**SIMULADOR DE BANCO (Noviembre de 2024)**

Autores: Torres Pachón Juan Esteban, González Gonzáles Dinalut y Ruiz Rincon Jessica Jineth estudiantes de la foundation universitaria de San gil unisangil (SEDE-CHIQUINQUIRA)

***Resumen*—En el presente documento hemos diseñado un algoritmo para simular la gestión de cuentas bancarias, donde se pueden realizar operaciones de depósitos, retiros y consultas de saldo. Además de simular el procesamiento de clientes en la fila del banco Por lo que aplicamos una metodología en la que establecimos los requisitos y limitaciones del algoritmo. Además, utilizamos PSeInt para crear un pseudocódigo que luego implementamos en Python.**

**Palabras clave- cuenta bancaria, depositicos, retiros, consulta de saldo, filas de clientes, Matrices, Funciones, Clases.**

INTRODUCCIÓN

 En este documento se busca diseñar un algoritmo donde se simule la gestión de cuentas bancarias que permita hacer depósitos, retiros y consultas de saldo, además de simular el procesamiento de clientes en la fila del banco.

*ANÁLISIS*

1. *Contexto*

Simular un banco es útil para entender cómo funciona la gestión de las cuentas bancarias por tal motivo es necesario crear un sistema bancario donde se puedan simular la realización de operaciones de depósito, retiro y consulta de saldo. Además de simular el procesamiento de clientes en las filas del banco.

*B. Población*

Usuarios que desean entender como funciona la gestión de las cuentas bancarias y el procesamiento de clientes en las colas del banco.

*C. Limitaciones y alcance*

Las limitaciones y alcances que se pudieron identificar son las siguientes:

Limitaciones

* El usuario tiene que ingresar datos.
* Tienen opciones limitadas que son depósitos, retiros o consultar saldo.

Alcances

* El usuario podrá ingresar los datos requeridos.
* El usuario puede simular un depósito bancario.
* El usuario puede simular un retiro.
* El usuario puede simular una consulta de saldo.
* El usuario puede entender como funciona la gestión de una cuenta bancaria.
* El usuario puede entender como funciona el procesamiento de clientes en una fila del banco.

III. OBJETIVOS

*Objetivo general*

Diseñar un algoritmo donde el usuario pueda simular la gestión de cuentas bancarias, haciendo depósitos, retiros, y consultas de saldo, además de gestionar el procesamiento de clientes el las filas del banco.

1. *Objetivos específicos*

* Determinar el lenguaje de programación que se va a usar para crear el algoritmo.

* Solicitar al usuario los datos necesarios para la simulación de depósitos bancarios

* Solicitar al usuario los datos específicos para simular un retiro bancario.
* Solicitar al usuario los datos específicos para simular una consulta de saldo en una cuenta bancaria.
* Solicitar al usuario los datos específicos para simular la gestión de usuarios en una fila del banco.

* Generar la salida correspondiente de los datos.

* Presentar al usuario un programa que pueda manejar de forma comprensible.

IV. ESPECIFICACIÓN DE REQUERIMIENTOS.

1. *CON RESPECTO AL DISPOSITIVO*

A continuación, se indican los requerimientos mínimos:

* Mínimo 4 GB de RAM.
* Sistema operativo: Windows 8 en adelante.
* CPU: Intel Core i3.
* Disco: HDD o SSD.

