

Actividad 3 - Programa 2 (parte 2)

Desarrollo de aplicaciones móviles IV

Ingeniería en Desarrollo de Software

Tutor: Marco Alonso Rodríguez Tapia

Alumno: Fernando Pedraza Garate

Fecha: 07 de diciembre del 2024

Índice

Etapa 1 – Instalación de Xcode / Compilador Online.

- Introducción. Pág. 3 - 4
- Descripción Pág. 5 - 6
- Justificación Pág. 7 - 8
- Desarrollo Pág. 9 - 15
 - Codificación
 - Prueba del programa

Etapa 2 – Banco Mexicano / Deposito y retiro.

- Desarrollo Pág. 16 - 21
 - Codificación
 - Prueba del programa

Etapa 3 – Banco Mexicano / Saldo y salir.

- Desarrollo Pág. 22 - 25
 - Codificación
 - Prueba del programa
- Conclusión Pág. 26
- Referencias bibliográficas. Pág. 27

Introducción

Xcode es el entorno de desarrollo integrado (IDE) oficial de Apple, una herramienta poderosa que incluye todo lo necesario para programar, diseñar, probar y depurar aplicaciones para sus distintas plataformas, como **iOS**, **macOS**, **watchOS** y **tvOS**, algunas de sus características es que es un editor de código que admite lenguajes como Swift, Objective-C y C++, ofreciendo funciones como autocompletado de código, resaltado de sintaxis y navegación entre archivos, es un diseñador de interfaces gráficas (Interface Builder) que Permite crear interfaces de usuario para aplicaciones mediante un enfoque visual, con soporte para el diseño adaptativo, con simuladores para poder probar aplicaciones en diferentes dispositivos de Apple sin necesidad de usar un hardware físico, con herramientas avanzadas para poder identificar errores en el código y monitorear el rendimiento, facilitando la organización y configuración de proyectos, con opciones para gestionar dependencias y configuraciones de compilación, e incluye herramientas como XCTest para escribir pruebas automatizadas e Instruments para analizar el rendimiento y funciona únicamente en computadoras con macOS.

Si se cuenta con equipo con sistema operativo Windows, se deberá utilizar un compilador online, herramienta basada en la web que permite escribir, compilar y ejecutar código directamente en un navegador web, sin necesidad de instalar un entorno de desarrollo o un compilador en el equipo a utilizar.

Algunos ejemplos de compiladores online populares son:

- **Replit** (replit.com): Soporta múltiples lenguajes y proyectos colaborativos.
- **JDoodle** (jdoodle.com): Ligero y fácil de usar para varios lenguajes.
- **Compiler Explorer** (godbolt.org): Excelente para explorar el funcionamiento interno de compiladores como GCC y Clang.
- **Programiz** (programiz.com): Focalizado en enseñar programación con soporte para Python, C, y más.

Descripción.

La boutique Norma requiere desarrollar una aplicación que funcione como una tienda de ropa en línea, donde los clientes puedan visualizar productos y realizar compras directamente desde la app, este proyecto debe implementarse utilizando el lenguaje de programación Swift.

Se deberá instalar Xcode como entorno de desarrollo o utilizar un compilador en línea compatible con Swift y programar la aplicación siguiendo las especificaciones indicadas, para los requisitos del programa que deberá poder mostrar 4 artículos disponibles, incluyendo su nombre, precio y cantidad en stock e incluir un menú interactivo para que el cliente pueda:

- Seleccionar un artículo para comprar.
- Salir de la aplicación.

Y al finalizar una compra, el programa debe mostrar el nombre del artículo adquirido y el monto total a pagar, con el objetivo de crear una experiencia sencilla y funcional que simule una tienda en línea básica para mejorar la interacción cliente-producto y facilitar la compra directa desde la app.

Posteriormente se requiere crear un programa en lenguaje Swift para la banca en línea del Banco Mexicano, en la que se podrán realizar diversas acciones, como deposito, retiro, consultar saldo o salir, por medio de su menú principal, en caso de que el usuario ingrese la opción

Depósito, el programa deberá ser capaz de capturar por teclado los siguientes datos:

- Cantidad a depositar
- Preguntar si desea realizar otro depósito; si la respuesta es “No”, deberá preguntar si desea realizar otra operación.

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.

Justificación.

El usar Xcode o un compilador en línea depende de las necesidades y el contexto del proyecto, en el caso de Xcode por ser un entorno integrado profesional y ser el entorno oficial de Apple para desarrollar aplicaciones en Swift, que ofrece herramientas avanzadas como un simulador de dispositivos, autocompletado, depuración y diseño de interfaces gráficas, permitiendo desarrollar, probar y optimizar aplicaciones para iOS, iPadOS, macOS y más, asegurando un rendimiento óptimo en los dispositivos Apple, ayudando a los programadores a familiarizarse con herramientas profesionales utilizadas en la industria, lo que mejora sus habilidades prácticas.

