

Gastón Fuentes

gastonandres.fuentes@gmail.com

**Informe patrón de diseño MVC en Tkinter Python**

INDICE

[CREACION DE MANTENEDOR PRODUCTO CON TKINTER MVC 2](#_Toc54636982)

[Descripción del requerimiento 2](#_Toc54636983)

[Entregable 2](#_Toc54636984)

[Pantalla principal del mantenedor del producto 3](#_Toc54636985)

[Código ventana producto 4](#_Toc54636986)

[Agregar producto 5](#_Toc54636987)

[Código Controlador producto 6](#_Toc54636988)

[Método que comunica el Controlador con el Modelo y agrega un producto 7](#_Toc54636989)

[Método del Modelo que inserta un producto en la BD 7](#_Toc54636990)

[Eliminar producto 8](#_Toc54636991)

[Método que comunica el controlador con el Modelo y elimina un producto 9](#_Toc54636992)

[Método del Modelo que elimina un producto de la BD 9](#_Toc54636993)

[Vista editar producto 10](#_Toc54636994)

[Código ventana modificar 11](#_Toc54636995)

[Método que abre la Vista editar desde el Controlador producto 12](#_Toc54636996)

[Método que envía la información de la Vista al Controlador 13](#_Toc54636997)

[Método que comunica el Controlador con el Modelo y modifica un producto 13](#_Toc54636998)

[Método del Modelo que modifica un producto en la BD 13](#_Toc54636999)

[Error que tiene la aplicación 14](#_Toc54637000)

[Hipótesis de por qué se da este error 18](#_Toc54637001)

[Descripción de las acciones realizadas 19](#_Toc54637002)

[Conclusiones 19](#_Toc54637003)

# CREACION DE MANTENEDOR PRODUCTO CON TKINTER MVC

|  |  |
| --- | --- |
| **ALUMNO:** | Gastón Andrés Fuentes Mazuela |
| **FECHA:** | 19/10/2020 |
| **ACTIVIDAD:** | **Aplicar metodologías de programación (Estructurada, POO) optimizadas para la construcción de software**   * Utiliza lenguaje orientado a objeto y/o estructurado para la resolución de problemas considerando las reglas de negocio del cliente.   **Utilizar metodologías de base de datos relacionales y orientado a objetos que resuelven la persistencia en aplicaciones de clientes** |

## **Descripción del requerimiento**

Haciendo un curso de Python en Udemy, aprendiendo Tkinter para hacer aplicaciones de escritorio quise implementar el patrón de diseño MVC. Busqué información por internet y no encontré. Por lo que intente hacerlo con mis conocimientos, pero tuve ciertas complicaciones con el controlador, ya que, al instanciar las clases de las ventanas en la vista, se generaban ventanas o se desordenaban, por lo que adapté el patrón MVC a Tkinter. No sería 100% MVC, pero si quedó como un patrón modular bastante ordenado. Se podría considerar más como N capas.

El mantenedor de producto en Tkinter utilizando el patrón de diseño MVC, consta de un CRUD de inserción, modificación, lectura y eliminación de datos. Es bastante simple. Solo es una tabla de producto, con dos datos; nombre, precio.

## **E****ntregable**

El mantenedor del producto permite registrar, modificar, ver y eliminar los datos de un producto. Esta desarrollado en Python, utilizando Tkinter para la aplicación de escritorio y con base de datos MySQL.

## Pantalla principal del mantenedor del producto

Imagen que contiene Tabla

Descripción generada automáticamente

## Código ventana producto

Texto

Descripción generada automáticamente

Como verán la vista producto la hice como una clase en la capa de la Vista. Cada elemento como la ventana, el contenedor, los input, los tengo como variables en la clase, cosa de poder manipularlos en el Controlador.