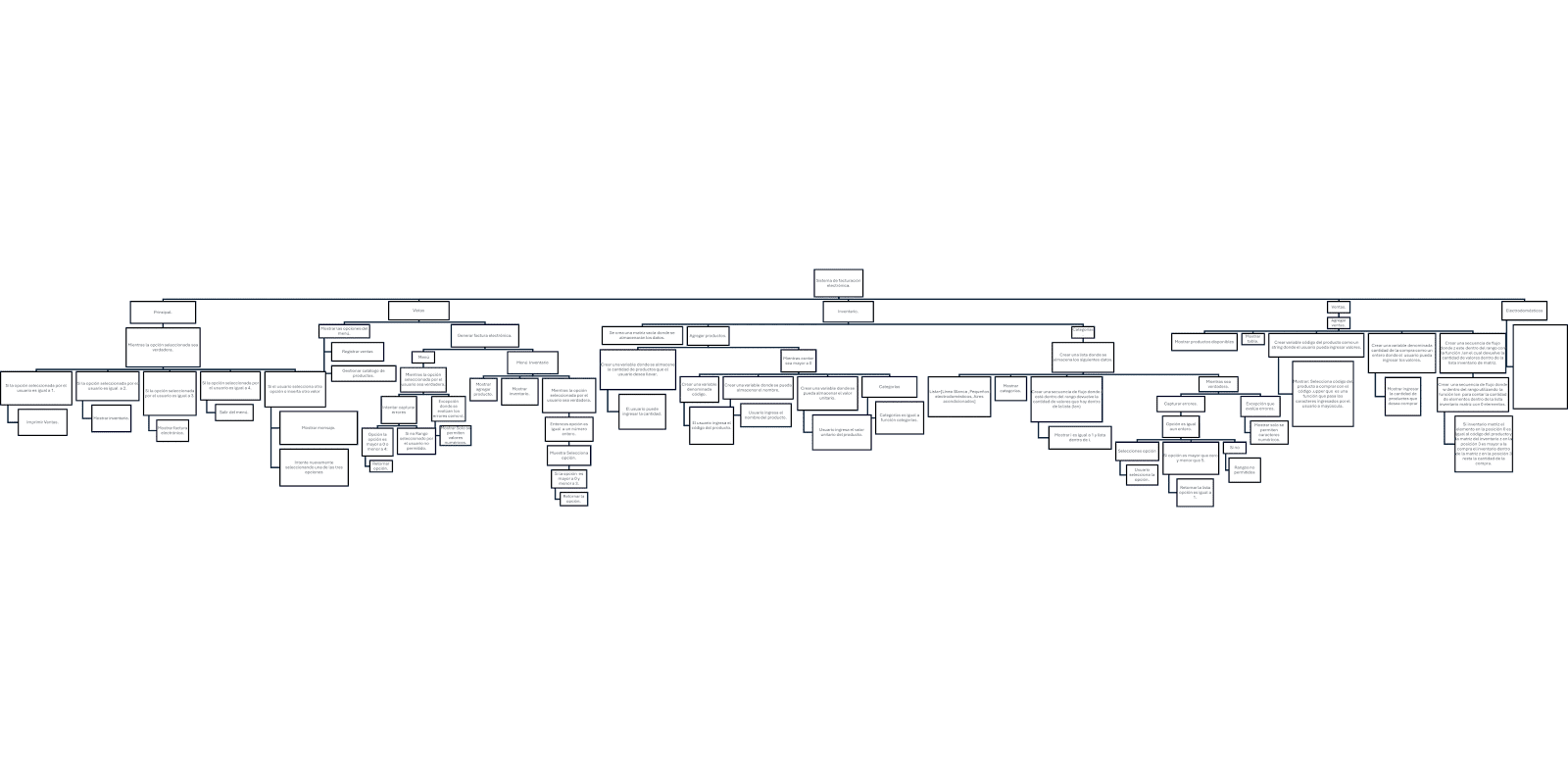
1. *CON RESPECTO AL USUARIO.*

*Requisitos mínimos*

* *El usuario debe ingresar los datos correspondientes para la simulación de la cuenta bancaria, tales como depósitos, retiros o consultas de saldo.*
* *El usuario debe ingresar los datos correspondientes para simular la gestión de usuarios en una fila del banco.*
* *El usuario debe contar con una necesidad seleccionada, la cual es entender como funciona la gestión de las cuentas bancarias y la gestión de usuarios en las filas del banco.*

V. DISEÑO DEL ALGORITMO

Utilizamos un programa generador de pseudocódigo denominado PSeInt, el cual utilizamos como guía, para luego usar el lenguaje de programación Python, siendo este el estructurador del software requerido, diseñamos mapa conceptual y diseñamos la tabla con las clases.



*Ilustración 1. Análisis basado en la programación modular basado en el algoritmo.*

|  |  |
| --- | --- |
| Clase | Cuenta bancaria |
| Atributos | 1. seleccionar cuenta |
| 2. depositar |
| 3. retirar |
| 4. consultar |
|  |
| Método | 1. procesar clientes |
| 2. opciones de operación |
| 3. ejecutar operación y mostrar |
|  |

*Ilustración 2. Clase dentro de la simulación de cuenta bancaria la cual se divide en atributos y método.*

VI.CODIFICACIÓN.

 Para la codificación del algoritmo utilizamos los siguientes programas:

* PSeInt.
* Visual Studio Conde.
* python
* Git hub.

*Instrucciones del código fuente*

1. *Código fuente*

Dentro del código fuente se importó la librería random y deque del módulo collections para la gestión de las filas en el banco.

­­­­­­­­­­­­­­­­­­­­­­­­­­­­­Trabajamos con programación orientada a objetos, donde el objeto es CuentaBamcaria, la cual es ejecutada por funciones teniendo 3 opciones de ejecucuion por cada cliente en la fila.

