Ingeniería
Desarrollo de Software

Ingeniería en Desarrollo de Software

Nombre de la actividad

Actividad 3

Programa 2 parte 2

Materia

Lenguaje de programación IV

Tutor: Marco Alonso Rodríguez Tapia

Alumno: Javier Cajero Valiente

Fecha de entrega: 26 de noviembre de 2023

ÍNDICE

- Contextualización 3
- Introducción 4
- Descripción 4
- Justificación 5
- Desarrollo 7
  - Codificación 7
  - Prueba del programa 9
- Conclusión 12
- Referencias 13

Contextualización

El Banco Mexicano necesita un programa para su banca en línea. En este se podrán realizar diversas acciones, como depositar, retirar, consultar saldo o salir. La app deberá estar creada con el lenguaje Swift.

Actividad:

En la actividad anterior, se creó la interfaz del menú principal del programa de Banco Mexicano. Además, se crearon las opciones “Depósito” y “Retiro”. Ahora, se deberá terminar de darle funcionalidad a las opciones faltantes: “Saldo” y “Salir”.

En caso de que el usuario ingrese la opción Saldo, el programa deberá de mostrar en pantalla:

- El saldo del usuario
- Preguntar si desea realizar otra operación.

En caso de que el usuario ingrese la opción Salir, el programa deberá mostrar en pantalla que se ha cerrado la sesión.

En caso de que el usuario ingrese la opción Retiro, el programa deberá ser capaz de capturar por teclado los siguientes datos:

- Si el cliente ingresa por primera vez, deberá mostrar un mensaje donde diga que no cuenta con saldo.
- Si el cliente ingresa por segunda vez, y ya se tiene dinero en la cuenta, deberá mostrar lo siguiente:
- Cantidad a retirar. El sistema deberá validar si cuenta con el saldo a retirar; y si no, mostrar un mensaje diciendo que no cuenta con el saldo tecleado.
- Preguntar si desea realizar otro retiro; si la respuesta es no, deberá preguntar si desea realizar otra operación.

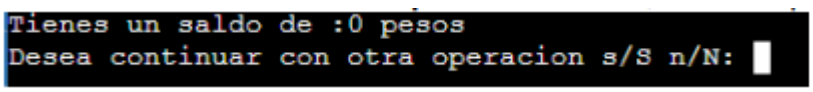
Introducción

En esta segunda actividad hablaremos de como diseñar una aplicación para un banco, donde necesita un programa que funcione y en esa aplicación se puedan realizar depósitos y retiros por parte del cliente. Se requiere que se puedan mostrar los servicios que el cliente realice dentro de la misma aplicación. Para realizarla, se deberá utilizar el lenguaje de programación Swift en Replit.
Conoceremos el compilador Replit y sus características y cuales seria las ventajas y beneficio del lenguaje Swift y la sencillez al utilizarlo.
Replit es una plataforma en línea que proporciona un entorno de desarrollo integrado (IDE, por sus siglas en inglés) basado en la nube para varios lenguajes de programación, incluido Swift. Ofrece a los desarrolladores un espacio de trabajo virtual donde pueden escribir, probar y ejecutar código directamente desde el navegador, sin necesidad de instalar software adicional en sus dispositivos locales.

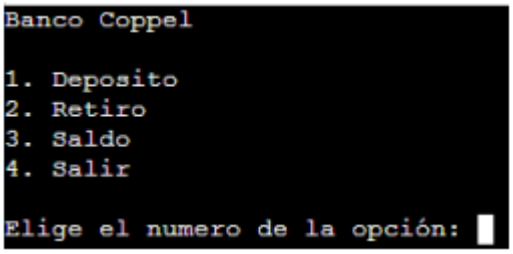
Al estar basado en la nube, Replit permite a los desarrolladores acceder a sus proyectos desde cualquier lugar con conexión a internet. Además, ofrece funciones de colaboración en tiempo real, lo que facilita el trabajo conjunto en proyectos, ya que varios desarrolladores pueden contribuir y ver cambios simultáneamente.
Replit simplifica el flujo de trabajo de desarrollo al integrar la escritura de código, la ejecución y la visualización de resultados en una única plataforma. Esto agiliza el proceso de desarrollo y permite a los programadores centrarse en la lógica de su aplicación.

