DESARROLLAR LAS ARQUITECTURA DE SOFTWARE DE ACUERDO AL PATRÓN DE DISEÑO SELECCIONADO

GA4-220501095

ANDRÉS ALBERTO BUILES MUÑOZ

INSTRUCTOR

JHON ALEJANDRO NIÑO TAMBO

CENTRO METALMECÁNICO

SERVICIO NACIONAL DE APRENDIZAJE

SENA

MARZO 16, 2025

TABLA DE CONTENIDO

INTRODUCCIÓN

La arquitectura de Software para un sistema escolar generalmente incluye un diseño en capas, como la arquitectura en 3 niveles que abarca la interfaz de usuario, la lógica de negocio y la persistencia de datos. Esto permite una mejor organización y gestión de las diferentes áreas del colegio, facilitando la integración de módulos y funcionalidades

PATRONES DE DISEÑO

Los patrones de diseño son soluciones reutilizables a problemas comunes en el desarrollo del Software. En el contexto de un colegio, estos patrones pueden aplicarse para tener una mejora en la organización, la gestión de recursos y la comunicación

Vamos a explicar algunos patrones de diseño que podrían ser útiles en un entorno escolar:

- **Modelo-Vista-Controlador (MVC)**
  - **Descripción:** Separa la lógica de negocio (**Modelo**) , la interfaz de usuario (**Vista**) y la gestión de la interacción del usuario (**Controlador**)
  - **Aplicación:** Puede ser utilizado en el desarrollo de aplicaciones web para la gestión de estudiantes, calificaciones y asistencia

- **Singleton**
  - **Descripción:** Asegura que una clase tenga una única instancia y proporciona un punto de acceso global a ella
  - **Aplicación:** Ideal para gestionar la configuración de la aplicación o el acceso a recursos compartidos, como una base de datos

- **Fábrica (Factory)**
  - **Descripción:** Define una interfaz para crear un objeto, pero permite que las subclases alteren el tipo de objeto que se crea
  - **Aplicación:** Puede ser utilizado para crear diferentes tipos de usuarios (estudiantes, profesores, administradores) en un sistema de gestión escolar

- **Observador (Observer)**
  - **Descripción:** Define una dependencia uno a muchos entre objetos, de modo que cuando un objeto cambia de estado, todos sus dependientes son notificados y actualizados automáticamente

- **Aplicación:** Útil para notificar a los estudiantes y padres sobre cambios en las calificaciones, eventos escolares o anuncios importantes

- **Comando (Command)**
  - **Descripción:** Encapsula una solicitud como un objeto, lo que permite parametrizar a los clientes con diferentes solicitudes, encolar o registrar solicitudes y soportar operaciones que se pueden deshacer
  - **Aplicación:** Puede ser utilizado para implementar acciones en un sistema de gestión de tareas, como asignar tareas a estudiantes o profesores

- **Estrategia (Strategy)**
  - **Descripción:** Define una familia de algoritmos, encapsula cada uno y los hace intercambiables
  - **Aplicación:** Puede ser utilizado para implementar diferentes métodos de evaluación o calificación en un sistema educativo

- **Decorator**
  - **Descripción:** Permite agregar funcionalidad a un objeto de manera dinámica
  - **Aplicación:** Puede ser utilizado para añadir características adicionales a los cursos, como materiales complementarios o tutorías

- **Facade**
  - **Descripción:** Proporciona una interfaz simplificada a un conjunto de interfaces en un subsistema
  - **Aplicación:** Puede ser utilizado para crear una interfaz sencilla para que los estudiantes accedan a diferentes servicios, como calificaciones, horarios y recursos educativos

- **Proxy**
  - **Descripción:** Proporciona un sustituto o representante de otro objeto para controlar el acceso a este
  - **Aplicación:** Puede ser utilizado para gestionar el acceso a recursos limitados, como laboratorios de computación o bibliotecas

