**ARQUITECTURAS DE SOFTWARE**



LEDY ASTUDILLO CALDERÓN

SANTIAGO NIETO

**Practica número 2 en el curso LABORATORIO DE INGENIERIA DE SW2**

**Profesor:**

**RICARDO ZAMBRANO SEGURA**

**Universidad del Cauca**

**Facultad de Ingeniería Electrónica y Telecomunicaciones**

**Departamento de Sistemas**

Laboratorio De Ingeniería De Software II

**Popayán, septiembre 2023**

LEDY ASTUDILLO CALDERÓN

SANTIAGO NIETO

**ARQUITECTURAS DE SOFTWARE**

Practica número 2 en el curso LABORATORIO DE INGENIERIA DE SW2

Estudiantes del:

Programa de Ingeniería de Sistemas

Profesor:

Ricardo Zambrano Segura

Popayán

2023

**CONTENIDO**

[¿Qué es una arquitectura de software? 3](#_Toc1063838535)

[¿Cuál es el papel de un arquitecto de software? 4](#_Toc25401341)

[ESTILO ARQUITECTÓNICO ORIENTADO A OBJETOS 5](#_Toc1235246744)

[Características 5](#_Toc1078983898)

[Ventajas 5](#_Toc1098203343)

[Desventajas 6](#_Toc209360487)

[Sistemas Recomendados 6](#_Toc1500846810)

[ESTILO ARQUITECTONICO MODELO VISTA CONTROLADOR (MVC) 6](#_Toc1023089790)

[Características 6](#_Toc1556964878)

[Ventajas 6](#_Toc2139043684)

[Desventajas 7](#_Toc398830668)

[Sistemas Recomendados 7](#_Toc510086446)

[TABLA COMPARATIVA 7](#_Toc956674128)

# ¿Qué es una arquitectura de software?

La arquitectura de software es un concepto que surge en los años 60 y se refiere a una planificación basada en modelos, patrones y abstracciones teóricas para desarrollar una pieza de software de cierta complejidad. También se le denomina arquitectura lógica y consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido y claro para interactuar con el código fuente del software.

En términos más simples, la arquitectura de software son patrones o lineamientos que ayudan a la construcción de un programa o aplicación. Esta planificación previa permite entender cómo van a encajar cada una de las piezas del producto o servicio, proporcionando una guía teórica detallada sobre el camino a seguir en el desarrollo.

La arquitectura de software define de manera abstracta los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Además, está compuesta por más arquitecturas de datos articuladas entre sí, lo que la convierte en un elemento complejo y fundamental en el desarrollo de software.

Algunas de las ventajas de utilizar una arquitectura de software adecuada son la reducción de costos, la mejora en la toma de decisiones, la efectividad de las rutas de acción y una proyección más acertada. Por lo tanto, es una fase de planificación crítica al comenzar cualquier proyecto de software, ya que proporciona una hoja de ruta clara sobre el camino a seguir en el desarrollo y brinda instrucciones sobre qué componentes utilizar y cómo deben interactuar entre sí.

# ¿Cuál es el papel de un arquitecto de software?

El papel de un arquitecto de software es crucial en el desarrollo de sistemas y aplicaciones de software. Algunas de sus responsabilidades clave incluyen:

* Diseñar y crear una arquitectura de software sólida y eficiente que cumpla con los requisitos técnicos y funcionales de un proyecto.
* Trabajar en estrecha colaboración con desarrolladores, ingenieros y otros miembros del equipo de desarrollo de software para definir la estructura del sistema, los componentes clave y las interacciones entre ellos.
* Tomar decisiones importantes sobre las tecnologías y herramientas que se utilizarán en el desarrollo del software.
* Colaborar con compañeros y usuarios finales de otras partes del negocio, lo que requiere habilidades de comunicación efectivas.
* Presentar conceptos e informar sobre resultados a miembros de la dirección, algunos de los cuales pueden carecer de conocimientos técnicos.
* Realizar tareas de seguimiento de la arquitectura después de la implementación del sistema y mejorar la arquitectura de forma continua.

3. **Seleccione dentro de la siguiente lista de estilos arquitectónicos dos para su estudio en profundidad:**

**Tuberías y filtros**

**Pizarra**

**Capas**

**Orientados a Objetos**

**REST**

**Máquina Virtual**

**Modelo Vista controlador**

**Analice sus características, principales ventajas, desventajas y sistemas en los cuales es recomendable su uso. Construya una tabla comparativa.**

# ESTILO ARQUITECTÓNICO ORIENTADO A OBJETOS

Características**:**

**Basado en el concepto de objetos:** Este estilo se centra en la creación de objetos que encapsulan tanto datos como comportamiento. Los objetos son instancias de clases que definen su estructura y comportamiento.

