GUIA 3

MODELO C4 Y PRINCIPIOS SOLID

MIGUEL ANGEL ROMERO ALFONSO

HEMER SANTIAGO PEREZ NIEVES

ARQUITECTURA DE SOFTWARE

INGENEIRIA DE SOFTWARE

2025-1

1. **Guiding Questions**

Which one of the following options is needed in the lifecycle of software architecture process:  
1. Investigation  
2. Analysis  
3. prerequisites  
4. Induction  
Which of the following are correct statements:  
1. An architecture may or may not defined components  
2. An architecture is not dependable on requirements  
3. An architecture is foremost an abstraction of a system that suppresses details of the components that do not affect how they are used.  
4. Invocation based programming sequences.

1. **Autonomous Work Activity**
2. Interfaz de usuario gráfica

   El contenido generado por IA puede ser incorrecto.

**3. Procedure and Methodology of practice**

**Modelo de Arquitectura del ERP Escolar**

**Descripción de Funcionalidades**

El ERP Escolar centraliza y automatiza la gestión administrativa y académica en instituciones educativas. Sus principales funcionalidades incluyen:

**Gestión de Estudiantes:** Registro, historial académico, asistencia y seguimiento de desempeño.

**Gestión de Docentes:** Asignación de cursos, control de horarios y evaluaciones.

**Administración Académica:** Planificación de clases, calificaciones y reportes.

**Gestión Financiera:** Control de pagos, matrículas y generación de facturas.

**Comunicación Interna:** Notificaciones entre docentes, estudiantes y padres.

**Recursos Humanos:** Contratación, nóminas y permisos.

**Mantenimiento Institucional:** Gestión de infraestructura y recursos.

**Trade-offs**

|  |  |  |
| --- | --- | --- |
| Aspecto | Beneficio | Compromiso |
| Simplicidad vs. Escalabilidad | Fácil desarrollo y mantenimiento | Puede ser difícil escalar para grandes instituciones |
| Costo vs. Seguridad | Tecnologías económicas y de fácil acceso | Menos medidas avanzadas de seguridad |
| Velocidad vs. Funcionalidad | Rápido desarrollo con Frameworks sencillos | Algunas funcionalidades avanzadas pueden requerir más tiempo |
| Almacenamiento en la nube vs. Local | Acceso remoto y mantenimiento simplificado | Dependencia de una conexión estable a Internet |

**C. Tecnologías Seleccionadas**

**Backend:** PHP con Laravel

**Frontend:** HTML, CSS y JavaScript (Vanilla o Bootstrap)

**Base de Datos:** MySQL

**Hosting:** Hostinger

**Control de Versiones:** GitHub

**Diagramas del Modelo C4**

Diagrama de Contexto:

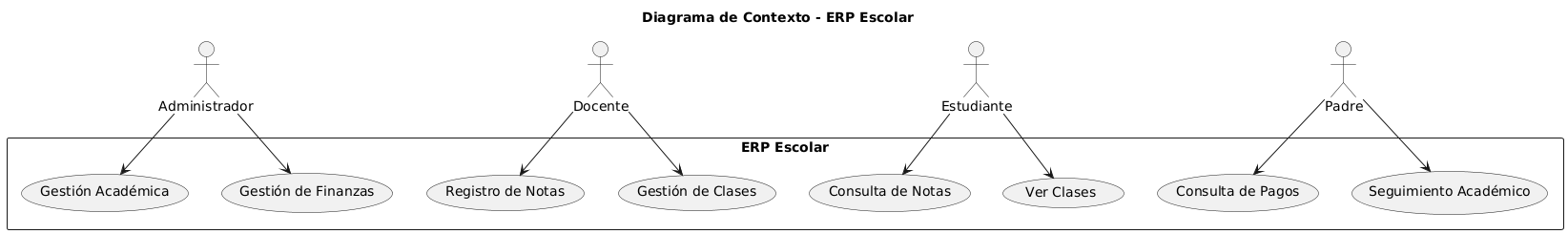


Diagrama de Contenderes:

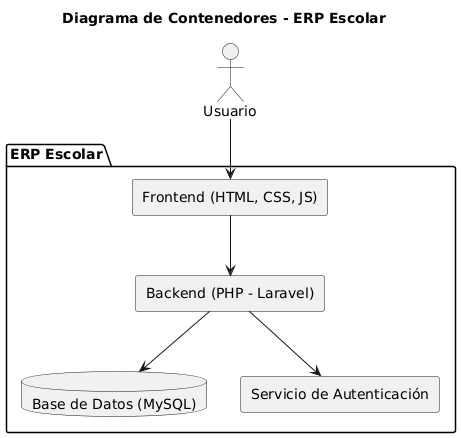


Diagrama de Componentes:

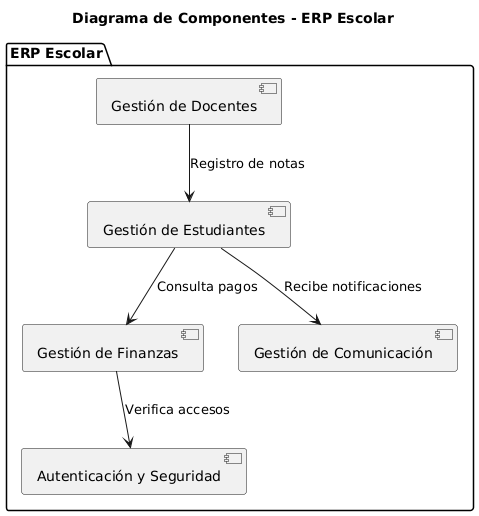
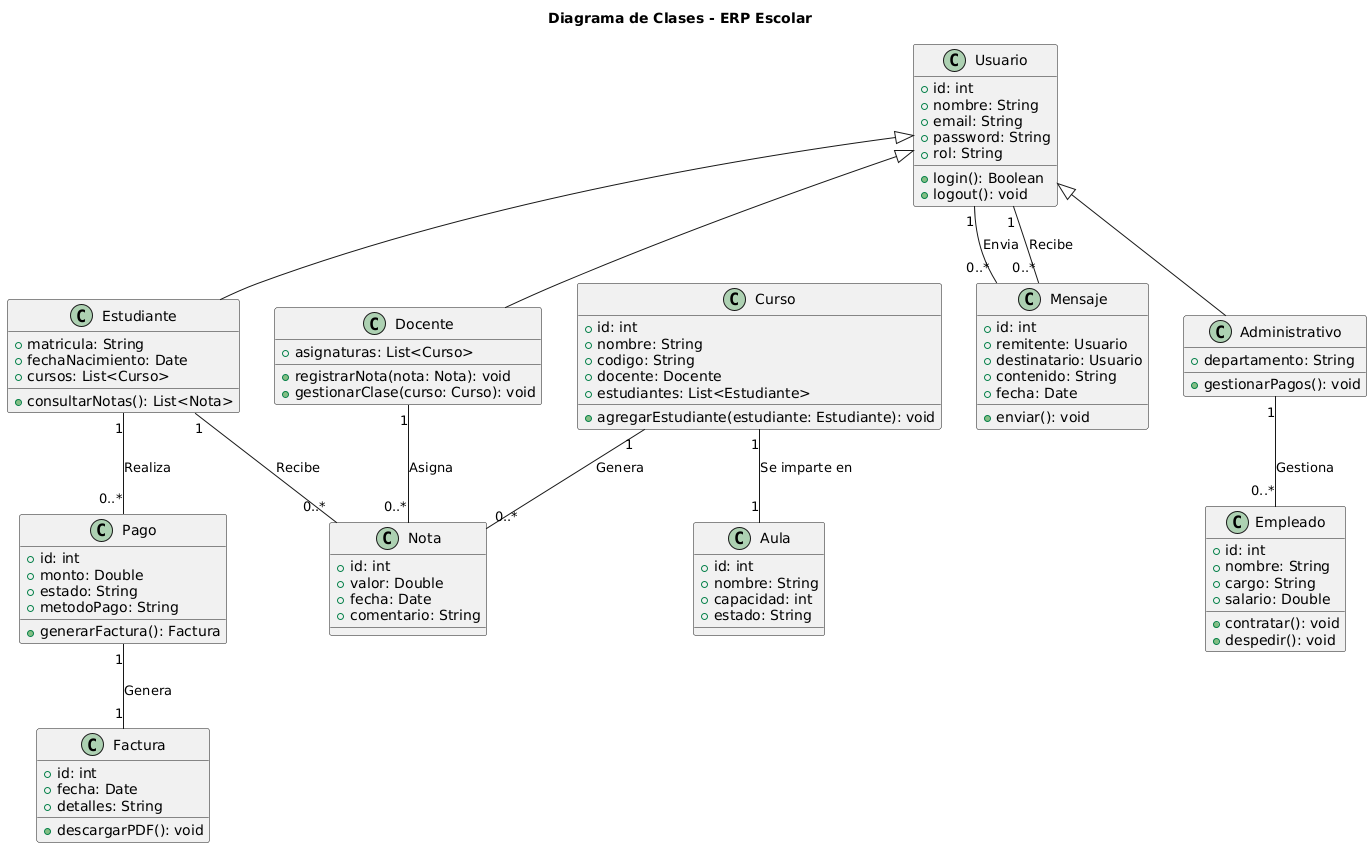


