**CLASIFICACION DE REQUISITOS – ACTIVIDAD**

**Danilo Andrés Montezuma Ibarra**

**Tomas Benavides Calderón**

**Ingeniero, Miguel Tovar Bastidas**

**Documentación del Código de Plataforma de Venta de Aplicaciones**

Este código implementa una plataforma de venta de aplicaciones en línea llamada "Fandroid". Permite a los clientes crear cuentas, comprar aplicaciones, acumular y canjear puntos, y realizar diversas acciones relacionadas con sus cuentas y compras.

**Requerimientos Funcionales:**

1. **Crear una cuenta de usuario:** Los usuarios pueden crear cuentas proporcionando su nombre, información de tarjeta de crédito y otros detalles requeridos.
2. **Iniciar sesión:** Los usuarios pueden iniciar sesión con sus cuentas existentes para acceder a la plataforma.
3. **Comprar aplicaciones:** Los usuarios pueden explorar un catálogo de aplicaciones disponibles y comprarlas utilizando su saldo.
4. **Acumular puntos:** Los usuarios acumulan puntos por cada compra de aplicación que realizan con su tarjeta de crédito/debito.
5. **Consultar puntos acumulados:** Los usuarios pueden verificar la cantidad de puntos que han acumulado por sus compras.
6. **Canjear puntos por bonos:** Los usuarios pueden canjear sus puntos acumulados por bonos con diferentes valores.
7. **Cerrar sesión:** Los usuarios pueden cerrar sesión en sus cuentas para salir de la plataforma.

**Requerimientos No Funcionales:**

1. **Límites de Nombre:** El nombre del usuario no debe exceder los 30 caracteres. Si se ingresa un nombre más largo, se debe mostrar un mensaje de error.
2. **Información de Tarjeta Válida:** La información de la tarjeta de crédito debe ser válida y pertenecer a una de las marcas admitidas (Visa, Mastercard, American Express, Diners Club).
3. **Límites de Longitud de Tarjeta:** El número de tarjeta debe tener entre 13 y 18 dígitos.
4. **Vencimiento de Tarjeta Válido:** La fecha de vencimiento de la tarjeta debe estar entre los años 2024 y 2029.
5. **CVV Válido:** El CVV debe constar de tres dígitos.
6. **Saldo Inicial para los Primeros 10:** Los primeros 10 usuarios que se registren recibirán un saldo inicial de $50000 en su cuenta.
7. **Prevención de Compras Repetidas:** Los usuarios no pueden comprar la misma aplicación dos veces.
8. **Requisitos de Contraseña:** La contraseña debe tener al menos 6 caracteres, incluyendo al menos una letra y un número.

El código se ha estructurado en varias clases y funciones para cumplir con los requisitos funcionales y no funcionales establecidos. La función principal **main()** actúa como el punto de entrada y guía a los usuarios a través de las opciones disponibles, como crear una cuenta, iniciar sesión y acceder a funciones relacionadas con la cuenta y las compras. El código utiliza un archivo JSON llamado "customers.json" para guardar y cargar los datos de los clientes, incluidos sus nombres, información de tarjetas, puntos, saldo, aplicaciones compradas y contraseña

**Estructura del Código:**

1. **Clase Saldo:** Representa el saldo de una cuenta. Puede tener un valor inicial y se utiliza para realizar transacciones.
2. **Clase Application:** Representa una aplicación disponible para comprar. Tiene un nombre y un precio.
3. **Clase Customer:** Representa un cliente con su nombre, información de tarjeta, puntos, saldo, aplicaciones compradas y contraseña
4. **Funcion Display\_info:** Esta cada que es llamada muestra la información de la cuenta siendo nombre, información de tarjeta, puntos acumulados, saldo, aplicaciones compradas.
5. **Funcion Save\_Costumers:** se encarga de guardar la información de los clientes en un archivo JSON llamado "customers.json". Esta función toma como entrada una lista de objetos de la clase **Customer**, que representan a los usuarios de la plataforma de venta de aplicaciones.
6. **Funcion Load\_Customers:** La función **load\_customers()** se encarga de cargar la información de los clientes desde un archivo JSON y convertirla de nuevo en objetos de la clase **Customer**. Esta función es esencial para mantener la persistencia de los datos de los clientes entre diferentes sesiones de ejecución del programa.
7. **Función Main:** Contiene la lógica principal del programa, incluyendo la interacción con el usuario y el manejo de las diferentes opciones del menú.
8. **Función buy\_application:** Permite a los usuarios comprar aplicaciones, deduciendo el costo de la aplicación de su saldo y acumulando puntos.