Y el uso de un compilador en línea ofrece acceso inmediato, al no requerir instalación ni configuraciones avanzadas, ideal para usuarios que no disponen de macOS o equipos potentes, es útil para proyectos pequeños o educativos, enfocándose en la lógica y el código sin distracciones de herramientas avanzadas, ideal para aprender Swift sin necesidad de adquirir un dispositivo Apple.

Ambas opciones son válidas según los recursos disponibles y la complejidad del proyecto. Xcode es ideal para producción, mientras que un compilador en línea es más práctico para aprendizaje rápido o demostraciones.

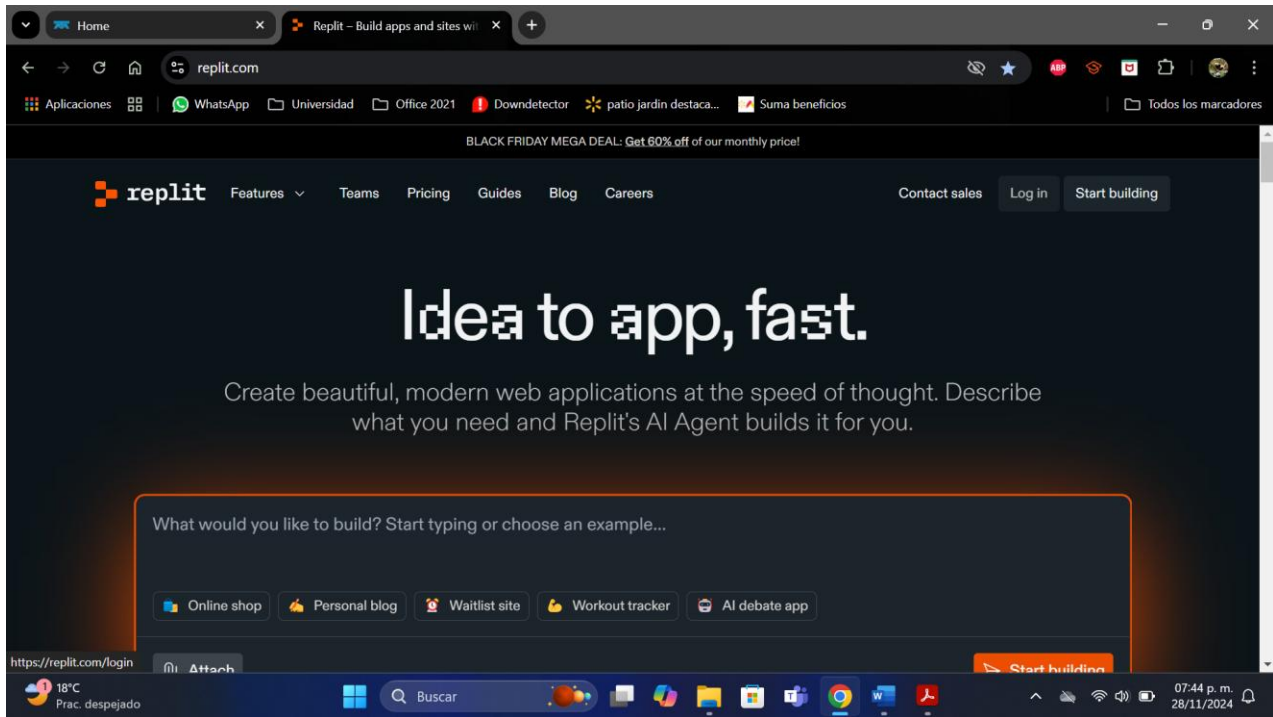
Crear una aplicación bancaria en **Swift** es ideal si se busca ofrecer una experiencia óptima en dispositivos Apple, es rápido, eficiente y seguro, con características clave para manejar datos sensibles y garantizar una experiencia de usuario fluida, este lenguaje al ser respaldado por Apple, asegura compatibilidad con futuras versiones de iOS y acceso a tecnologías avanzadas como Face ID, Touch ID y Apple Pay.

Swift también permite crear interfaces modernas con frameworks como **SwiftUI**, que mejoran la usabilidad y estética de la aplicación, además, su gestión de memoria segura y prevención de errores comunes refuerzan la seguridad, características esenciales dentro del sector bancario.