Descripción

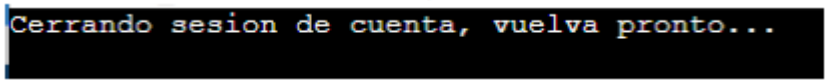
Realizar la codificación del programa solicitado. Tener en cuenta las especificaciones dadas en la actividad anterior y ahora le agregaremos las siguientes especificaciones:
En caso de seleccionar la “Opción: 3 Saldo”, deberá aparecer lo siguiente:



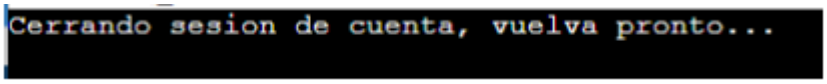
En caso de teclear la letra “S”, deberá mostrar la siguiente pantalla:



En caso de teclear la letra “N”, deberá volver a mostrar la siguiente pantalla:



En caso de seleccionar la “Opción 4: Salir”, deberá aparecer lo siguiente:



Para finalizar, correr el programa y ver si funciona correctamente lo creado hasta el momento.

Justificación

Es momento de poder justificar el por qué debemos de conocer a detalle la implementación de nuestra actividad, la elección de Replit con Swift como nuestra plataforma de desarrollo ofrece ventajas sustanciales que mejoran la eficiencia, la colaboración y la accesibilidad para nuestro equipo. Esta decisión respalda nuestro compromiso de lograr el éxito en nuestro proyecto y fomenta un entorno de desarrollo ágil y eficaz, donde se necesita un programa que funcione como una tienda de ropa en línea. Se requiere que se puedan mostrar productos y el cliente realice su compra en la misma aplicación.
Replit es una plataforma en línea que proporciona un entorno de desarrollo integrado (IDE, por sus siglas en inglés) basado en la nube para varios lenguajes de programación, incluido Swift. Ofrece a los desarrolladores un espacio de trabajo virtual donde pueden escribir, probar y ejecutar código directamente desde el navegador, sin necesidad de instalar software adicional en sus dispositivos locales.

Algunas características destacadas de Replit incluyen:

Accesibilidad: Replit es accesible desde cualquier lugar con conexión a internet y un navegador web compatible, lo que facilita la colaboración y el desarrollo en equipos distribuidos.

Colaboración en Tiempo Real: Permite a los desarrolladores colaborar en proyectos de forma simultánea, viendo los cambios en tiempo real, lo que facilita el trabajo en equipo.

Facilidad de Uso: Proporciona un entorno de desarrollo simplificado, eliminando la necesidad de configurar y mantener entornos locales complejos. Esto hace que sea más fácil para los principiantes comenzar a programar.

Soporte para Múltiples Lenguajes: Además de Swift, Replit es compatible con varios lenguajes de programación, lo que brinda flexibilidad a los desarrolladores para trabajar en diferentes proyectos.

Integración con Plataformas Externas: Puede integrarse con servicios como GitHub para la gestión de versiones y colaboración en proyectos.

Ejecución en la Nube: Replit permite ejecutar y probar el código en la nube, lo que significa que los desarrolladores no necesitan preocuparse por la infraestructura subyacente. Trabajar con el lenguaje de programación Swift ofrece diversas ventajas y beneficios, especialmente en el contexto del desarrollo de aplicaciones para dispositivos Apple.

Rendimiento Optimizado: Swift ha sido diseñado para ser rápido y eficiente. Proporciona un rendimiento similar al de lenguajes de bajo nivel como C++ y Objective-C, lo que lo hace adecuado para aplicaciones que requieren alta velocidad y capacidad de respuesta.

Seguridad: Swift ha sido desarrollado con un enfoque en la seguridad del código

Sintaxis Clara y Concisa: La sintaxis de Swift es moderna, fácil de leer y escribir. Su diseño busca minimizar la redundancia y mejorar la legibilidad del código, lo que facilita la escritura y el mantenimiento de aplicaciones.

Interoperabilidad con Objective-C: Swift es interoperable con el lenguaje de programación Objective-C, lo que permite a los desarrolladores utilizar bibliotecas y frameworks existentes de Objective-C en proyectos Swift.