**Uso del Código:**

El código comienza con un menú principal que permite a los usuarios crear cuentas, iniciar sesión y salir de la plataforma. Después de iniciar sesión, los usuarios pueden realizar diversas acciones, como comprar aplicaciones, consultar puntos acumulados y canjear puntos por bonos. Los primeros 10 usuarios que se registren recibirán un saldo inicial de $50000. El código utiliza clases para representar el saldo, las aplicaciones y los clientes, facilitando la gestión de los datos y las operaciones.

Este código proporciona una base para la implementación de una plataforma de venta de aplicaciones en línea. Los requerimientos funcionales y no funcionales están diseñados para garantizar la seguridad, usabilidad y funcionalidad del sistema.

**Clase Saldo:**

**class Saldo:**

**def \_\_init\_\_(self, saldo\_inicial=0):**

**self.valor = saldo\_inicial**

La clase Saldo representa el saldo de una cuenta. Tiene un atributo valor que indica el saldo actual de la cuenta. La clase tiene un constructor \_\_init\_\_ que permite establecer un saldo inicial opcional. Si no se proporciona un saldo inicial, el saldo se establece en 0.

**Clase Application:**

**class Application:**

**def \_\_init\_\_(self, name, price):**

**self.name = name**

**self.price = Price**

La clase Application representa una aplicación disponible para comprar en la plataforma. Tiene dos atributos: name, que almacena el nombre de la aplicación, y price, que almacena el precio de la aplicación.

**Clase Customer:**

**class Customer:**

**def \_\_init\_\_(self, name, card\_info, points, saldo):**

**self.name = name**

**self.card\_info = card\_info**

**self.points = points**

**self.saldo = saldo**

**self.password = password**

**self.apps\_purchased = apps\_purhased**

La clase Customer representa a un cliente de la plataforma. Tiene varios atributos:

* name: Almacena el nombre del cliente.
* card\_info: Almacena la información de la tarjeta de crédito del cliente.
* points: Almacena la cantidad de puntos acumulados por el cliente.
* saldo: Almacena un objeto de la clase Saldo que representa el saldo de la cuenta del cliente.
* apps\_purchased: Almacena una lista de nombres de aplicaciones que el cliente ha comprado.
* Password: almacena la contraseña que el cliente a ingresado

**Función main:**

La función main es el punto de entrada del programa y contiene la lógica principal del flujo de la aplicación. Permite a los usuarios crear cuentas, iniciar sesión y realizar diversas acciones después de iniciar sesión.

**Función buy\_application:**

Esta función permite a los usuarios comprar aplicaciones. Muestra un catálogo de aplicaciones disponibles, permite al usuario elegir una aplicación para comprar y actualiza el saldo del cliente y los puntos acumulados en función de la compra.

**Uso de Clases y Datos:**

El código utiliza las clases Saldo, Application y Customer para representar los conceptos relacionados con la plataforma de venta de aplicaciones. Los datos se organizan en objetos de estas clases para facilitar la gestión y manipulación de la información.

**Flujo General:**

1. El programa comienza en el menú principal, donde los usuarios pueden crear cuentas, iniciar sesión o salir.
2. Cuando los usuarios crean cuentas, se les asigna un saldo inicial de $50000 si son uno de los primeros 10 en registrarse.
3. Después de iniciar sesión, los usuarios pueden comprar aplicaciones, consultar puntos, canjear puntos por bonos u otras acciones.
4. Los datos se almacenan en objetos de las clases Customer, Saldo y Application.

**Beneficios de la Estructura:**

La estructura orientada a objetos facilita la comprensión y el mantenimiento del código. Los requerimientos funcionales y no funcionales se implementan de manera modular, lo que permite un mejor control y manejo de la lógica de la plataforma de venta de aplicaciones.