

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

INFORME DE PROYECTO FINAL

TÍTULO: ABC

**Autores:**

APELLIDOS, Nombres *en orden alfabético*

**Curso:**

Análisis de Algoritmos y Estrategias de Programación

**Docente del Curso:**

APELLIDOS, Nombres

Cajamarca – Perú

2024-1

**Contenido**

[**RESUMEN.** 3](#_Toc144990320)

[**I.** **INTRODUCCIÓN.** 3](#_Toc144990321)

[**1.1.** **Definición de objetivos** 3](#_Toc144990322)

[**1.2.** **Descripción del problema** 3](#_Toc144990324)

[**II.** **ANÁLISIS DE LA SOLUCIÓN** 3](#_Toc144990325)

[**2.1.** **Elección de la solución** 3](#_Toc144990326)

[**2.2.** **Herramientas de Ingeniería** 4](#_Toc144990327)

[**III.** **DESARROLLO DE LA SOLUCIÓN** 4](#_Toc144990328)

[**3.1.** **Formulación del Pseudocódigo** 4](#_Toc144990332)

[**3.2.** **Implementación del Algoritmo** 4](#_Toc144990333)

[**IV.** **RESULTADOS** 4](#_Toc144990334)

[**4.1.** **Análisis empírico** 4](#_Toc144990339)

[**4.2.** **Evaluación** 4](#_Toc144990340)

[**V.** **CONCLUSIONES** 5](#_Toc144990341)

[**VIII.** **REFERENCIAS O BIBLIOGRAFÍA** 5](#_Toc144990342)

[**IX.** **ANEXOS** 5](#_Toc144990343)

**INDICE DE TABLAS**

**INDICE DE FIGURAS**

**INDICE DE ANEXOS**

# **RESUMEN.**

Máximo 200 palabras (se realiza al final del proyecto)

Palabras clave: Incluir máximo 05 palabras clave

Abstract (All traslated to English)

Key (All traslated to English)

# **INTRODUCCIÓN.**

* 1. **Definición de objetivos**

### Objetivo General

Crear un sistema multifuncional de cajero automático que facilite la administración completa de transacciones bancarias, incluyendo retiros, depósitos, transferencias, pagos de servicios y consultas de saldos y movimientos. Este sistema debe garantizar una gestión adecuada de los cajeros en distintas ubicaciones y una actualización precisa de los saldos tanto de los clientes como de los dispensadores de billetes.

### Objetivos Específicos

1. **Gestión de Cajeros:**
   * Implementar un sistema que permita gestionar múltiples cajeros automáticos en distintas ubicaciones, con la capacidad de monitorear y actualizar la cantidad de billetes en cada denominación para cada cajero.
2. **Administración de Clientes:**
   * Implementar funciones que permitan agregar nuevos clientes, asignarles cuentas y contraseñas, y posibilitar la actualización de su información bancaria.
   * Asegurar la seguridad y confidencialidad de los datos de los clientes a través de métodos de autenticación.
3. **Transacciones de Retiro:**

* Facilitar a los clientes la realización de retiros de efectivo, comprobando previamente la disponibilidad de saldo tanto en su cuenta como en el dispensador de billetes.
* Producir un desglose detallado de los billetes entregados y actualizar tanto el saldo de la cuenta del cliente como la cantidad de billetes en el dispensador.

1. **Transacciones de Depósito:**
   * Implementar la funcionalidad de depósitos de efectivo, solicitando el desglose de los billetes depositados.
   * Actualizar el saldo de la cuenta del cliente y la cantidad de billetes en el dispensador de manera precisa.
2. **Transacciones de Transferencia:**

* Facilitar el movimiento de dinero entre cuentas de clientes, asegurando que tanto la cuenta origen como la cuenta destino registren la operación con la correcta disminución del saldo en la cuenta de origen y el incremento en la cuenta de destino.
* Registrar de manera detallada todas las transferencias realizadas, asegurando que la información esté disponible para consultas futuras.