Desarrollo en Apple Ecosystem: Swift está estrechamente integrado con las herramientas y tecnologías del ecosistema Apple, como Xcode (IDE oficial de Apple), Cocoa Touch (framework para iOS), y SwiftUI (framework para interfaces de usuario declarativas).
Comunidad Activa: Swift cuenta con una comunidad activa de desarrolladores que comparten conocimientos, recursos y contribuyen al desarrollo continuo del lenguaje. Esto facilita el aprendizaje y la resolución de problemas.
Soporte de Apple: Al ser respaldado por Apple, Swift recibe actualizaciones regulares y mejoras, asegurando que esté alineado con las últimas tecnologías y estándares de la industria.

Desarrollo

Codificación

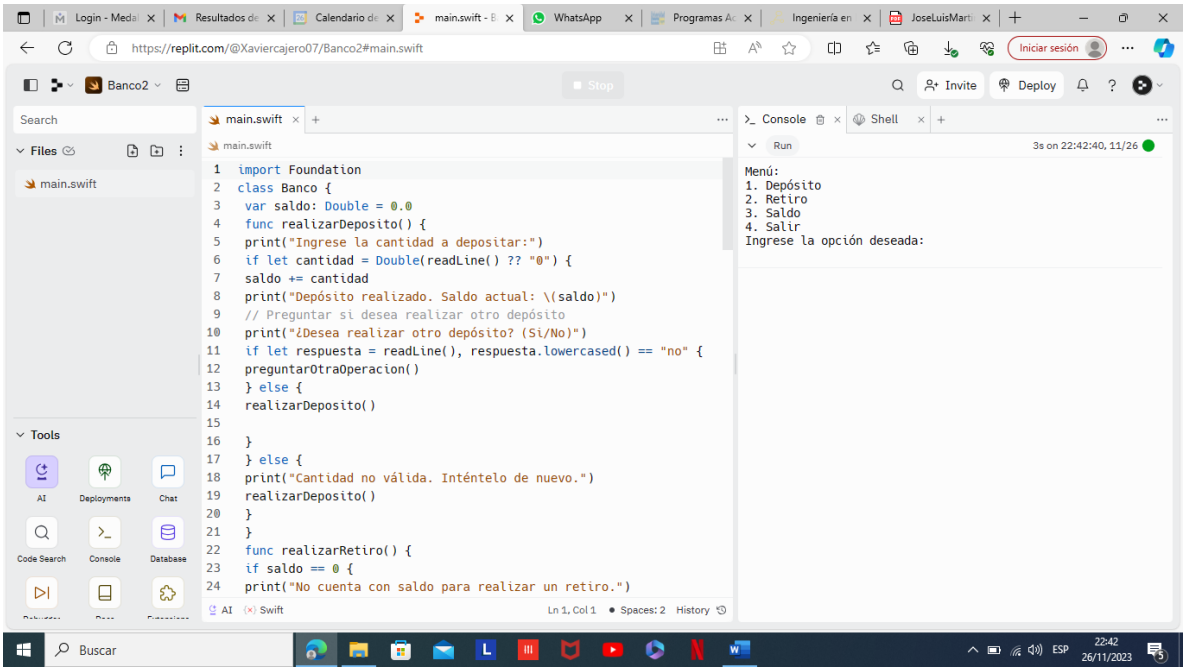
```
import Foundation
class Banco {
 var saldo: Double = 0.0
 func realizarDeposito() {
 print("Ingrese la cantidad a depositar:")
 if let cantidad = Double(readLine() ?? "0") {
 saldo += cantidad
 print("Depósito realizado. Saldo actual: \(saldo)")
 // Preguntar si desea realizar otro depósito
 print("¿Desea realizar otro depósito? (Si/No)")
 if let respuesta = readLine(), respuesta.lowercased() == "no" {
 preguntarOtraOperacion()
 } else {
 realizarDeposito()

 }
 } else {
 print("Cantidad no válida. Inténtelo de nuevo.")
 realizarDeposito()
 }
 }
 func realizarRetiro() {
 if saldo == 0 {
 print("No cuenta con saldo para realizar un retiro.")
 } else {
 print("Ingrese la cantidad a retirar:")
 if let cantidad = Double(readLine() ?? "0") {
 if cantidad <= saldo {
 saldo -= cantidad
 print("Retiro realizado. Saldo actual: \(saldo)")
```

```
print("¿Desea realizar otro retiro? (Si/No)")
if let respuesta = readLine(), respuesta.lowercased() == "no" {
preguntarOtraOperacion()
} else {
realizarRetiro()
}
} else {
print("Saldo insuficiente. No se puede realizar el retiro.")
realizarRetiro()
}
} else {
print("Cantidad no válida. Inténtelo de nuevo.")
realizarRetiro()
} } }
func consultarSaldo() {
print("Saldo actual: \(saldo)")
preguntarOtraOperacion()
}
func salir() {
print("Gracias por utilizar nuestro servicio. ¡Hasta luego!")
exit(0)
}
func preguntarOtraOperacion() {
print("¿Desea realizar otra operación? (Si/No)")
if let respuesta = readLine(), respuesta.lowercased() == "si" {
mostrarMenu()
} else {
salir()
}}
func mostrarMenu() {
print("Menú:")
print("1. Depósito")
print("2. Retiro")
print("3. Saldo")
print("4. Salir")
print("Ingrese la opción deseada:")
if let opcion = Int(readLine() ?? "0") {
switch opcion {
case 1:
realizarDeposito()
case 2:
realizarRetiro()
case 3:
consultarSaldo()
case 4:
salir()
default:
print("Opción no válida. Inténtelo de nuevo.")
mostrarMenu()
} } else {
print("Opción no válida. Inténtelo de nuevo.")
mostrarMenu()
} }}
let banco = Banco()
banco.mostrarMenu()
```