**Promueve la reutilización de código:** La reutilización de código es una de las piedras angulares del enfoque Orientado a Objetos. Se logra a través de mecanismos como la herencia, que permite a las clases heredar propiedades y métodos de otras clases, y la composición, que permite la construcción de objetos complejos a partir de objetos más simples.

**Diseño modular y abstracción de datos:** El diseño modular se logra al dividir el software en pequeñas unidades llamadas objetos, lo que facilita la comprensión y el mantenimiento del código. La abstracción de datos se refiere a ocultar los detalles internos de un objeto y proporcionar una interfaz clara para interactuar con él.

[¿Qué es una arquitectura de software? 3](#_Toc877612010)

[¿Cuál es el papel de un arquitecto de software? 4](#_Toc1570217435)

[ESTILO ARQUITECTÓNICO ORIENTADO A OBJETOS 5](#_Toc1386765520)

[Características 5](#_Toc1555557845)

[Ventajas 5](#_Toc1402648673)

[Desventajas 5](#_Toc1391591414)

[Sistemas Recomendados 6](#_Toc973008570)

[ESTILO ARQUITECTONICO MODELO VISTA CONTROLADOR (MVC) 6](#_Toc1438525731)

[Características 6](#_Toc628507057)

[Ventajas 6](#_Toc1754883512)

[Desventajas 7](#_Toc1031908158)

[Sistemas Recomendados 7](#_Toc853316768)

[TABLA COMPARATIVA 7](#_Toc2099134539)

Ventajas**:**

**Facilita la modularidad y extensibilidad:** La estructura modular de Orientado a Objetos permite a los desarrolladores dividir el sistema en partes más pequeñas y manejables. Esto facilita la adición de nuevas funcionalidades o la modificación de las existentes sin afectar otras partes del sistema.

**Reutilización de componentes y creación de bibliotecas:** La herencia y la composición permiten la creación de componentes reutilizables y bibliotecas de código que pueden ser utilizadas en múltiples proyectos.

**Diseño cercano al mundo real:** Al modelar el software en objetos que representan entidades del mundo real, como clientes, productos o transacciones, se facilita la comprensión del sistema tanto para los desarrolladores como para los usuarios finales.

Desventajas**:**

**Mayor complejidad de implementación:** El enfoque Orientado a Objetos puede ser más complejo de implementar en comparación con otros estilos arquitectónicos, especialmente para desarrolladores que están comenzando a trabajar con él.

**Rendimiento afectado por sobrecarga de objetos:** Si no se gestiona adecuadamente, la creación excesiva de objetos puede afectar el rendimiento del sistema debido a la sobrecarga asociada con la administración de objetos.

**Requiere planificación cuidadosa:** Para evitar la proliferación de clases y una estructura poco clara, se necesita una planificación cuidadosa y una comprensión profunda de cómo diseñar las clases y sus relaciones.

Sistemas Recomendados**:**

* **Aplicaciones de software que requieren una estructura modular y una alta reutilización de código:** Orientado a Objetos es especialmente efectivo en proyectos de mediana a gran escala, donde la estructura modular y la reutilización de componentes son críticas.
* **Desarrollo de aplicaciones de escritorio, aplicaciones web y sistemas embebidos:** Este estilo es versátil y se puede aplicar en una variedad de contextos, desde aplicaciones de escritorio hasta sistemas embebidos en dispositivos hardware. Su flexibilidad lo hace adecuado para una amplia gama de proyectos de software.

ESTILO ARQUITECTONICO MODELO VISTA CONTROLADOR (MVC):

Características**:**

**Divide la aplicación en tres componentes:** MVC divide una aplicación en tres partes interconectadas pero separadas: el Modelo, que maneja la lógica de negocios y los datos; la Vista, que se encarga de la presentación y la interfaz de usuario; y el Controlador, que actúa como intermediario entre el Modelo y la Vista.

**Responsabilidades claras:** Cada componente tiene responsabilidades bien definidas, lo que facilita la comprensión y el mantenimiento del código.

Ventajas**:**

**Separación clara de responsabilidades:** La separación de responsabilidades entre el Modelo, la Vista y el Controlador hace que sea más fácil mantener, depurar y escalar la aplicación. Los cambios en un componente no deberían afectar a los otros dos.

**Facilita la colaboración en equipos de desarrollo:** El enfoque MVC permite que diferentes desarrolladores trabajen en cada uno de los componentes de manera independiente, lo que agiliza el desarrollo en equipos grandes.

**Permite cambios en la interfaz de usuario sin afectar la lógica subyacente:** Puesto que la Vista y el Modelo están desacoplados a través del Controlador, es posible realizar modificaciones en la interfaz de usuario sin necesidad de cambiar la lógica de negocio, lo que agiliza las actualizaciones y mejoras.

Desventajas**:**

**Aumento de la complejidad en aplicaciones pequeñas:** En aplicaciones pequeñas o simples, la introducción de la estructura MVC puede parecer innecesaria y aumentar la complejidad del código.