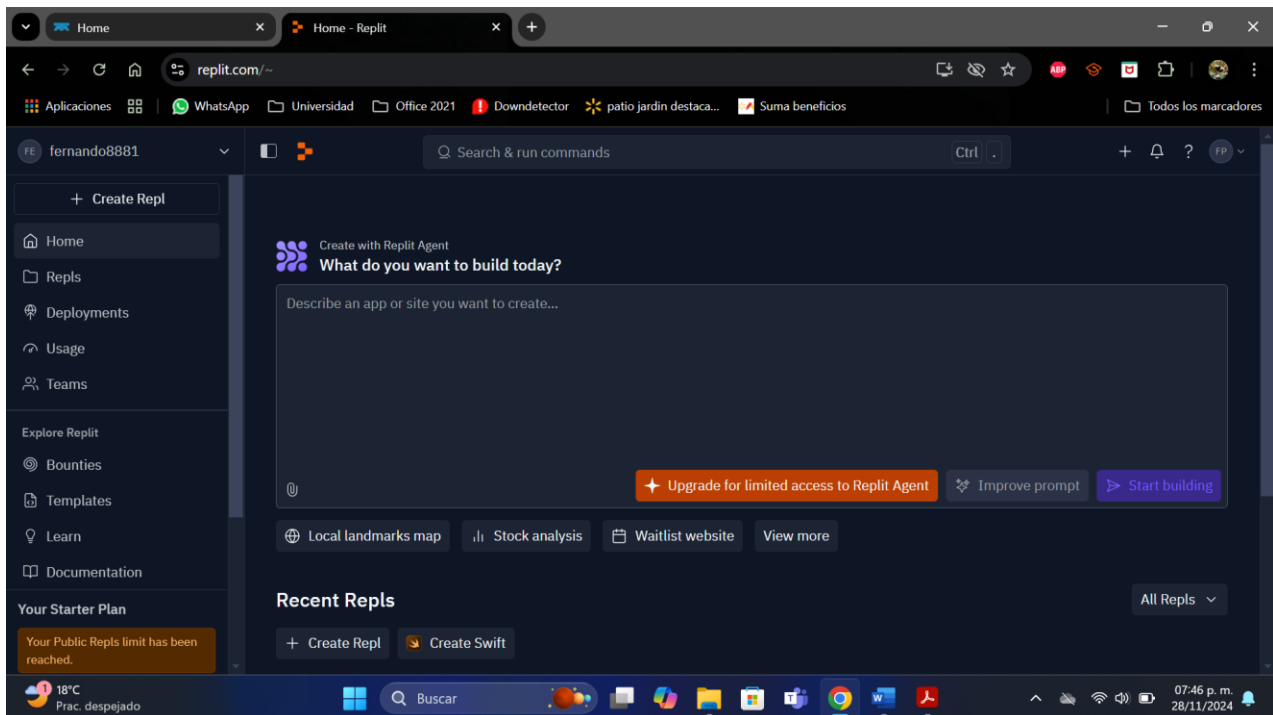
Desarrollar en Swift brinda acceso a herramientas como **Xcode**, que facilita las pruebas y análisis de rendimiento, haciendo que sea fácil de encontrar soporte técnico en la comunidad activa y los recursos de Apple, y por último, su integración con tecnologías emergentes, como **CoreML** para la detección de fraudes, les permite innovar y adaptarse a las necesidades del cliente.

