Tecnológico Nacional de México campus Colima

Maestría en Sistemas Computacionales

Tecnologías de programación

Actividad - "Investigar e implementar en Python el patrón de diseño arquitectural MVC"

D. en C. Patricia Elizabeth Figueroa Millán

Dante Alberto Márquez Almaraz | G2146002

Villa de Álvarez, Colima - 30 de noviembre de 2022

Indice:

MATERIALES

DESARROLLO

- MVC
- MODELO
- VISTA
- CONTROLADOR

RESULTADOS

- EJEMPLO

CONCLUSIÓN

REFERENCIAS

Resumen:

En el presente documento es una libreta de Python, la cual contiene un reporte de investigación referente a los patrones de diseño, los cuales son técnicas para resolver problemas comunes que ocurren en el desarrollo de software. En este documento se plasma una investigación referente al MVC (Modelo Vista Controlador), siendo este un estilo de arquitectura de software enfocado en separar los datos de una aplicación, la interface de usuario y la lógica de control. Este patrón es empleado en aplicaciones web modernas son muy complejas, y hacer un cambio a veces puede ser un gran dolor de cabeza. Administrar el frontend y el backend en componentes separados más pequeños permite que la aplicación sea escalable, mantenible y fácil de expandir.

Introducción:

Los patrones de diseño o design patterns, son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error [1].

Objetivo:

Los alumnos de manera individual deberán investigar qué es el patrón de diseño arquitectural MVC y cómo funciona, así como su implementación en python. Para esto el alumno deberá:

Documentar la teoría necesaria para evidenciar qué es y cómo funciona el patrón de diseño.

Implementar un código de ejemplo sobre el patrón de diseño MVC en python.

El código debe estar documentado (no copy-paste); contar con las clases para modelo, vista, controlador (ej. Class Model, Class View, Class Controler y función main)

Para el modelo pueden emplear una estructura de datos (lista, diccionarios, etc.) o csv.

Debe entregarlo en una libreta de Jupyter, considerando la rúbrica para organizadores gráficos.

Recuerde desarrollar su reporte considerando los detalles finos y de calidad.

Metodología:

Se realizó una investigación referente a los temas de patrones de deseño, en específico MVC.

Materiales:

Computadora.

Internet.

Visual Studio Code.

Jupyter Notebook.

Python.

Desarrollo:

MVC

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo [2].

MVC se usa inicialmente en sistemas donde se requiere el uso de interfaces de usuario, aunque en la práctica el mismo patrón de arquitectura se puede utilizar para distintos tipos de aplicaciones. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos [3].

A continuación se muestran las partes de un MVC [4].

MODELO

Se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. todo eso va aquí, en el modelo.

VISTA

Son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va aquí. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.

CONTROLADOR

Se encarga de controlar, recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.

Resultados:

Ejemplo

Se necesita desarrollar una aplicación que informe a los usuarios sobre los servicios de marketing proporcionados por la empresa en la nube, incluidos correo electrónico, SMS y servicios de voz [5].

In [ ]:
```
# Primero desarrollamos la clase Model (modelo) para definir los servicios y costos pr
class Model(object):
    services = {
        'email': {'number': 1000, 'price': 2},
        'sms': {'number': 1000, 'price': 10},
        'voice': {'number': 1000, 'price': 15},
    }

# Luego se define la clase de View (vista), que proporciona un método para retroalimen
# define dos métodos list_services () y list_pricing () para mostrar los servicios y p
class View(object):
    def list_services(self, services):
        for svc in services:
            print(svc, " ")

    def list_pricing(self, services):
        for svc in services:
            print("For", Model.services[svc]['number'], svc, "message you pay $", Mode

# Se define la clase Controller (controlador), que define dos métodos get_services ()
# Ambos métodos se utilizan para consultar el modelo y obtener datos, y luego enviar l
class Controller(object):
    def __init__(self):
        self.model = Model()
        self.view = View()

    def get_services(self):
        services = self.model.services.keys()
        return self.view.list_services(services)

    def get_pricing(self):
        services = self.model.services.keys()
        return self.view.list_pricing(services)

# La clase Client creará una instancia del controlador, y luego el objeto controlador
class Client(object):
    con = Controller()
```

```
    print("Services Provided:")
    con.get_services()
    print("Pricing for Services:")
    con.get_pricing()
```

```
Services Provided:
email
sms
voice
Pricing for Services:
For 1000 email message you pay $ 2
For 1000 sms message you pay $ 10
For 1000 voice message you pay $ 15
```

Conclusión:

El gran crecimiento del sector de las tecnologías de la información ha hecho que las prácticas de desarrollo de software evolucionen. Antes se requería completar todo el software antes de realizar pruebas, lo que suponía encontrarse con problemas. Para ahorrar tiempo y evitar volver a la etapa de desarrollo una vez que este ha finalizado, se introdujo una práctica de prueba durante la fase de desarrollo. Esta práctica se usa para identificar condiciones de error y problemas en el código que pueden no ser evidentes en ese momento. En definitiva, los patrones de diseño te ayudan a estar seguro de la validez de tu código, ya que son soluciones que funcionan y han sido probados por muchísimos desarrolladores siendo menos propensos a errores.

En este caso el uso de MVC es la separación de preocupaciones. Las aplicaciones web modernas son muy complejas, y hacer un cambio a veces puede ser un gran dolor de cabeza. Administrar el frontend y el backend en componentes separados más pequeños permite que la aplicación sea escalable, mantenible y fácil de expandir.

Referencias:

[1] MartÍnez, M. 2022. Profile. ¿Qué son los patrones de diseño de software?. https://profile.es/blog/patrones-de-diseno-de-software/

[2] Universidad de Alicante. 2022. Modelo vista controlador (MVC). https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

[3] Alvarez, M. 2020. Qué es MVC. https://desarrolloweb.com/articulos/que-es-mvc.html

[4] Hernández, U. 2015 .MVC (Model, View, Controller) explicado. https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

[5] Programador clic. 2020. Modo Python modelo-vista-controlador-compuesto. https://programmerclick.com/article/32121550755/