UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE COMPUTACION

MATERIA: DESARROLLO DE SOFTWARE PARA MOVILES

DOCENTE: MARIO ALVARADO

CICLO O1-2022

TRABAJO DE INVESTIGACION 2

INTEGRANTES Y N.º DE CARNÉ:

EDUARDO ANTONIO BARILLAS NERIO BN181161

ALAN WILFREDO CANAS UMANA AC180566

ÍNDICE

Contenido **Página**

Introducción

La presente investigación consta en realizar una comparación entre dos patrones de arquitectura, los cuales son MVP y MVC. Se compararán ventajas y desventajas de cada uno, así como una breve explicación de su funcionamiento y aplicación y comentarios sobre los patrones estudiados.

Buscamos sacar conclusiones en la comparación entre los dos patrones para identificar cual es mas recomendable

Se realizará una sencilla aplicación que ponga en práctica uno de los patrones para aplicar los conceptos aprendidos y la elaboración de un video explicativo sobre la aplicación

Objetivo General

Aplicar uno de los patrones investigados en el desarrollo de una aplicación sencilla

Objetivos Específicos

Identificar en la comparativa que patrón es más conveniente

Despejar dudas sobre el uso, funcionamiento y como aplicar los patrones MVC y MVP

MVC

MVC es una propuesta de arquitectura de software que se utiliza para separar códigos por sus responsabilidades, generando capas que realizan tareas concretas, esto obtiene muchos beneficios

Se utiliza principalmente en sistemas que requieren uso de interfaces de usuario. Esta herramienta surge de la necesidad de crear un software mas robusto con un ciclo de vida adecuado para así potenciar el mantenimiento de una forma mas sencilla, reutilizando el código y separar conceptos

Su fundamento es la separación del código en 3 capas llamadas Modelos, Vistas y Controladores

Modelo: Es donde se trabaja con los datos. Es una representación. Los datos estarán habitualmente en una base de datos

Vista: es la interfaz de usuario. el código que nos permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas están los códigos HTML y PHP que permite mostrar la salida.

Controlador: es la conexión entre modelo y vista. Contiene el código para responder a las acciones que se necesitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

Ventajas

El desarrollo de los distintos componentes se puede realizar de manera simultánea entre varios desarrolladores.

Funciona muy bien para aplicaciones web.

El soporte es más sencillo, orientado a un nuevo tipo de clientes.

Desventajas

La curva de aprendizaje para nuevos desarrolladores es un poco superior a los otros modelos que son más simples.

Tener varias capas nos incrementa la complejidad del sistema.

Comentarios

MVC tiene un buen desempeño al separar modelo y vista. Es sencillo escribir pruebas con relación al modelo, ya que no está atado a nada, y no hay mucho que testear en la vista.

En el controlador si hay algunos problemas.

MVP

MVP es un patrón de interfaz de usuario diseñada para facilitar pruebas de unidad y mejorar la separación de inquietudes en lógica de presentación.

En una aplicación Java, el patrón MVP puede ser utilizado dejando la clase de interfaz del usuario implementar una interfaz de vista, de igual forma puede aproximarse para ser utilizada para aplicaciones basadas en web Java.

El entorno .NET apoya el patrón MVP, la misma clase modelo y presentador pueden usar soporte interfaces múltiples, como un Aplicación web ASP.NET, una aplicación de Windows Form o una aplicación Silverlight.

Modelo: es una interfaz que define los datos que se mostrarán o sobre los que actuará la interfaz de usuario.

Presentador: actúa sobre el modelo y la vista. Recupera datos de los repositorios y los formatea para mostrarlos en la vista.

Vista: es una interfaz pasiva que exhibe datos y órdenes de usuario de las rutas al presentador para actuar sobre los datos.

Ventajas

Reducir el acoplamiento

División clara de responsabilidades

Propicio para el desarrollo basado en pruebas

Desventajas

La representación de la vista se coloca en el Presentador, por lo que la interacción entre la vista y el Presentador será demasiado frecuente.

Si el presentador representa demasiado la vista, tiende a hacerla muy relacionada con una vista específica.

Comentarios

El resultado es mucho más limpio.

Se puede testear de forma más sencilla la lógica del presenter porque no está vinculada a ninguna vista ni API específicas de Android.

Los Presenters, al igual que los Controllers, son propensos a recopilar lógica de negocio con el tiempo. En proyectos de larga duración, los desarrolladores se encuentran con grandes Presenters, difíciles de separar.

Comparación MVC vs MVP

MVP realiza un mejor trabajo que MVC al dividir la aplicación en componentes modulares, con propósitos mejor definidos.

La implementación de MVP será diferente dependiendo de la implementación de View. Una parte tiende a colocar lógica simple en View y lógica compleja en Presenter, la otra parte tiende a colocar toda la lógica en Presenter. Estos dos se denominan: vista pasiva y controlador de supervisión.

El formulario web ASP.NET y la tecnología de desarrollo basada en eventos basados en WinForms con la que están familiarizados los programadores .NET es el modelo MVP utilizado.

CONCLUSIONES

Esta investigación amplia nuestros conocimientos sobre los patrones estudiados, pudimos identificar las ventajas y desventajas de cada uno al igual que su funcionamiento mediante la investigación de sus conceptos. En el desarrollo de la aplicación pudimos poner en practica lo aprendido

Enlace de repositorio

https://github.com/Alan-AC7/TrabajoInvestigacion2

Enlace de video explicativo

https://drive.google.com/file/d/11-FQzPZMQO2ktx_-kiKuvf1ca949ANGa/view?usp=sharing

NOMBRE DEL ESTUDIANTE	**USUARIO DE GITHUB**
Alan Wilfredo Cañas Umaña	**Alan-AC7**
Eduardo Antonio Barillas Nerio	**EduardoNerio**