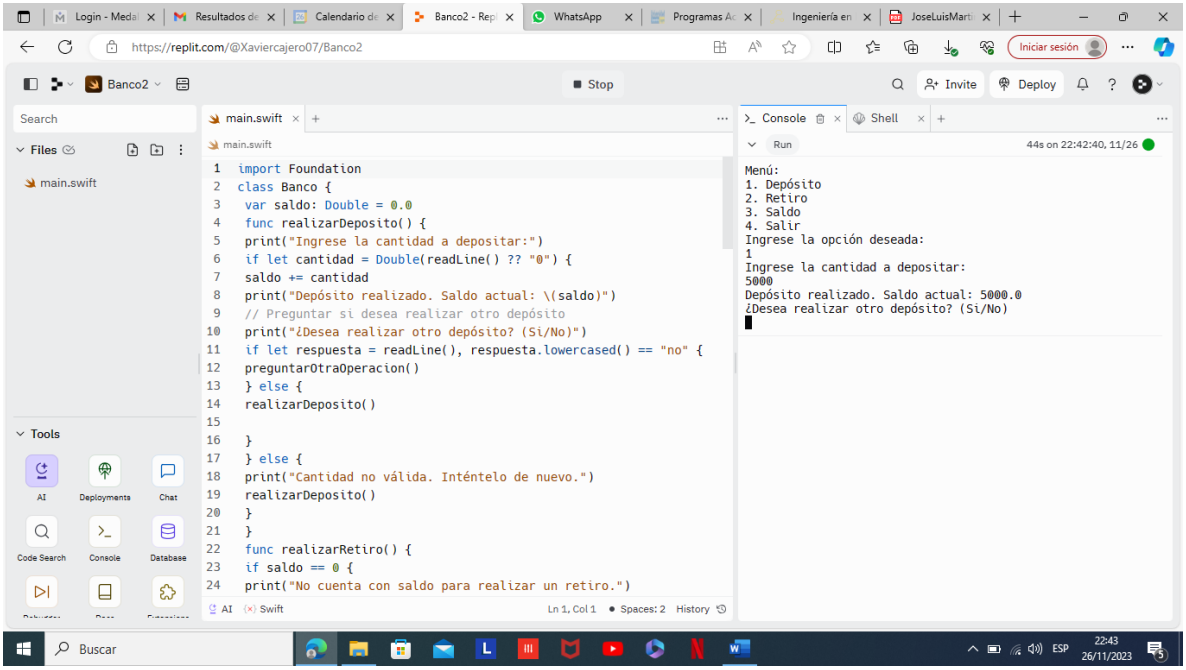
Ingeniería
Desarrollo de Software

Prueba del programa

