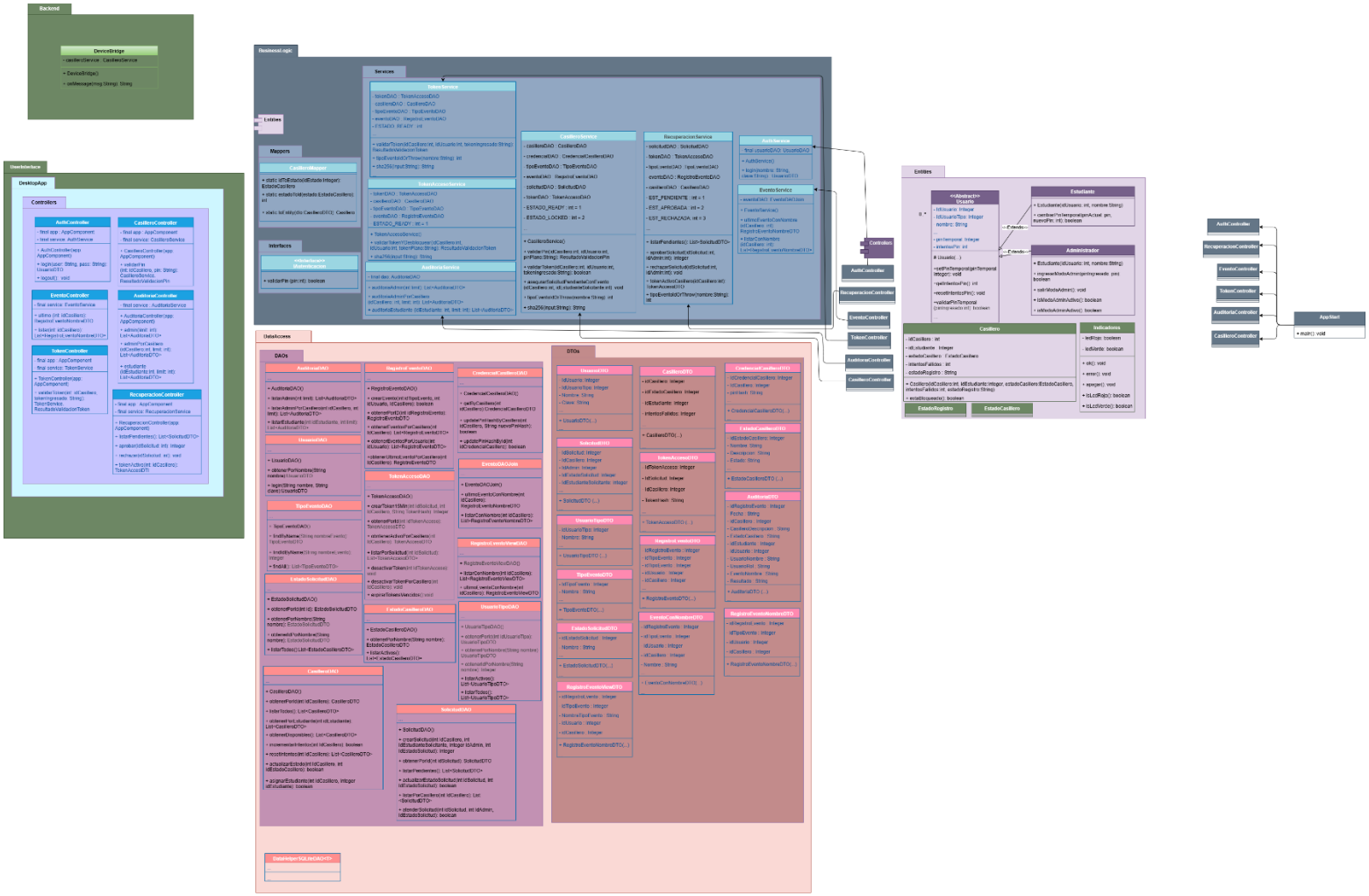
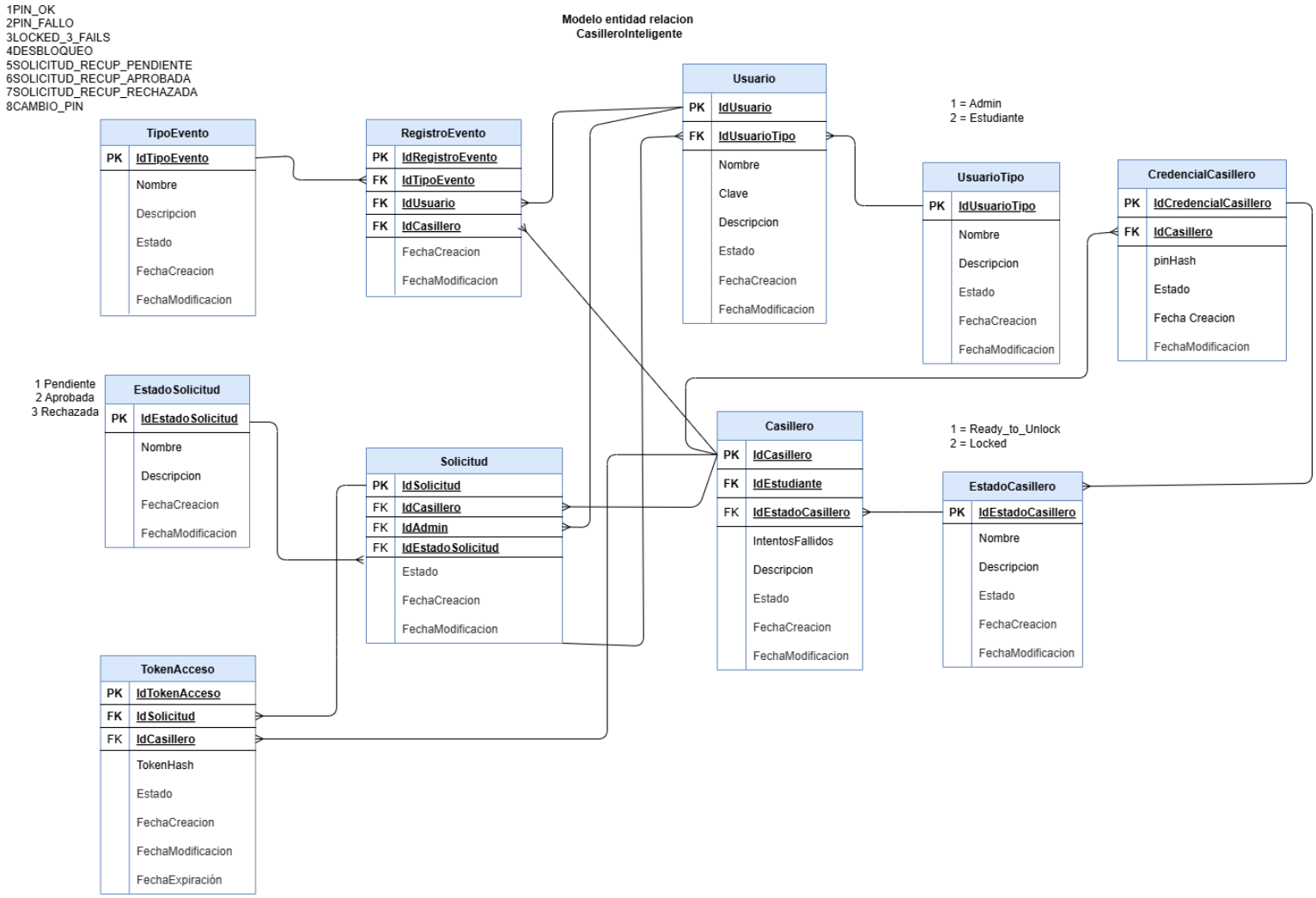