Diagrama de código:



**PATRONES DE DISEÑO**

**Patrones de Diseño Aplicados**

En el desarrollo del ERP Escolar, se implementan patrones de diseño que garantizan escalabilidad, mantenibilidad y modularidad en el sistema.

**Patrón Modelo-Vista-Controlador (MVC)**

Descripción: Este patrón separa la lógica de negocio, la presentación y la manipulación de datos en capas independientes.

* Modelo: Define la estructura de los datos y la lógica de negocio (Ejemplo: clases Estudiante, Docente, Curso).
* Vista: Interfaz de usuario y representación de datos (HTML, CSS, JavaScript).
* Controlador: Maneja las solicitudes del usuario y actualiza el modelo y la vista (Ejemplo: EstudianteController).

**Patrón Factory Method**

Descripción: Permite la creación de objetos sin especificar su clase concreta.

* Ejemplo de Aplicación: UsuarioFactory instancia objetos de Estudiante, Docente o Administrativo según el tipo de usuario.

**Patrón Singleton**

Descripción: Asegura que una clase tenga una única instancia y proporciona un punto de acceso global a ella.

* Ejemplo de Aplicación: ConexionBD maneja una única conexión a la base de datos para mejorar el rendimiento y evitar conexiones innecesarias.

**Patrón Repository**

Descripción: Proporciona una capa de abstracción entre la base de datos y la lógica de negocio, facilitando la manipulación de datos.

* Ejemplo de Aplicación: CursoRepository, PagoRepository, encapsulan la lógica de acceso a MySQL y separan la gestión de la base de datos del resto del sistema.

**Patrón Observer**

Descripción: Define una relación de dependencia entre objetos, de modo que cuando uno cambia su estado, sus dependientes son notificados automáticamente.

* Ejemplo de Aplicación: NotificacionService envía alertas automáticas a los estudiantes cuando se publican calificaciones o hay novedades administrativas.

**Principios SOLID Aplicados**

Para garantizar una arquitectura robusta, el ERP Escolar sigue los principios SOLID, asegurando la flexibilidad y facilidad de mantenimiento del código.

**Principio de Responsabilidad Única (SRP)**

Cada clase debe tener una única responsabilidad.

* Ejemplo de Aplicación: PagoService maneja la lógica de pagos, mientras que FacturaService se encarga de generar comprobantes, evitando que una sola clase realice múltiples funciones.

**Principio Abierto/Cerrado (OCP)**

El software debe estar abierto para su extensión, pero cerrado para modificación.