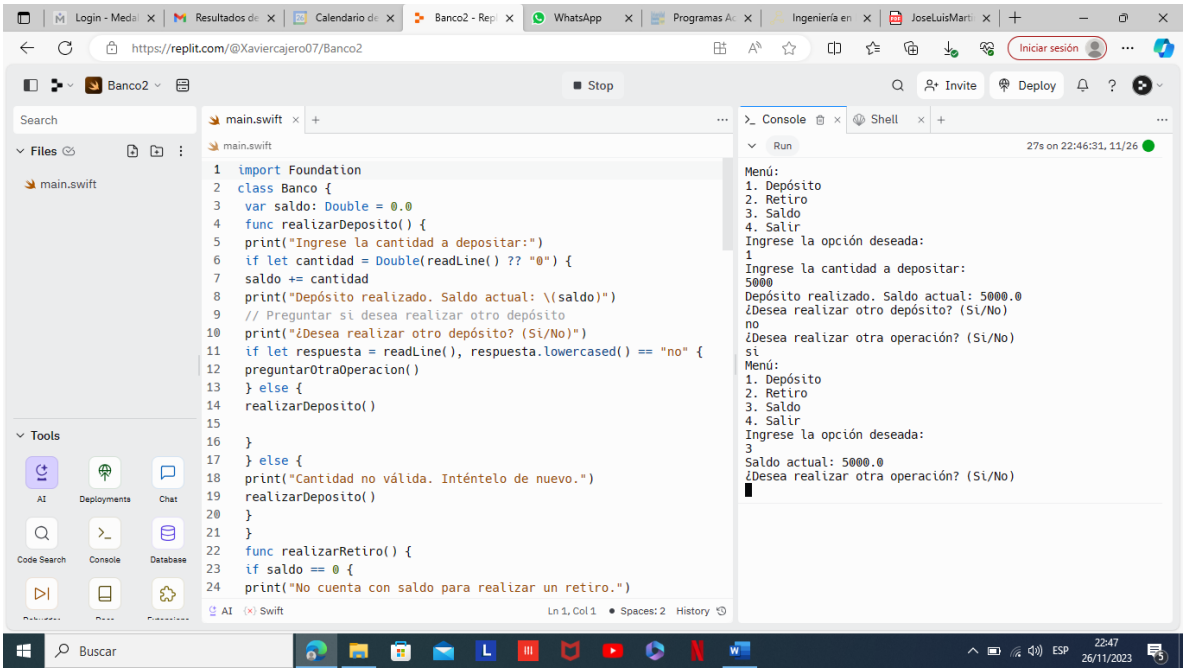
Tenemos el diseño de la interfaz



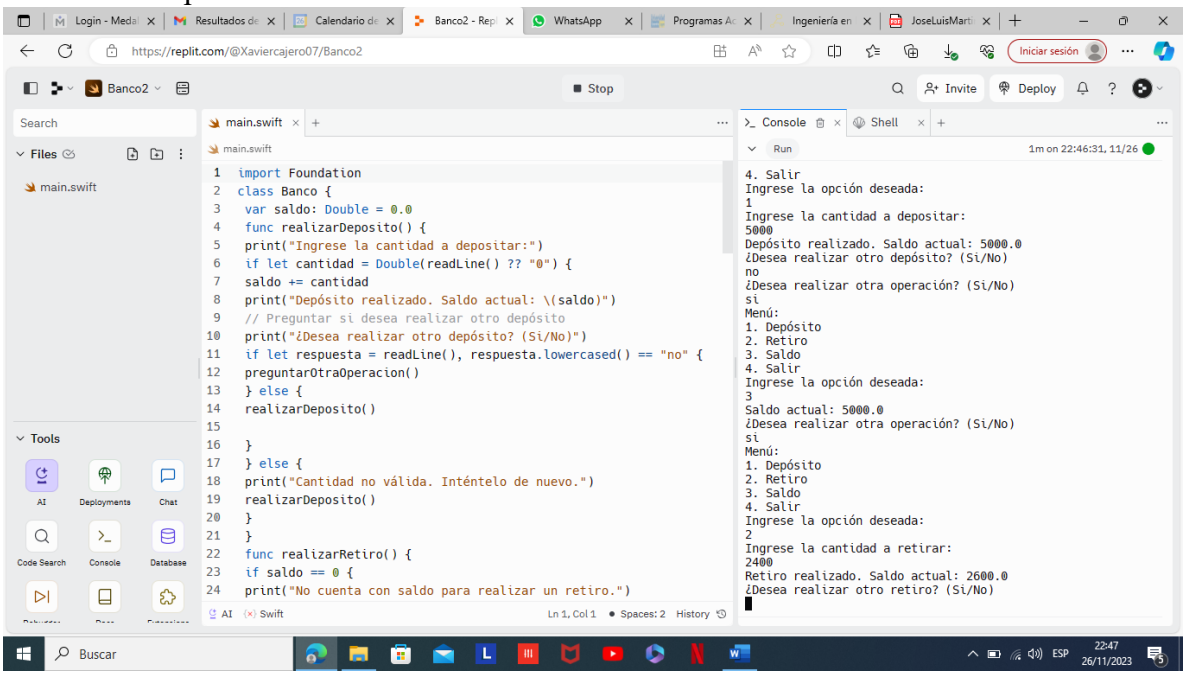