1. **Pago de Servicios:**
   * Implementar la funcionalidad de pago de servicios, permitiendo que los clientes realicen pagos directamente desde sus cuentas bancarias.
   * Garantizar que el saldo de la cuenta del cliente se disminuya de acuerdo con el monto del servicio pagado.
2. **Consulta de Saldos y Movimientos:**
   * Desarrollar una interfaz que permita a los clientes consultar sus saldos y movimientos de cuenta de manera detallada y en tiempo real.
   * Asegurar que la información proporcionada sea precisa y refleje todas las transacciones realizadas.
3. **Seguridad:**
   * Integrar mecanismos de autenticación de usuarios y registro de actividades.
4. **Interfaz de Usuario:**
   * Diseñar una interfaz de usuario intuitiva y accesible que facilite la interacción de los clientes con el sistema de cajero automático.
   * Asegurar que la interfaz sea compatible con diversos dispositivos y resoluciones, proporcionando una experiencia de usuario consistente y eficiente.

Con estos objetivos, el sistema de cajero automático multifunción garantizará una gestión eficiente de las transacciones bancarias y una experiencia satisfactoria para los usuarios, al tiempo que asegura la integridad y seguridad de las operaciones.

1. 1. **Descripción del problema**

El problema implica desarrollar un sistema completo para cajeros automáticos que pueda realizar varias funciones financieras como retiros, depósitos, transferencias entre cuentas, pagos de servicios, y consultar saldos y movimientos. Este sistema debe manejar múltiples cajeros ubicados en diferentes lugares, permitir la gestión de clientes y sus cuentas, incluyendo la creación y actualización de cuentas con contraseñas. También debe administrar un dispensador de billetes, con la capacidad de cargar y actualizar billetes de diferentes denominaciones. El sistema debe asegurar la correcta gestión de saldos tanto en las cuentas de los clientes como en el dispensador, registrando todas las transacciones realizadas para futuras consultas.

# **ANÁLISIS DE LA SOLUCIÓN**

* 1. **Elección de la solución**

*Justifica las ventajas y desventajas del algoritmo implementado en una tabla.*

|  |  |
| --- | --- |
| *Algoritmo: “Guardar (Cajeros)”* | |
| *Ventajas* | *Desventajas* |
| * **Persistencia de Datos:** Al igual que la función anterior, utiliza pickle.dump para serializar y guardar los datos de los cajeros en un archivo binario (Cajero.pkl). Esto asegura que los datos se conserven incluso después de cerrar el programa. * **Flexibilidad en la Estructura de Datos:** Puede guardar estructuras complejas de datos (en este caso, parece ser una lista o diccionario de cajeros) de manera sencilla y eficiente, lo cual es útil para programas que manejan múltiples objetos o datos complejos. | * **Dependencia de la Biblioteca pickle:** Al igual que antes, pickle es específico de Python y puede tener problemas de compatibilidad entre versiones. Esto puede dificultar la interoperabilidad de los archivos generados con otros sistemas o lenguajes. * **Gestión de Errores y Mantenimiento:** La gestión de errores durante la serialización y deserialización con pickle puede ser compleja. Los errores al cargar datos podrían causar fallos en la aplicación o pérdida de datos si no se manejan adecuadamente. |

|  |  |
| --- | --- |
| *Algoritmo: “Añadir Dinero”* | |
| *Ventajas* | *Desventajas* |
| * **Actualización Dinámica:** Permite añadir billetes nuevos al cajero automáticamente. Si encuentra un billete del mismo valor en la lista self.Billetes, incrementa la cantidad de ese tipo de billete en el cajero. * **Eficiencia en la Implementación:** Utiliza un bucle for para buscar el billete por su valor en self.Billetes, lo cual es una forma eficiente de actualizar la cantidad de billetes existentes. | * **Falta de Validación:** No realiza una validación exhaustiva de los datos de entrada (BilletesNuevos). Podría beneficiarse de comprobar si los billetes nuevos son válidos antes de añadirlos al cajero. * **Seguridad y Concurrencia:** No hay medidas implementadas para gestionar la concurrencia en la actualización del dinero del cajero. En entornos donde múltiples operaciones podrían modificar el estado del cajero simultáneamente, esto podría llevar a inconsistencias o errores. |