- **Chain of Responsibility**
  - **Descripción:** Permite que varios objetos manejen una solicitud sin que el remitente de la solicitud sepa cuál objeto la manejará
  - **Aplicación:** Puede ser útil en la gestión de quejas o solicitudes de los estudiantes, donde diferentes niveles de administración pueden manejar la solicitud

VISTA DE COMPONENTES

Un colegio generalmente se compone en:

- **Áreas Académicas:** Aulas, Laboratorios de Ciencias, Salas de Informática, Bibliotecas
- **Zonas Administrativas:** Oficinas de Dirección, Sala de profesores, Áreas de recepción
- **Espacios Recreativos:** Parques, Patios, Canchas Deportivas, Auditorios
- **Áreas de Apoyo:** Cafeterías, Enfermería, Salones para reuniones o Actividades Extracurriculares
- **Infraestructura Técnica:** Servicios de Mantenimiento, Almacenamiento de materiales y equipos

IMPLEMENTACIÓN EN LOS PATRONES EN LOS COMPONENTES

La implementación en los patrones en los componentes de un sistema de gestión escolar se puede mejorar en la organización, la mantenibilidad y la escalabilidad del Software

En este caso vamos a describir cuales son los patrones de diseño más comunes y cómo podrían aplicarse en un sistema de gestión de colegio

- **Patrón MVC (Modelo-Vista-Controlador)**
  - **Descripción:** Este patrón separa la lógica de negocio (Modelo), la interfaz de usuario (Vista) y la lógica de control (Controlador)
  - Implementación
    1. **Modelo:** Clases que representan entidades del colegio, como **Estudiante**, **Profesor**, **Curso**, etc. Estas clases manejarían la lógica de negocio y la interacción con la base de datos
    2. **Vista:** Interfaces de usuario que muestran la información a los usuarios. Podrían ser páginas web, aplicaciones móviles, etc
    3. **Controlador:** Clases que manejan las solicitudes del usuario, interactúan con el modelo y actualizan la vista. Por ejemplo, un **"EstudianteController"** que maneje las operaciones relacionadas con los estudiantes

- **Patrón Singleton**
  - **Descripción:** Asegura que una clase tenga una única instancia y proporciona un punto de acceso global a ella
  - **Implementación:** Podrías usar este patrón para la clase de configuración del sistema, como **"ConfiguracionSistema"**, que carga y proporciona configuraciones globales del colegio (por ejemplo, parámetros de conexión a la base de datos, configuraciones de correo, etc.)

- **Patrón Factory**
  - **Descripción:** Proporciona una interfaz para crear objetos en una superclase, pero permite a las subclases alterar el tipo de objetos que se crearán
  - **Implementación:** Una "**Factory"** para crear diferentes tipos de usuarios (por ejemplo, "**Usuario Factory"** que puede crear **Estudiante**, **Profesor**, **Administrador**, etc.) según el rol que se necesite

- **Patrón Observer**
  - **Descripción:** Define una dependencia uno a muchos entre objetos de tal manera que cuando un objeto cambie de estado, todos sus dependientes sean notificados y actualizados automáticamente
  - **Implementación:** En un sistema de gestión escolar, podrías tener un sistema de notificaciones donde los estudiantes se suscriben a cambios en sus cursos. Cuando hay un cambio (como una nueva tarea o un cambio de horario), los estudiantes son notificados automáticamente

- **Patrón Strategy**
  - **Descripción:** Permite definir una familia de algoritmos, encapsular cada uno de ellos y hacerlos intercambiables
  - **Implementación:** Podrías tener diferentes estrategias de evaluación para los estudiantes (por ejemplo, **Evaluación Por Examen**, **Evaluación Por Proyecto**, etc.). La clase **Curso** podría tener un método que use una estrategia de evaluación específica según el tipo de curso