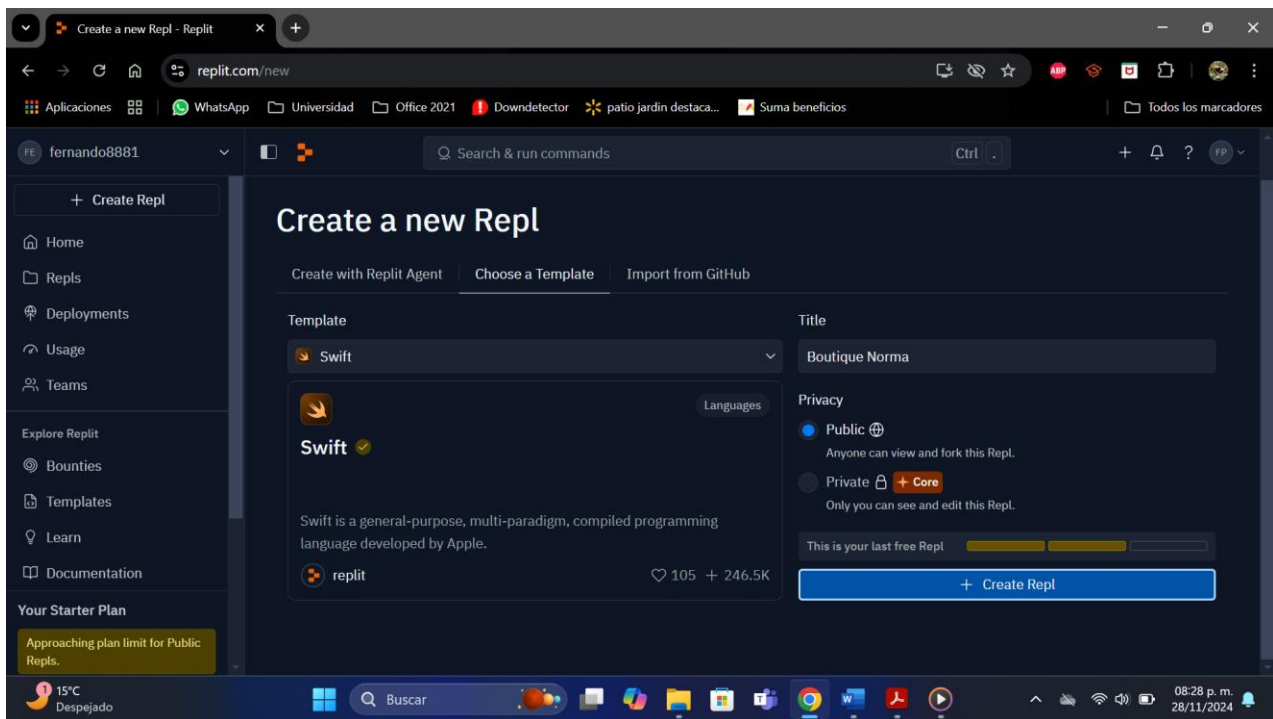
Desarrollo.