Se utiliso el while y el if para la selección y procesamiento de cada opción. A continuación se anexan imágenes del código

­­­­­­­­­­­­­­­­­­­­­­­­­­­­­

import random

from collections import deque

class CuentaBancaria:

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

        self.titular = titular

        self.saldo = saldo\_inicial

    def depositar(self, cantidad):

        self.saldo += cantidad

        print(f"Depósito de {cantidad} realizado. Saldo actual: {self.saldo}")

    def retirar(self, cantidad):

        if cantidad > self.saldo:

            print("Fondos insuficientes.")

        else:

            self.saldo -= cantidad

            print(f"Retiro de {cantidad} realizado. Saldo actual: {self.saldo}")

    def consultar\_saldo(self):

        print(f"Saldo actual de {self.titular}: {self.saldo}")

Imagen. 3. Código fuente.

*Imagen. 2:En esta imágenes se muestra el diseño del algoritmo donde se utiliza el lenguaje de programación python para ejecutarlo .*

*VII. EJECUCIÓN DEL PROGRAMA.*

En este caso comenzamos con la apertura de la terminal de Git Bash. Si ya se creó el repositorio remoto y se sincronizan las cuentas de los desarrolladores con el repositorio local, se envía el archivo al Staging área con el comando git add.  Luego ejecutamos el git commit -m “Comentario” para confirmar los cambios, donde, posteriormente usamos git push origin o pb master o main dependiendo la rama que se esté usando con el fin de almacenar la información en el repositorio remoto de GitHub.

VIII. VERIFICACIÓN Y DEPURACIÓN

En el versionamiento de la aplicación iniciando desde la versión -V 0.1. Este código inicial fue la base estructural con la cual mediante un análisis anteriormente realizado, se pudo establecer los requisitos que debe cumplir el programa para funcionar según lo dictaminado por el usuario final.

Dentro de este código tenemos 3 métodos y 3 funciones:

Depositar: método para depositar una cantidad en la cuenta.

Retirar: método para retirar una cantidad de la cuenta si hay suficiente saldo para el retiro.

Consultar saldo: método para consultar el saldo de la cuenta.

seleccionar\_cuenta(): Función encargada de la selección de la cuenta en la fila.

Menú(): Función que muestra el menú de operaciones y permite seleccionar la operación deseada.

procesar\_clientes(): función que procesa las operaciones que el cliente desea realizar.

1. ***Interna.***

*Toda la información relacionada con el desarrollo del código se encuentra almacenada en un repositorio en GitHub.*

*Enlace:*

[*https://github.com/Jessica-Ruiz/Laboratorio2*](https://github.com/Jessica-Ruiz/Laboratorio2)

***Descripción del problema:***

Se identificó l necesidad de diseñar un algoritmo. Desarrollar una simulación de gestión de cuentas bancarias. que permita consultar el saldo de la cuenta, hacer depósitos y retiros de la misma.

***Modelos de análisis:***

Como modelo de análisis podemos identificar que se utilizó la programación orientada a objetos para resolver el problema

Manual de usuario.

Ccomo usuario principalmente visualizará un menú dentro del cual puede seleccionar varias opciones.

1.) Depositar

2.) Retirar

3.) Consultar saldo.

Selecciona opción:

Si selecciona la opción 1. Depositar se le pedirá la cantidad a depositar, la cual de ingresar en números enteros.

Si selecciona la opción 2 Retirar se le pedirá la cantidad que desea retirar, la cual debe ser ingresada en números enteros.

Si selecciona la opción 3. Consultar saldo se le mostrara el saldo que actualmente tiene la cuenta.

VIII CONCLUSIONES.

* En el presente trabajo logramos describir el diseño y codificación de un algoritmo el cual permite al cliente simular la gestión de cuentas en un banco.

* Se desarrolló una fase de planeación la cual nos permitió garantizar una buena gestión de las fases del proyecto, tomando en cuenta fases como: requerimientos del cliente, el análisis y diseño de la estructura de la aplicación, la respectiva codificación teniendo en cuenta la fase anterior y por último la respectiva documentación aquí escrita.

* Implementamos un análisis situacional específico para captar con mayor detalle los requisitos del cliente, el cual utilizamos para escribir el código de manera correcta

* Incorporamos la evaluación y pruebas para el programa, con ellas constatamos y verificamos que el código realizado por nosotros los desarrolladores esté a fin con los requerimientos del cliente.

* Aprendimos que es necesario aplicar una metodología que hace más comprensible y efectivo el proceso a la hora de desarrollar un algoritmo.

* Empleamos un generador de pseudocódigo denominado PSeInt como guía para estructurar de una forma clara el algoritmo en Python.

* Para realizar la codificación del código utilizamos varios programas como Visual Studio Code y GitHub los cuales facilitan el trabajo en grupo y la gestión de información.