* Ejemplo de Aplicación: La clase Usuario permite agregar nuevos tipos (Estudiante, Docente, Administrativo) sin necesidad de modificar la clase base.

**Principio de Sustitución de Liskov (LSP)**

Los objetos de una subclase deben poder reemplazar a los de la clase base sin alterar la funcionalidad.

* Ejemplo de Aplicación: Estudiante y Docente heredan de Usuario, y pueden ser utilizados en cualquier parte del sistema donde se espere un Usuario, sin causar errores.

**Principio de Segregación de Interfaces (ISP)**

Las interfaces deben ser específicas y no obligar a las clases a implementar métodos que no necesitan.

* Ejemplo de Aplicación: Se definen interfaces específicas como ICalificaciones para docentes y IPagos para administrativos, evitando métodos innecesarios en clases que no los usan.

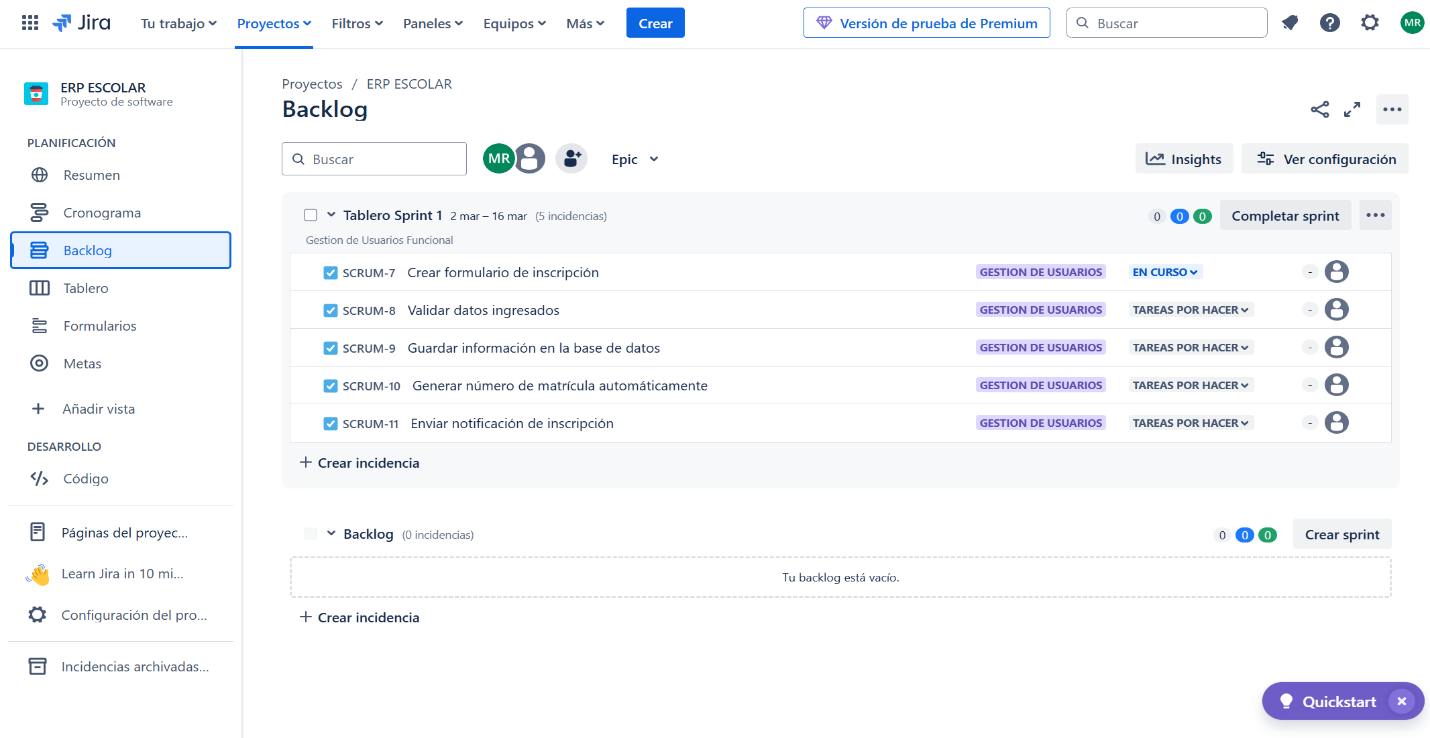
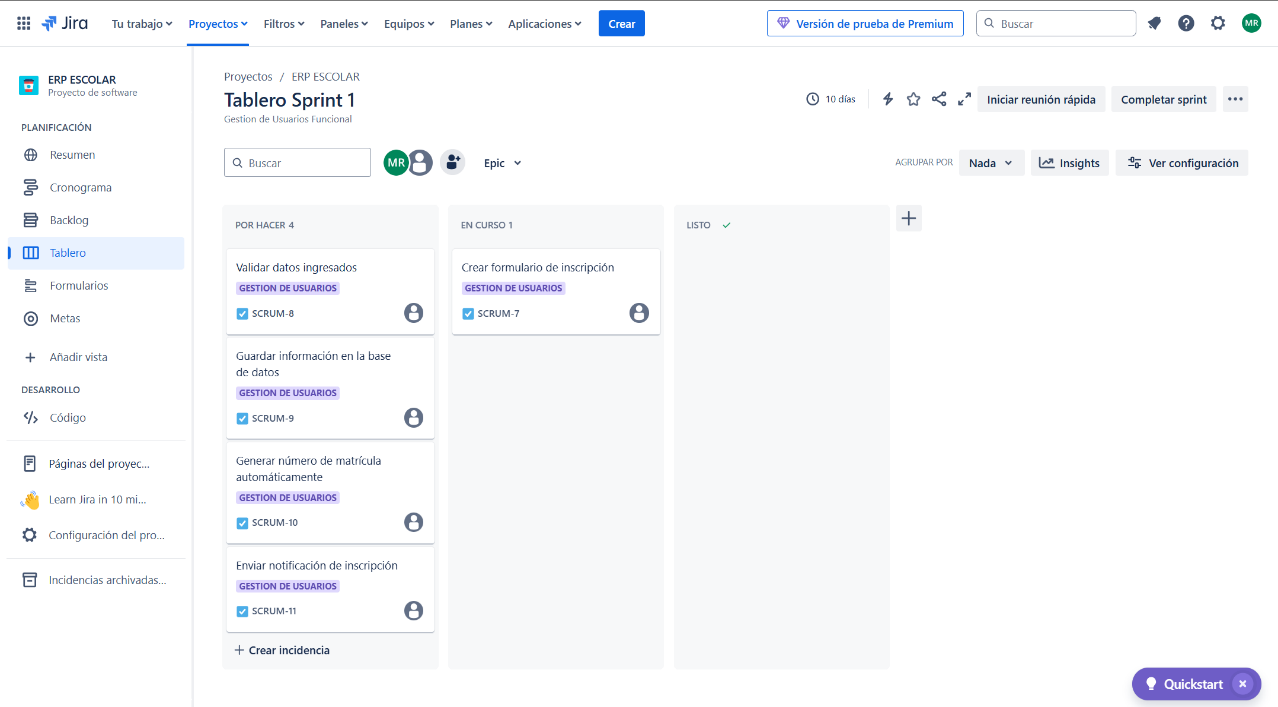
**Principio de Inversión de Dependencias (DIP)**

Los módulos de alto nivel no deben depender de módulos de bajo nivel, sino de abstracciones.

* Ejemplo de Aplicación: CursoService depende de ICursoRepository en lugar de una implementación concreta, permitiendo cambios en la base de datos sin afectar la lógica de negocio.

**PLANEACION DE PRUEBAS DEL SPRINT-1**

Se implemento la plataforma JIRA para la Gestion de Incidencias, Sprint Planing, Product Backlog

****