Universidad La Salle – Bolivia

Carrera de Ingeniería de Sistemas

Tema:

Arquitecturas de Desarrollo de Software

Estudiante:

Diego Carlos Rojas Apaza

Materia:

ISI- 356 Lenguaje Audiovisual

Docente:

Luis Jiménez Peña

Gestión:

II/23

Tarea: Arquitecturas de desarrollo de software más usadas en la actualidad y dar una explicación de cómo funcionan.

Bueno primeramente las arquitecturas de desarrollo de software son un conjunto de principios y patrones que se utilizan para diseñar y construir sistemas de software, estas arquitecturas proporcionan una estructura y un marco para el desarrollo de software.

Lo que ayuda a garantizar que los sistemas sean **escalables**, **flexibles** y **mantenibles**.

Las arquitecturas de desarrollo de software más usadas en la actualidad son:

- **Arquitectura en capas**: Esta arquitectura divide el sistema en capas funcionales, como la capa de **presentación**, la capa de **aplicación** y la capa de **datos**.
  Cada capa se comunica con las capas adyacentes a través de interfaces bien definidas.

  ¿Cómo funciona?
  La arquitectura en capas es una de las arquitecturas más simples y fáciles de entender, pues en este el sistema se divide en capas funcionales, cada una de las cuales tiene un propósito específico.

  - **La capa de presentación** se encarga de la interacción con el usuario.
  - **La capa de aplicació**n implementa la lógica de negocios.
  - **La capa de datos** almacena los datos.

  Las capas se comunican entre sí a través de interfaces bien definidas, algunas de las interfaces más utilizadas, o más comunes son:

  - **Interfaz de proceso de datos (DIF):** Proporciona un servicio de acceso a datos a las capas que los necesitan.

- **Interfaz de servicio de base de datos (DBSIF):** Proporciona una interfaz unificada para acceder a diferentes bases de datos.
- **Interfaz de servidor de aplicaciones (AIF):** Proporciona una interfaz para las capas de aplicación que necesitan interactuar con las capas de servicios de aplicación.
- **Interfaz de usuario (UI):** Proporciona una interfaz de usuario para que los usuarios interactúen con el sistema.
- **Interfaz de control de datos (DIC):** Proporciona un servicio de control de datos para la gestión de datos en el sistema.

Estas interfaces son una forma de **separar** las diferentes **capas del sistema** y de **definir la comunicación entre ellas**, al definir estas interfaces se **pueden especificar los servicios** que ofrecen y **cómo deben ser usados** por otras capas.
Esto permite que las capas se desarrollen de forma independiente y que se puedan intercambiar fácilmente.

- **Arquitectura orientada a servicios**: Esta arquitectura divide el sistema en **servicios autónomos** que interactúan entre sí a través de **interfaces definidas.**
  Los servicios pueden ser implementados en diferentes **tecnologías** y **plataformas**, lo que facilita su **escalabilidad** y **flexibilidad**.

  ¿Cómo funciona?
  La arquitectura orientada a servicios **divide el sistema en servicios autónomos**, cada servicio implementa una función específica del sistema. Los servicios se comunican entre sí a través de interfaces definidas llamadas interfaces de **acceso a servicios (SIA)**, en la que los servicios **interactúan** y se **comunican** entre sí, las más comunes pueden ser:

  - **Solicitud de servicios (SIF):** Es la interfaz que define como una solicitud debe ser formateada y enviada a un servicio.
  - **Respuesta de servicios (SIR):** Es la interfaz que define qué información debe devolverse cuando un servicio recibe una solicitud y qué formato debe tener esa información.

    - **Servicio de registro (SIR):** Es la interfaz que define cómo la información de registro y seguimiento de las transacciones se debe registrar y enviar desde un servicio a otro.
    - **Servicio de notificación (SIN):** Es la interfaz que define cómo las notificaciones y alertas deben ser enviadas desde un servicio a otro.
    - **Servicio de autenticación (SIA):** Es la interfaz que define cómo se debe autenticar un usuario o una aplicación antes de permitir el acceso a un servicio.

  Cada uno de estos tipos de **SIA** define cómo se deben **formatear** y **enviar** las **solicitudes** y **respuestas** entre los servicios, entablando la comunicación entre los servicios de forma estructurada y consistente,

  La arquitectura orientada a servicios ofrece varios beneficios, como:

    - **Flexibilidad:** Los servicios pueden ser implementados en diferentes tecnologías y plataformas, lo que facilita su escalabilidad y flexibilidad.
    - **Reusabilidad:** Los servicios pueden ser reutilizados en diferentes sistemas.
    - **Mantenibilidad:** Los servicios se pueden desarrollar, implementar y escalar de forma independiente.

- **Arquitectura de microservicios**: Esta arquitectura es una especialización de la **arquitectura orientada a servicios**, estos microservicios son **servicios pequeños** y **autónomos** que se pueden **desarrollar**, **implementar** y **escalar** de forma **independiente**.

  ¿Cómo funciona?
  La arquitectura de microservicios es una **especialización de la arquitectura orientada a servicios**, los **microservicios** son **servicios pequeños** y **autónomos** que se pueden **desarrollar**, **implementar** y **escalar** de forma **independiente**.