Solicitamos realizar un deposito



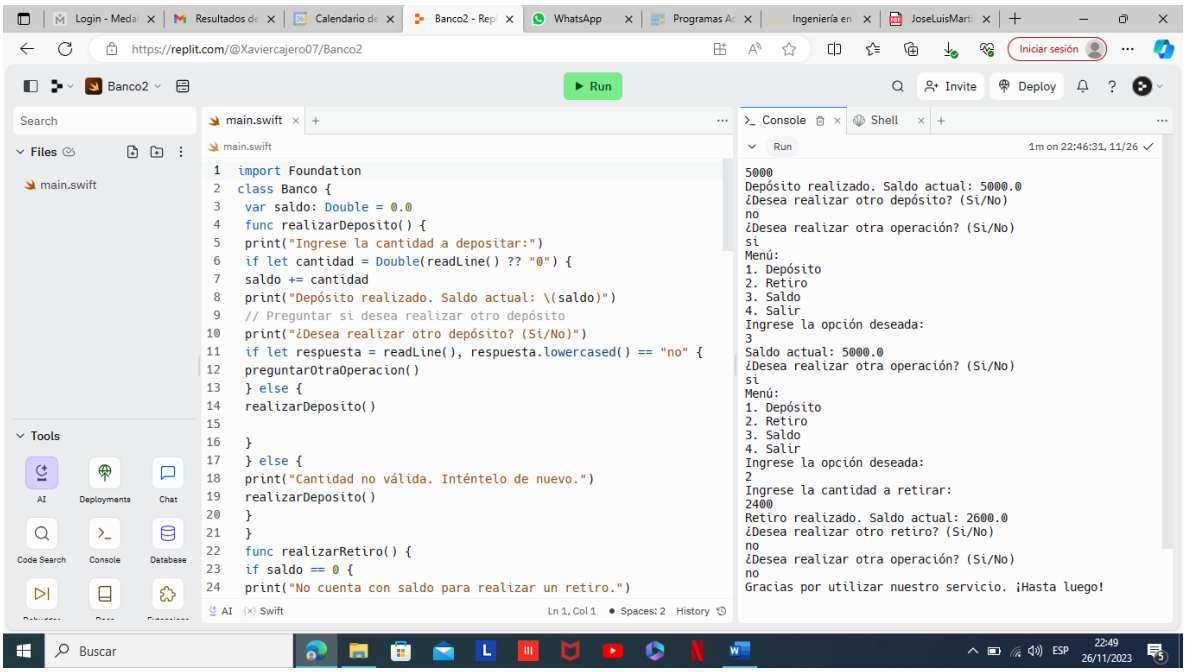
Solicitamos realizar la consulta de saldo