**Requiere planificación adecuada para definir interfaces:** Para que el MVC funcione correctamente, es esencial definir interfaces claras entre los componentes. Si no se planifica correctamente, puede llevar a confusión y errores en el desarrollo.

**No es la mejor opción para aplicaciones en tiempo real o altamente concurrentes:** MVC no es el estilo arquitectónico más adecuado para aplicaciones en tiempo real que requieren una alta concurrencia, ya que puede introducir latencia debido a la necesidad de coordinación entre el Controlador y la Vista.

Sistemas Recomendados**:**

* **Aplicaciones web y de escritorio que requieren una separación clara entre la lógica de negocio y la interfaz de usuario:** MVC es especialmente útil en aplicaciones donde se necesita una separación clara de responsabilidades para facilitar el mantenimiento y las actualizaciones.
* **Proyectos donde múltiples desarrolladores trabajan en diferentes partes de la aplicación:** MVC es ideal para proyectos colaborativos en los que diferentes equipos o desarrolladores se enfocan en el Modelo, la Vista o el Controlador, ya que permite una división efectiva del trabajo.

# TABLA COMPARATIVA

|  |  |  |
| --- | --- | --- |
| **Aspecto** | **Orientados a Objetos** | **Modelo Vista Controlador (MVC)** |
| ***Características principales*** | Basado en objetos, reutilización de código, modularidad. | Separación de responsabilidades, estructura en tres componentes. |
| ***Ventajas*** | Reutilización de código, diseño modular, cercano al mundo real. | Mantenimiento más fácil, colaboración en equipo, cambios en la interfaz sin afectar la lógica. |
| ***Desventajas*** | Mayor complejidad, sobrecarga de objetos, planificación cuidadosa. | Mayor complejidad en aplicaciones pequeñas, definición de interfaces, no ideal para tiempo real. |
| ***Sistemas Recomendados*** | Aplicaciones diversas con alta reutilización de código. | Aplicaciones web y de escritorio con separación clara de responsabilidades. |

4. Recientemente ha habido muchas inconformidades con el censo en Colombia que se realizó[[1]](#endnote-3158) en 2018, el gobierno ha decidido estudiar la posibilidad de realizar uno nuevo para el año 2022 y así corregir los errores del anterior para tener unas cifras más reales, las cuales además han cambiado dado el fenómeno migratorio venezolano. Su empresa ha ganado la licitación para este nuevo proyecto, y una de sus primeras decisiones será la de seleccionar una arquitectura adecuada para el nuevo sistema.

**¿Considera que alguna de las arquitecturas estudiadas sería útil para esta tarea?, ¿por qué sí?, ¿por qué no?**

**Para ello revise bien que es un censo, ¿cómo se realiza?, ¿qué tipo de información se colecta?, ¿qué información debe arrojar?, ¿qué componentes debe tener un sistema que lo soporte, igualmente revise que atributos de calidad debe tener.**

**Recuerde somos más de 50 millones de colombianos hoy (50.270.571).**

En el contexto de seleccionar una arquitectura adecuada para el nuevo sistema del censo en Colombia, tanto las arquitecturas orientadas a objetos como la modelo vista controlador podrían ser útiles para esta tarea.

Las arquitecturas orientadas a objetos permiten organizar el sistema en componentes reutilizables, lo que facilita la creación y mantenimiento del software. Esto sería beneficioso para el desarrollo de un sistema de censo, ya que se podrían diseñar clases y objetos que representen adecuadamente los conceptos y entidades relacionadas con la recolección y análisis de datos demográficos

Por otro lado, la modelo vista controlador es un patrón de diseño que separa la lógica de la aplicación en tres componentes: el modelo (que representa los datos y la lógica de negocio), la vista (que muestra la información al usuario) y el controlador (que maneja las interacciones del usuario). Esta separación permite un mayor modularidad y facilita el mantenimiento y la evolución del sistema

En cuanto a los atributos de calidad que se deben tener en cuenta, un sistema de censo debe ser confiable y preciso en la recolección y procesamiento de datos demográficos, ya que la exactitud de los resultados es fundamental. También debe ser seguro para garantizar la protección de la información sensible de los ciudadanos. Además, la escalabilidad y el rendimiento son importantes para manejar grandes volúmenes de datos y asegurar un procesamiento eficiente.

En resumen, tanto las arquitecturas orientadas a objetos como la modelo vista controlador podrían ser útiles para el nuevo sistema de censo en Colombia. Ambas ofrecen ventajas en términos de organización, modularidad y mantenibilidad del software. Sin embargo, la elección final dependerá de las necesidades y requisitos específicos del proyecto, así como de la experiencia y competencias del equipo de desarrollo.

1. [↑](#endnote-ref-3158)