- **Patrón Repository**
  - **Descripción:** Proporciona una colección de objetos de dominio en memoria y encapsula la lógica de acceso a datos
  - **Implementación:** Crear un "**Estudiante Repository"** que maneje todas las operaciones de acceso a datos relacionadas

con los estudiantes, como agregar, eliminar, buscar y actualizar estudiantes en la base de datos

VISTA DE DESPLIEGUE

La vista de despliegue de un Software para colegios es una representación de cómo se distribuyen los diferentes componentes del sistema en la infraestructura de Hardware y Red. Esta vista ayuda a entender cómo se ejecuta el Software en servidores, dispositivos y redes

- **Servidores**
  - Servidor de Aplicación (Para lógica de negocio)
  - Servidor de Base de Datos (Almacena información de alumnos, docentes, calificaciones, etc.)
  - Servidor Web (Si el sistema es accesible desde Internet)

- **Clientes o Dispositivos de Acceso**
  - Computadoras de administrativos
  - Tablets o Laptops de estudiantes
  - Dispositivos móviles con aplicaciones conectadas

- **Redes y Conectividad**
  - Internet o VPN para acceso remoto
  - Red Local del colegio (LAN)
  - Servidores en la nube (AWS, Azure, Google Cloud) o en servidores físicos locales

- **Componentes de Software**
  - Aplicación web o móvil para estudiantes, docentes y administrativos
  - API para integración con otros sistemas
  - Módulos como gestión académica, asistencia, calificaciones, pagos y comunicación

HERRAMIENTAS UTILIZADAS

- **Herramientas de creación de diagramas y visualización**
  - **Microsoft Vision:**
    - Ideal para diagramas técnicos y empresariales
    - Permite crear representaciones claras de la infraestructura y los componentes
  - **Lucidchart:**
    - Herramienta en línea colaborativa para crear diagramas UML, de despliegue, de flujo, etc
    - Fácil de usar y accesible desde cualquier lugar
  - **Draw.io:**
    - Basada en texto para crear diagramas a partir de código
    - Útil para desarrolladores que prefieren trabajar en un entorno de programación
  - **PlantUML:**
    - Basada en texto para crear diagramas a partir de código
    - Útil para desarrolladores que prefieren trabajar en un entorno de programación
  - **SQL**

- **Herramientas para despliegue y modelado en la nube**
  - **AWS Architecture Icons:**
    - Para usuarios de Amazon Web Services, puedes diseñar diagramas usando sus plantillas y símbolos prediseñados
  - **Azure Architecture Center:**
    - Similar a la de AWS, pero para usuarios de Microsoft Azure
    - Ofrece guías y herramientas para estructurar servicios en la nube
  - **Google Cloud Diagrams:**

- Perfecto para representar arquitecturas que usan Google Cloud

➢ **Herramientas de gestión de infraestructura**

- **Terraform:**
  - Para describir infraestructuras como código.
  - Útil para automatizar y gestionar los despliegues en la nube

- **Ansible**
  - Herramienta para automatización de configuraciones y despliegues
- **Docker y Kubernetes**
  - Para la gestión y despliegue de contenedores en sistemas distribuidos
- **Nagios**
  - Monitoreo de la infraestructura para asegurar que todo funcione correctamente

➢ **Herramientas de colaboración y seguimiento**

- **Jira y Confluence**
  - Para documentar los diagramas y planificar tareas relacionadas con el despliegue
- **Slack o Microsoft Teams**
  - Para coordinar la implementación entre equipos

CONCLUSIÓN

La arquitectura de software en el contexto educativo, como en un colegio, es fundamental para garantizar que los sistemas de información y las aplicaciones utilizadas sean eficientes, escalables y fáciles de mantener

Una arquitectura de software bien diseñada en un colegio no solo mejora la eficiencia y la efectividad de los procesos educativos, sino que también contribuye a crear un entorno de aprendizaje más dinámico y seguro. La inversión en una buena arquitectura es, por lo tanto, una decisión estratégica que puede tener un impacto significativo en la calidad de la educación ofrecida