Codificación / compilador online

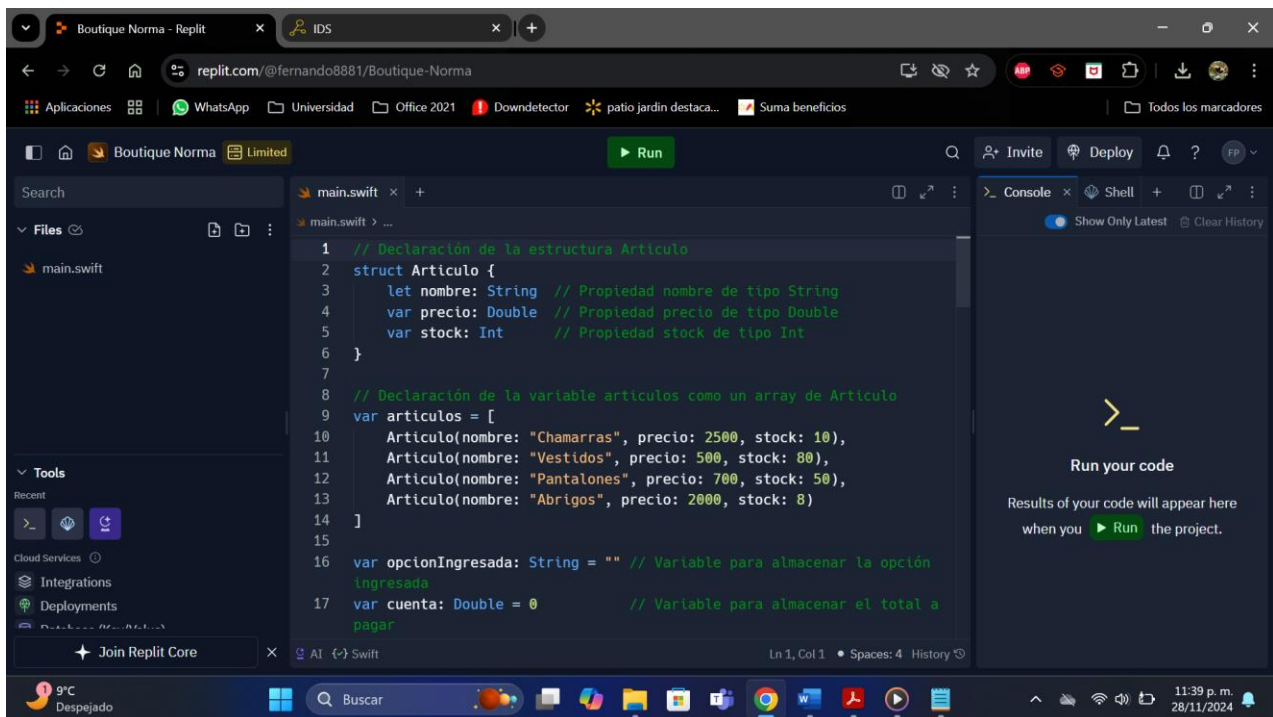


Se ingresa a la página de Replit





Se crea el nuevo Repl con el nombre de Boutique Norma



Se declara la estructura de los artículos y las variables como un arreglo

```
15
16 var opcionIngresada: String = "" // Variable para almacenar la opción
    ingresada
17 var cuenta: Double = 0           // Variable para almacenar el total a
    pagar
18
19 while opcionIngresada != "2" { // Mientras no se seleccione "Salir"
20     // Mostrar el catálogo de artículos
21     print("***** Bienvenido a su boutique Norma *****\n")
22
23     // Mostrar la lista de productos de forma dinámica
24     for (index, articulo) in articulos.enumerated() {
25         print("Artículo \(index + 1): \(articulo.nombre)")
26         print("Precio: \(articulo.precio)")
27         print("Stock: \(articulo.stock)")
28         print("-----")
29     }
30
31     // Mostrar opciones del menú
32     print("\n1.- Comprar artículo")
33     print("2.- Salir")
34     print("Seleccione una opción:")
35     opcionIngresada = readLine() ?? "" // Leer la entrada del usuario
36
37     switch opcionIngresada {
38     case "1": // Comprar artículo
39         print("Ingrese el número del artículo deseado:")
40         if let numeroArticulo = Int(readLine() ?? ""), numeroArticulo
41             > 0, numeroArticulo <= articulos.count {
42             let articuloSeleccionado = articulos[numeroArticulo - 1]
43             print("Ingrese la cantidad de \(
44                 articuloSeleccionado.nombre) que desea comprar:")
45             if let cantidadIngresada = Int(readLine() ?? ""),
46                 cantidadIngresada > 0 {
47                 if cantidadIngresada <= articuloSeleccionado.stock {
48                     let total = Double(cantidadIngresada) *
49                         articuloSeleccionado.precio
50                     cuenta += total
51                 }
52             }
53         }
54     }
55 }
```

Se crean las variables para los datos ingresados por el usuario y el ciclo while para continuar dentro del programa mientras no se seleccione la opción salir, mostrando las opciones del menú.

```
31 // Mostrar opciones del menú
32 print("\n1.- Comprar artículo")
33 print("2.- Salir")
34 print("Seleccione una opción:")
35 opcionIngresada = readLine() ?? "" // Leer la entrada del usuario
36
37 switch opcionIngresada {
38 case "1": // Comprar artículo
39     print("Ingrese el número del artículo deseado:")
40     if let numeroArticulo = Int(readLine() ?? ""), numeroArticulo
41         > 0, numeroArticulo <= articulos.count {
42         let articuloSeleccionado = articulos[numeroArticulo - 1]
43         print("Ingrese la cantidad de \(
44             articuloSeleccionado.nombre) que desea comprar:")
45         if let cantidadIngresada = Int(readLine() ?? ""),
46             cantidadIngresada > 0 {
47             if cantidadIngresada <= articuloSeleccionado.stock {
48                 let total = Double(cantidadIngresada) *
49                     articuloSeleccionado.precio
50                 cuenta += total
51             }
52         }
53     }
54 case "2": // Salir
55     print("¡Gracias por su compra!")
56 }
```