## Agregar producto

Interfaz de usuario gráfica

Descripción generada automáticamente

## Código Controlador producto

Texto

Descripción generada automáticamente

El Controlador del producto es como una especie de hibrido entre un controlador y una vista. Esto es debido a que para poder acceder a la vista del producto debo instanciar la clase y para esto necesito crear una ventana Tk() que es lo que recibe la ventana del producto por parámetro. Para instanciar tanto la ventana, como el modelo, cree variables globales. Al mismo tiempo podemos ver en el constructor del Controlador que creo botones como si fuera una vista, esto es porque es una forma más sencilla de comunicar el botón a la función del controlador a través del command, aunque también pude haber hecho los botones en la vista y utilizar funciones en la vista solo para comunicar la vista y el controlador. Pero sentí que esta solución se adaptaba más a Tkinter.

## Método que comunica el Controlador con el Modelo y agrega un producto

Texto

Descripción generada automáticamente

Aquí lo primero que hacemos es usar el método validar el cual está dentro del Controlador. Este devuelve True o False, y es para ver que los datos no vengan vacíos. Luego hacemos variables que contengan los datos de la vista y una variable fila la cual será igual al método registrarProducto del Modelo, este método del Modelo además de registrar un producto en la BD nos devolverá un 1 o un 0 dependiendo si se ejecuta correctamente. Además, utilizamos la variable mensaje el cual es un label vacío que está en la Vista y le damos un valor dependiendo el caso. Luego para finalizar utilizamos el método obtenerDatos de la Vista para refrescar el árbol (grilla) con el nuevo producto. Esto lo considero un tanto indebido ya que la vista no debería tener métodos de este estilo, pero siento que se adecua más a Tkinter desde mi punto de vista.

## Método del Modelo que inserta un producto en la BD

Texto

Descripción generada automáticamente

## Eliminar producto

Interfaz de usuario gráfica

Descripción generada automáticamente

## Método que comunica el controlador con el Modelo y elimina un producto

Texto

Descripción generada automáticamente

Este método es muy similar al de agregar producto, solo que primero recibimos el nombre del producto que seleccione el usuario en el árbol (grilla) de la Vista. Luego lo demás funciona exactamente igual que en el anterior método.

## Método del Modelo que elimina un producto de la BD

Texto

Descripción generada automáticamente

## Vista editar producto

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

## Código ventana modificar

Texto

Descripción generada automáticamente

Aquí tenemos otra Vista, la de editar un producto. Al igual que la anterior de producto, esta es una clase para así poder manejar todas las variables en el controlador. Aquí quise utilizar una forma distinta de usar el patrón MVC por decirlo de una forma. Debido a que creé una variable global del Controlador editar. Esto para comunicar la Vista y el Controlador. También esta Vista recibe como parámetro además de la ventana, un nombre y precio, estos vienen del Controlador del producto y se pueden visualizar en los input a través de textvariable el cual le asignamos un StringVar y pasamos por parámetro el contenedor de la vista (en donde se posicionara) y el nombre que selecciona el usuario en el arbol (grilla).

## Método que abre la Vista editar desde el Controlador producto

Texto

Descripción generada automáticamente

Este método del Controlador producto, recibe el nombre y el precio que selecciona el usuario del árbol (grilla) luego se instancia la Vista editar del producto y se abre como una ventana aparte.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Como podemos apreciar en la imagen, podemos editar el producto y aparecerá un mensaje en la vista principal del producto, así como también se actualizarán los datos en el árbol (grilla). Esta captura es posterior a haber actualizado el producto. Lo menciono porque luego de apretar el botón editar producto, la ventana modificar producto, se destruye.

## Método que envía la información de la Vista al Controlador

Texto

Descripción generada automáticamente

Esto es distinto a como traté la Vista en la pantalla principal. A diferencia de esa, aquí la Vista tiene un método que envía la información al controlador, al mismo tiempo el botón esta dentro de la misma Vista y no en el controlador, como lo hacía en la pantalla principal.

## Método que comunica el Controlador con el Modelo y modifica un producto

Texto

Descripción generada automáticamente

## Método del Modelo que modifica un producto en la BD

Texto

Descripción generada automáticamente

## Error que tiene la aplicación

Interfaz de usuario gráfica

Descripción generada automáticamente

Hasta aquí no hay ningún error. Se selecciona el producto y la ventana de modificar lo muestra.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Aquí está listo para ser modificado el producto en la ventana modificar producto.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Aquí podemos ver que el producto ha sido actualizado correctamente y la ventana modificar producto fue destruida, como debía pasar. Pero en la siguiente imagen veremos cuál es el error.