Diagrama de Clase



MER


1PIN_OK
2PIN_FALLO
3LOCKED_3_FAILS
4DESBLOQUEO
5SOLICITUD_RECUP_PENDIENTE
6SOLICITUD_RECUP_APROBADA
7SOLICITUD_RECUP_RECHAZADA
8CAMBIO_PIN
Modelo entidad relacion
CasilleroInteligente
Usuario
PK IdUsuario
FK IdUsuarioTipo
Nombre
Clave
Descripcion
Estado
FechaCreacion
FechaModificacion
1 = Admin
2 = Estudiante
UsuarioTipo
PK IdUsuarioTipo
Nombre
Descripcion
Estado
FechaCreacion
FechaModificacion
CredencialCasillero
PK IdCredencialCasillero
FK IdCasillero
pinHash
Estado
Fecha Creacion
FechaModificacion
TipoEvento
PK IdTipoEvento
Nombre
Descripcion
Estado
FechaCreacion
FechaModificacion
RegistroEvento
PK IdRegistroEvento
FK IdTipoEvento
FK IdUsuario
FK IdCasillero
FechaCreacion
FechaModificacion
1 Pendiente
2 Aprobada
3 Rechazada
EstadoSolicitud
PK IdEstadoSolicitud
Nombre
Descripcion
FechaCreacion
FechaModificacion
Solicitud
PK IdSolicitud
FK IdCasillero
FK IdAdmin
FK IdEstadoSolicitud
Estado
FechaCreacion
FechaModificacion
Casillero
PK IdCasillero
FK IdEstudiante
FK IdEstadoCasillero
IntentosFallidos
Descripcion
Estado
FechaCreacion
FechaModificacion
1 = Ready_to_Unlock
2 = Locked
EstadoCasillero
PK IdEstadoCasillero
Nombre
Descripcion
Estado
FechaCreacion
FechaModificacion
TokenAcceso
PK IdTokenAcceso
FK IdSolicitud
FK IdCasillero
TokenHash
Estado
FechaCreacion
FechaModificacion
FechaExpiración

Arquitectura