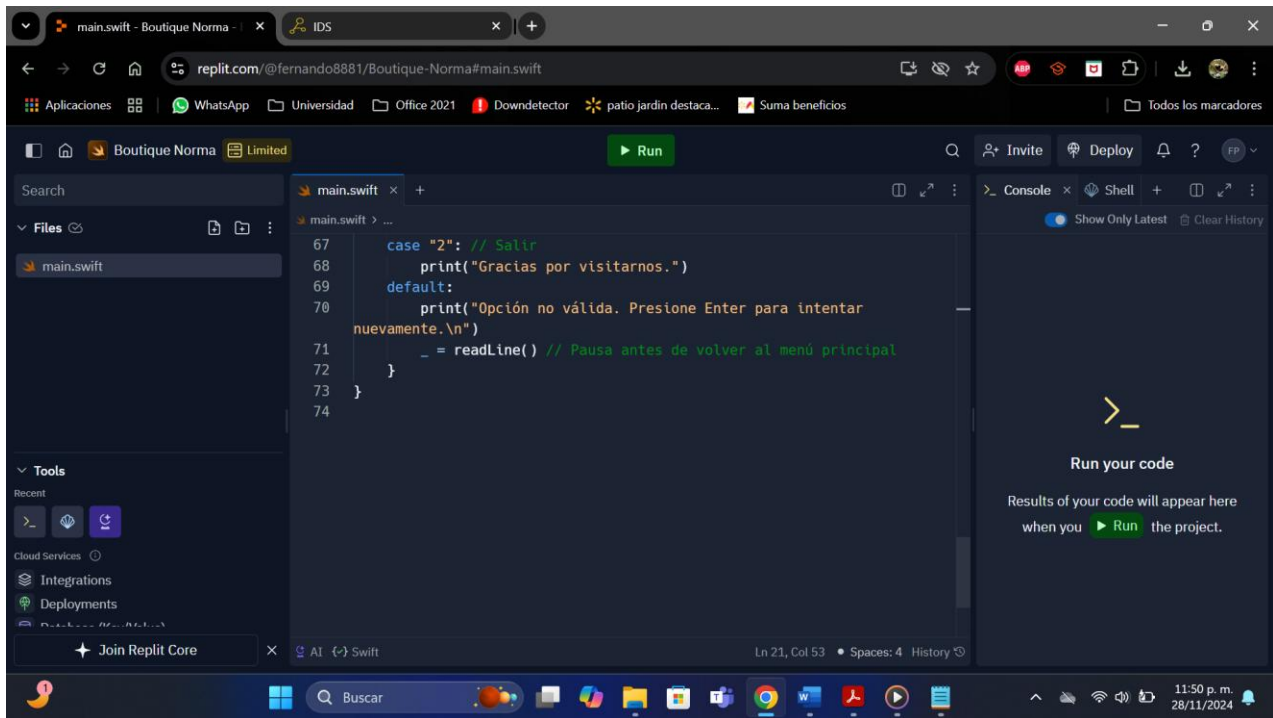
Creando el switch para que el programa ejecute una acción dependiendo el caso establecido.

```
42
43     print("Ingrese la cantidad de \
(articuloSeleccionado.nombre) que desea comprar:")
44     if let cantidadIngresada = Int(readLine() ?? ""),
cantidadIngresada > 0 {
45         if cantidadIngresada <= articuloSeleccionado.stock {
46             let total = Double(cantidadIngresada) *
articuloSeleccionado.precio
47             cuenta += total
48             articulos[numeroArticulo - 1].stock -=
cantidadIngresada
49             print("Compra realizada exitosamente.")
50             print("Total a pagar por esta compra: $\((total)")
51             print("Gracias por su compra. Le esperamos
nuevamente.\n")
52             print("Presione Enter para regresar al menú
principal.")
53             _ = readLine() // Pausa antes de volver al menú
54             principal
55         } else {
56             print("Lo sentimos, no hay suficiente stock para
su pedido.\n")
57             print("Presione Enter para regresar al menú
principal.")
58             _ = readLine() // Pausa antes de volver al menú
59             principal
60         } else {
61             print("Cantidad no válida. Presione Enter para
intentar nuevamente.\n")
62             _ = readLine() // Pausa antes de volver al menú
63             principal
64         } else {
65             print("Artículo no válido. Presione Enter para intentar
nuevamente.\n")
66             _ = readLine() // Pausa antes de volver al menú principal
67             principal
68         }
```

Imprimiendo en pantalla el contexto establecido según la actividad presentada, haciendo pausas antes de mostrar el menú principal.