Interfaz de usuario gráfica

Descripción generada automáticamente

Como podemos apreciar en la imagen a pesar de que selecciono el producto laptop gama baja, el producto que aparece en la ventana modificar producto es play 5 con el precio que tenía antes de ser modificado. No obstante, de que en el árbol (grilla) podemos verlo modificado en su precio.

## Hipótesis de por qué se da este error

Texto

Descripción generada automáticamente

Si la variable fila es igual a 1 creo una variable ventana la cual se le asigna Tk() y luego instancio la clase VentanaProducto la cual es la Vista pantalla principal, esto lo hago con el fin de poder utilizar la variable mensaje de esta clase y mostrar producto actualizado correctamente en la pantalla principal, así como utilizar el método obtenerDatos de esta misma clase y actualizar el árbol (grilla). Así mismo la razón por la cual destruyo esta ventana, es porque si no lo hago se abre una segunda pantalla principal, pero solo con los inputs.

Mi hipótesis es que al instanciar la VentanaProducto se queda con los item seleccionados del árbol (grilla) cuando se editó un producto anteriormente. De hecho, es mas, probé solo ejecutar el método editarProducto del modelo, borrando todo el resto del código y funciona perfectamente sin este error, pero al mismo tiempo no se muestra el mensaje en la pantalla principal que el producto fue actualizado, ni tampoco se actualiza el árbol (grilla).

## 

## **Descripción de las acciones realizadas**

*Para desarrollar el mantenedor del producto se utilizó el lenguaje Python, con Tkinter para diseñar la vista grafica de la aplicación de escritorio y para la base de datos se utilizó MySQL.*

El mantenedor debe registrar, ver, actualizar y eliminar datos de un producto. En la pantalla principal se pueden ver lo siguiente:

* Formulario de registro: en la parte superior de la pantalla principal podemos ver un formulario de registro con un nombre y un precio. Así como 3 botones, insertar producto, editar producto y eliminar producto. En la parte inferior podemos ver una grilla con los productos de la BD
* Insertar producto: al apretar este botón, si verifican que los datos no vengan vacíos y luego se insertan en la tabla producto de la BD. En la pantalla principal podremos apreciar un mensaje en rojo que dirá el nombre del producto insertado correctamente. Así como la actualización de la grilla en tiempo real.
* Editar producto: aquí seleccionamos un producto de la grilla y luego apretamos el botón editar producto, esto nos abrirá una ventana anexa, la cual tendrá los datos del nombre y precio del producto seleccionado, permitirá que el usuario sobre escriba la información, luego se presiona el botón editar producto y esta ventana se cerrara, luego aparecerá un mensaje en rojo de producto actualizado correctamente y se actualizara la grilla en tiempo real, así como el producto en la BD
* Eliminar producto: aquí al igual que en editar, seleccionaremos un producto de la grilla y al apretar el botón eliminar producto aparecerá un mensaje en rojo con el nombre del producto eliminado correctamente, este producto desaparecerá tanto de la grilla, como de la BD.

## **Conclusiones**

*En mi opinión si bien, no considero que esta aplicación se haya desarrollado con un patrón MVC estrictamente según la teoría. Debido a que por ejemplo en la Vista de la pantalla principal podemos ver un método obtenerDatos, el cual necesita información del Modelo para visualizar los datos en la grilla. También el Controlador del producto es una especie de hibrido entre una Vista y un Controlador, puesto que para manipular los datos de la Vista se debe instanciar la clase de la Vista y una ventana Tk, además en el Controlador de la pantalla principal tiene botones, cosa de que el command pueda comunicarse con los métodos en la misma clase.*

*Consideraría que esta es una forma de adaptar el patrón MVC a como funciona Tkinter, siendo mas un orden modular o quizás un patrón de diseño N capas que simula un MVC.*