|  |  |
| --- | --- |
| *Algoritmo: “Cargar Dinero”* | |
| *Ventajas* | *Desventajas* |
| * **Cálculo Preciso del Dinero en el Cajero:** Itera sobre la lista de billetes (self.Billetes) y calcula correctamente la cantidad total de dinero en el cajero multiplicando la cantidad de cada tipo de billete por su valor. * **Eficiencia en el Cálculo:** Utiliza un bucle for simple para sumar el valor total de todos los billetes en el cajero, lo cual es eficiente y directo. | * **Dependencia de la Estructura de Datos:** La función asume que self.Billetes es una lista de objetos Billete, lo cual limita su flexibilidad para trabajar con diferentes estructuras de datos o formatos de billetes sin modificaciones adicionales. * **Potencial Falta de Flexibilidad:** No proporciona opciones para manejar excepciones o errores que podrían surgir durante el cálculo del dinero en el cajero, lo que puede limitar su robustez en entornos de aplicación real donde la entrada de datos puede variar o ser impredecible. |

|  |  |
| --- | --- |
| *Algoritmo: “Retirar Dinero”* | |
| *Ventajas* | *Desventajas* |
| * **Manejo Detallado de Billetes:** La función permite retirar una cantidad específica de dinero del cajero automático, calculando la combinación óptima de billetes para cumplir con la cantidad solicitada (cantidad). Esto se logra mediante un algoritmo que itera sobre los billetes disponibles y los utiliza en la cantidad adecuada. * **Registro de Movimientos:** Después de realizar el retiro, la función registra el movimiento en la cuenta del usuario (Usuario), añadiendo detalles como la cantidad retirada y el tipo de transacción ("Retiro"). | * **Complejidad y Potencial para Errores:** El manejo de billetes y la lógica de la función pueden volverse complejos y propensos a errores en escenarios donde la cantidad solicitada no se puede cubrir exactamente con la combinación de billetes disponibles. Esto podría llevar a inconsistencias en el estado del cajero o en los registros de transacciones. * **Seguridad y Validación:** La función no realiza una validación exhaustiva de la cantidad solicitada (cantidad) antes de iniciar el proceso de retiro. Podría beneficiarse de una validación previa para asegurar que la cantidad solicitada no exceda los límites o disponibilidad de fondos del usuario y del cajero. |

|  |  |
| --- | --- |
| *Algoritmo: “Transferencia”* | |
| *Ventajas* | *Desventajas* |
| * **Transferencia de Fondos:** Permite transferir una cantidad específica de dinero desde la cuenta del Usuario a la cuenta del receptor (Usuario). Esto facilita el intercambio de fondos entre dos cuentas distintas. * **Registro de Movimientos:** Después de completar la transferencia, la función registra automáticamente el movimiento en las listas de movimientos tanto del Usuario como del receptor, proporcionando detalles como la cantidad transferida y la dirección de la transferencia. | * **Validación Limitada:** La función valida únicamente si el Usuario tiene suficiente dinero para realizar la transferencia. No verifica otras condiciones importantes, como límites de transferencia o la existencia del receptor. Esto podría llevar a problemas si los datos no están completamente validados antes de realizar la operación. * **Seguridad y Transacciones Concurrentes:** No se implementan medidas de seguridad adicionales, como bloqueos para manejar transacciones concurrentes. En entornos con múltiples transacciones simultáneas, esto podría causar inconsistencias o errores en los saldos de las cuentas involucradas. |

|  |  |
| --- | --- |
| *Algoritmo: “Deposito”* | |
| *Ventajas* | *Desventajas* |
| * **Manejo de Depósitos de Efectivo:** Permite realizar depósitos de efectivo en el cajero automático, incrementando la cantidad de billetes adecuados (billetes) en el cajero y añadiendo el monto total del depósito (monto) al saldo del Usuario. * **Registro de Movimientos:** Después de completar el depósito, la función registra automáticamente el movimiento en la lista de movimientos del Usuario, detallando el monto depositado y el tipo de transacción ("Depósito en Efectivo"). | * **Validación de Billetes:** La función asume que los billetes proporcionados en billetes son válidos y compatibles con los tipos de billetes (Billete) almacenados en self.Billetes. No hay validación explícita de la autenticidad de los billetes o de la consistencia entre los datos proporcionados y los almacenados. * **Seguridad y Transacciones Simultáneas:** No se implementan medidas adicionales para asegurar la integridad de los datos durante transacciones simultáneas. En entornos con múltiples depósitos concurrentes, esto podría resultar en errores de sincronización o inconsistencias en los saldos y registros de transacciones. |