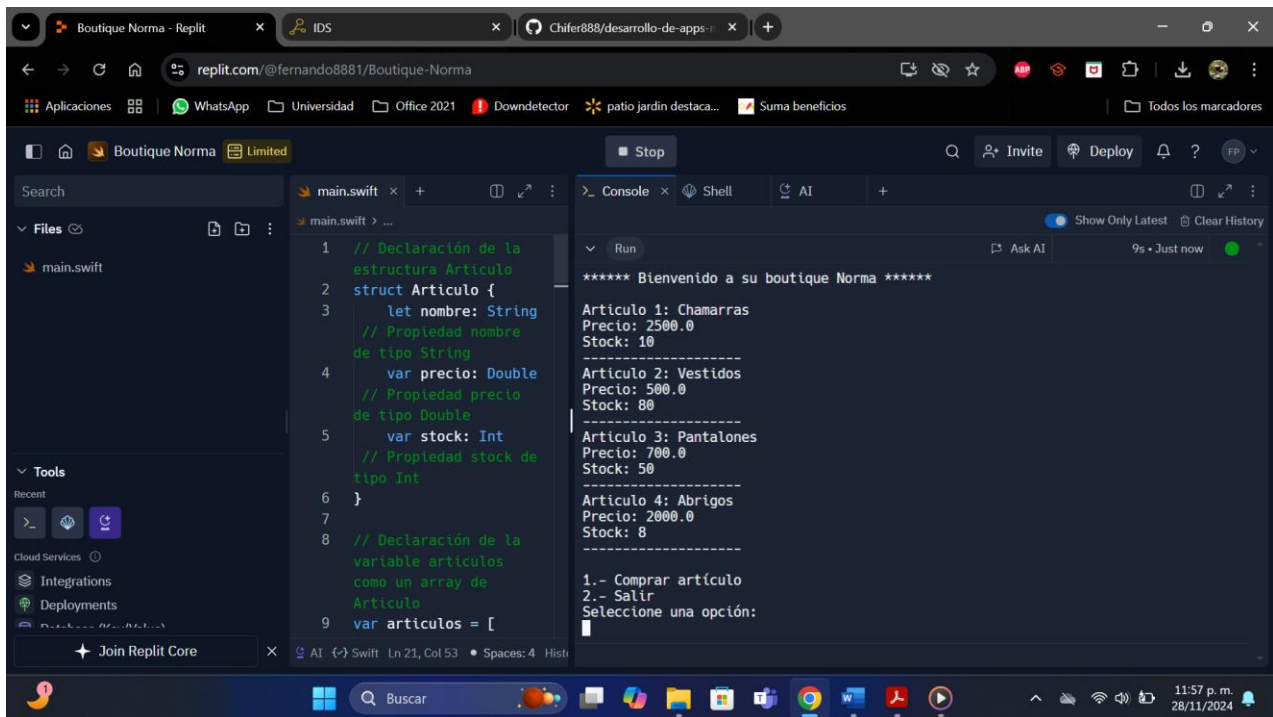
```
54         } else {
55             print("Lo sentimos, no hay suficiente stock para
su pedido.\n")
56             print("Presione Enter para regresar al menú
principal.")
57             _ = readLine() // Pausa antes de volver al menú
58             principal
59         } else {
60             print("Cantidad no válida. Presione Enter para
intentar nuevamente.\n")
61             _ = readLine() // Pausa antes de volver al menú
62             principal
63         } else {
64             print("Artículo no válido. Presione Enter para intentar
nuevamente.\n")
65             _ = readLine() // Pausa antes de volver al menú principal
66             principal
67         }
```

Creando un else para que se muestran advertencias en caso de ingresar algún dato erróneo.



Y para que el programa se detenga al seleccionar la opción salir.

Prueba del programa



Programa iniciado

```
1 // Declaración de la
2 struct Artículo {
3     let nombre: String
4     // Propiedad nombre
5     de tipo String
6     var precio: Double
7     // Propiedad precio
8     de tipo Double
9     var stock: Int
10    // Propiedad stock de
11    tipo Int
12 }
13 // Declaración de la
14 variable articulos
15 como un array de
16 Artículo
17 var articulos = [
```

***** Bienvenido a su boutique Norma *****

Artículo 1: Chamarras
Precio: 2500.0
Stock: 10

Artículo 2: Vestidos
Precio: 500.0
Stock: 80

Artículo 3: Pantalones
Precio: 700.0
Stock: 50

Artículo 4: Abrigos
Precio: 2000.0
Stock: 8

1.- Comprar artículo
2.- Salir
Seleccione una opción:
5
Opción no válida. Presione Enter para intentar nuevamente.

Al ingresar erróneamente una opción manda la observación y solicita dar enter para continuar.

```
1 // Declaración de la
2 struct Artículo {
3     let nombre: String
4     // Propiedad nombre
5     de tipo String
6     var precio: Double
7     // Propiedad precio
8     de tipo Double
9     var stock: Int
10    // Propiedad stock de
11    tipo Int
12 }
13 // Declaración de la
14 variable articulos
15 como un array de
16 Artículo
17 var articulos = [
```

Stock: 80

Artículo 3: Pantalones
Precio: 700.0
Stock: 50

Artículo 4: Abrigos
Precio: 2000.0
Stock: 8

1.- Comprar artículo
2.- Salir
Seleccione una opción:
1
Ingrese el número del artículo deseado:
4
Ingrese la cantidad de Abrigos que desea comprar:
5
Compra realizada exitosamente.
Total a pagar por esta compra: \$10000.0
Gracias por su compra. Le esperamos nuevamente.
Presione Enter para regresar al menú principal.