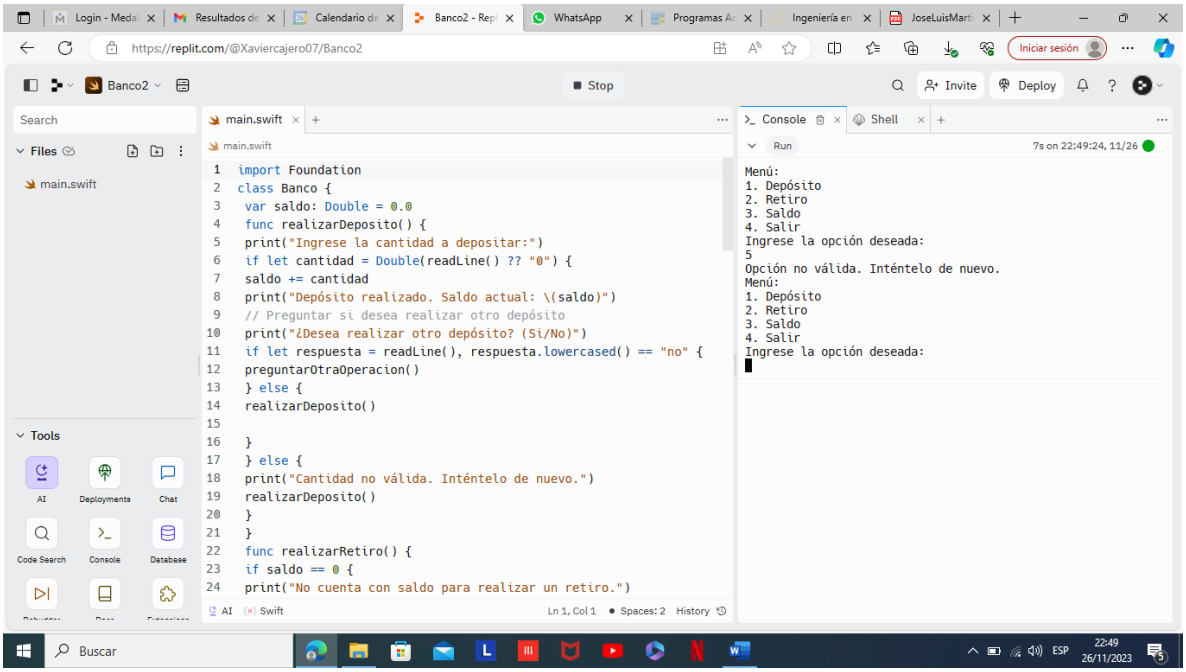
Le damos la opción de realizar retiro



Le damos opción de que ya no realizaremos ninguna operación



Si anotamos una opción no valida



Conclusión

Como conclusión de esta tercera actividad nuevamente utilizamos el compilador Replit y pude observar que ofrece un entorno de desarrollo integrado en la nube que elimina la necesidad de configurar entornos locales. Esto facilita la colaboración y el acceso a nuestros proyectos desde cualquier lugar con conexión a internet, lo cual es crucial para la flexibilidad de nuestro equipo. La capacidad de comenzar rápidamente sin preocuparse por la instalación y configuración de herramientas locales reduce las barreras para los nuevos miembros del equipo y facilita el proceso de incorporación. La función de colaboración en tiempo real de Replit permite a varios miembros del equipo trabajar simultáneamente en el mismo código. Esto mejora la eficiencia y facilita la revisión de código en tiempo real. Replit integra la escritura de código, la ejecución y la visualización de resultados en una única plataforma. Esto simplifica nuestro flujo de trabajo y permite a los desarrolladores centrarse en la lógica de la aplicación en lugar de la configuración del entorno.
Una de las ventajas de utilizar Replit es de mayor eficiencia en el desarrollo, facilidad de colaboración, simplicidad en el flujo de trabajo y flexibilidad para proyectos en el lenguaje de Swift que también nos ofrece a los desarrolladores un espacio de trabajo virtual donde pueden escribir, probar y ejecutar código directamente desde el navegador, sin necesidad de instalar software adicional en sus dispositivos locales.

Trabajo en Replit

https://replit.com/@Xaviercajero07/Banco2?v=1

Banco2.zip

Link en GitHub

https://github.com/Xaviercajero7/bancoLP4.git

Referencias

Swift - Apple (MX). (s/f). Apple. Recuperado el 23 de noviembre de 2023, de https://www.apple.com/mx/swift/

Replit. (2023, agosto 23). Mister Contenidos. https://mistercontenidos.com/directorio/replit/

ChatGPT. (s. f.). https://openai.com/chatgpt