En una arquitectura de microservicios, podemos encontrar los siguientes microservicios:

- **Microservicio de autenticación:** es responsable de validar la identidad de los usuarios y permitir su acceso al sistema.
- **Microservicio de autorización:** es responsable de asignar permisos a los usuarios para realizar acciones específicas.
- **Microservicio de pagos:** es responsable de procesar operaciones de pago.
- **Microservicio de inventario:** es responsable de manejar la información relacionada al stock de los productos.
- **Microservicio de pedidos:** es responsable de manejar los pedidos de los usuarios.
- **Microservicio de pedido de envío:** es responsable de manejar los pedidos de envío.
- **Microservicio de gestión de clientes:** es responsable de manejar la información relacionada a los clientes dentro del sistema.
- **Microservicio de gestión de personal:** es responsable de manejar la información relacionada hacia los empleados.
- **Microservicio de gestión de productos:** es responsable de manejar la información relacionada con los productos que están dentro del sistema.

La arquitectura de microservicios ofrece varios beneficios, como:

- **Escalabilidad:** Los microservicios se pueden escalar de forma independiente, lo que facilita el crecimiento del sistema.
- **Flexibilidad:** Los microservicios se pueden implementar en diferentes tecnologías y plataformas, lo que facilita su adaptación a los cambios.
- **Mantenibilidad:** Los microservicios se pueden desarrollar, implementar y escalar de forma independiente.

- **Arquitectura hexagonal**: Esta arquitectura se basa en el **concepto de puertos** y **adaptadores**. **Los puertos** representan los **puntos de entrada** y **salida del sistema**, y los **adaptadores** se utilizan para **conectar el sistema con el mundo exterior**.

  ¿Cómo funciona?
  En la arquitectura hexagonal, los **puertos** son los puntos en donde un componente **interactúa con los demás**, y los **adaptadore**s son **las capas que permiten a cada componente comunicarse con los demás**. El **objetivo** de esta arquitectura es **separar la lógica de negocio** de los **detalles de implementación** de cada componente, **mejorando la calidad** del código y haciendo que sea más fácil de darle **mantenimiento**.

  Estos son algunos ejemplos de **puerto**s y **adaptadores** en la **arquitectura hexagonal**:

  - **Puertos**:
    - **Puerto de datos:** Es la interfaz que define como un componente se comunicará con otro componente para acceder y modificar los datos.
    - **Puerto de usuario:** Es la interfaz que define como un usuario interactúa con un componente.
  - **Adaptadores:**
    - **Adaptador de datos:** Es la clase que implementa el puerto de datos para conectar un componente con una fuente de datos.
    - **Adaptador de usuario:** Es la clase que implementa el puerto de usuario para permitir que un usuario interactúa con un componente.

  La arquitectura hexagonal ofrece varios beneficios, como:
  - **Flexibilidad:** El sistema se puede adaptar fácilmente a los cambios.
  - **Mantenibilidad:** El sistema es fácil de mantener y depurar.
  - **Portabilidad:** El sistema se puede implementar en diferentes tecnologías y plataformas.

- **Arquitectura basada en eventos**: Esta arquitectura se **basa en la comunicación entre eventos**. Los eventos **son notificaciones** que se **propagan a través del sistema**.

  ¿Cómo funciona?
  En la arquitectura basada en eventos, todos **los registros de trabajo son procesados a través de eventos**, un **evento** es una **notificación** que se **propaga a través del sistema** cuando **ocurre un cambio** en **algún estado del sistema** o de **la información**.

  Los **eventos** se envían a través **de la pila de eventos**, que es un **conjunto de componentes** de software que **trabajan juntos para procesar los eventos**.

  Cuando un **evento se envía** a **la pila de eventos**, los **procesadores de eventos escuchan** a través de **canales para detectar los eventos** que les interesan. Si un **procesador de eventos detecta un evento en un canal al que está suscrito, se activa y procesa el evento**.

  Estos son algunos ejemplos de **la comunicación de eventos**:

  - **Un procesador de eventos**: puede detectar cuando un usuario ha hecho clic en un botón y enviar un evento a la pila de eventos.
  - **Cuando se crea un registro de trabajo**: se envía un evento que se propaga a través de la pila de eventos hasta que un procesador de eventos detecta el evento y lo procesa.
  - **Si un registro de trabajo falla**: se envía un evento que se propaga a través de la pila de eventos hasta que un procesador de eventos lo detecta y toma medidas para manejar el error.
  - **Si un registro de trabajo tiene éxito**: se envía un evento que se propaga a través de la pila de eventos hasta que un procesador de eventos lo detecta y toma medidas para procesar el registro de trabajo.

Este tipo de arquitectura es útil para crear **sistemas distribuidos** que estén **separados en componentes** que funcionan como **eventos** y que se comunican entre sí a **través de la pila de eventos**.

La arquitectura basada en eventos ofrece varios beneficios, como:

- **Escalabilidad:** El sistema se puede escalar fácilmente, ya que los eventos se pueden propagar a través de diferentes canales.
- **Flexibilidad:** El sistema se puede adaptar fácilmente a los cambios.
- **Mantenibilidad:** El sistema es fácil de mantener y depurar.

Pero bueno al final la elección de la arquitectura adecuada para un proyecto de software depende de varios factores, como el tamaño y la complejidad del sistema, los requisitos de rendimiento y escalabilidad, y las restricciones de presupuesto y tiempo.