|  |  |
| --- | --- |
| *Algoritmo: “Guardar (Usuario)”* | |
| *Ventajas* | *Desventajas* |
| * **Persistencia de Datos:** Utiliza pickle.dump para serializar y guardar la lista de usuarios (Usuarios) en un archivo binario (Usuario.pkl). Esto asegura que los datos se conserven entre sesiones del programa. * **Simplicidad de Implementación:** La función es directa y utiliza la biblioteca estándar de Python (pickle) para la serialización, lo cual simplifica el proceso de guardar estructuras complejas de datos como listas de objetos Python en archivos. | * **Dependencia de la Biblioteca pickle:** Al igual que otras funciones que utilizan pickle, esta función depende de una característica específica de Python que puede tener problemas de compatibilidad entre versiones. Esto puede dificultar la interoperabilidad de los archivos generados con otros sistemas o lenguajes. * **Seguridad y Validación:** pickle no proporciona mecanismos integrados para garantizar la seguridad de los datos serializados. Los archivos pickle pueden ser vulnerables a ataques si se cargan datos de origen no confiable. Además, los errores durante la deserialización pueden ser difíciles de depurar y pueden llevar a la pérdida de datos si no se manejan adecuadamente. |

* 1. **Herramientas de Ingeniería**
* *Texto

  Descripción generada automáticamentePython*

**Texto

Descripción generada automáticamente**

# **DESARROLLO DE LA SOLUCIÓN**

1. *Texto

   Descripción generada automáticamente*
3. 1. **Formulación del Pseudocódigo**

*Texto

Descripción generada automáticamente*

*Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente*

*Texto

Descripción generada automáticamente*

*Texto

Descripción generada automáticamente*

* 1. **Implementación del Algoritmo**

*Implementar el algoritmo formulado, haciendo uso del lenguajes de programación y herramientas trabajadas en el curso (colocar el código y capturas del funcionamiento) (anexar software, link de repositorio o en un zip enviarlo junto con el informe)*

# **RESULTADOS**

4. 1. **Análisis empírico**

*Realizar un análisis empírico (Un análisis "empírico" es el que se basa en experimentación y observación real de los resultados) para probar que el algoritmo implementado sea el más adecuado para solucionar el caso, lo presenta en una tabla.*

|  |  |  |
| --- | --- | --- |
| **Funcionalidad** | **Resultados** | **Acción** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. **Evaluación**

*Analizar a través de una tabla de doble entrada las ventajas y desventajas de haber utilizado otros algoritmos para solucionar el caso planteado*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Algoritmo elegido** | **Otro algoritmo 1** | **Otro algoritmo 2** |
| **Ventajas** |  |  |  |
| **Desventajas** |  |  |  |

# **CONCLUSIONES**

*Se debe tener una conclusión por cada objetivo*

# **REFERENCIAS O BIBLIOGRAFÍA**

*<<Anotar los libros, manuales, guías, páginas electrónicas que se hayan utilizado para la elaboración de este informe en formato APA>>.*

# **ANEXOS**

*Capturas de pantalla de proyecto*

*Link del producto software*

**NOTAS ACLARATORIAS**

1. El Informe de proyecto de fin de curso será redactado con procesador de textos en fuente Arial, tamaño 11, alineación justificada, interlineado 1,5, hoja A4. Utilizar formato APA 7ma edición.
2. Sobre el tema de proyecto

* Integración con el aprendizaje previo (temas aprendidos en otros cursos)
* Promueve un nuevo aprendizaje
* Brinda experiencias realistas
* Balance entre complejidad y carga de trabajo

1. Comunicación

* Lenguaje profesional
* Organización de la presentación
* Presentación efectiva
* Figuras y formato
* Redacción y gramática