Selección correcta del menú, del articulo y de la cantidad a comprar, mostrando el monto total a pagar y solicitando dar enter para regresar al menú principal.

```
1 // Declaración de la
2 struct Articulo {
3     let nombre: String
4     // Propiedad nombre
5     de tipo String
6     var precio: Double
7     // Propiedad precio
8     de tipo Double
9     var stock: Int
10    // Propiedad stock de
11    tipo Int
12 }
13 // Declaración de la
14 variable articulos
15 como un array de
16 Articulo
17 var articulos = [
```

Run

Artículo 2: Vestidos
Precio: 500.0
Stock: 80

Artículo 3: Pantalones
Precio: 700.0
Stock: 50

Artículo 4: Abrigos
Precio: 2000.0
Stock: 3

1.- Comprar artículo
2.- Salir
Seleccione una opción:
1
Ingrese el número del artículo deseado:
3
Ingrese la cantidad de Pantalones que desea comprar:
900
Lo sentimos, no hay suficiente stock para su pedido.
Presione Enter para regresar al menú principal.

Haciendo lo mismo pero con un stock elevado para que el programa notifique la falta de stock.

```
1 // Declaración de la
2 struct Articulo {
3     let nombre: String
4     // Propiedad nombre
5     de tipo String
6     var precio: Double
7     // Propiedad precio
8     de tipo Double
9     var stock: Int
10    // Propiedad stock de
11    tipo Int
12 }
13 // Declaración de la
14 variable articulos
15 como un array de
16 Articulo
17 var articulos = [
```

Run

***** Bienvenido a su boutique Norma *****

Artículo 1: Chamarras
Precio: 2500.0
Stock: 10

Artículo 2: Vestidos
Precio: 500.0
Stock: 80

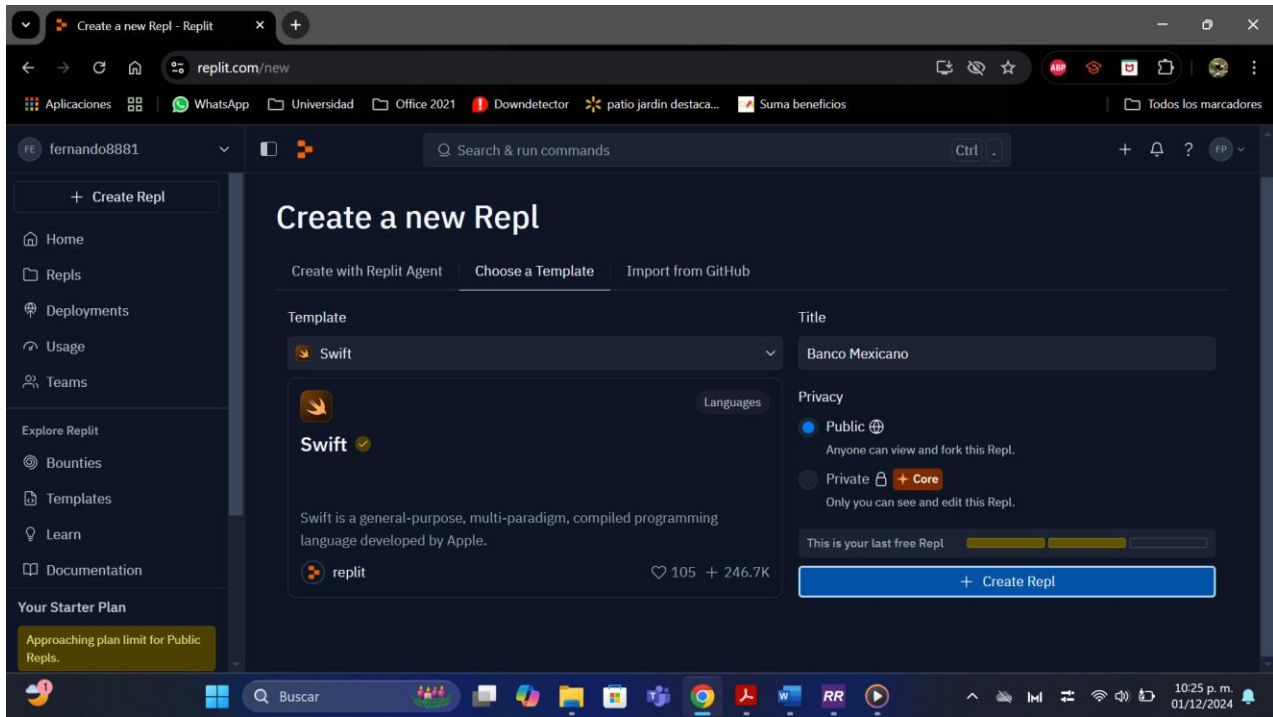
Artículo 3: Pantalones
Precio: 700.0
Stock: 50

Artículo 4: Abrigos
Precio: 2000.0
Stock: 3

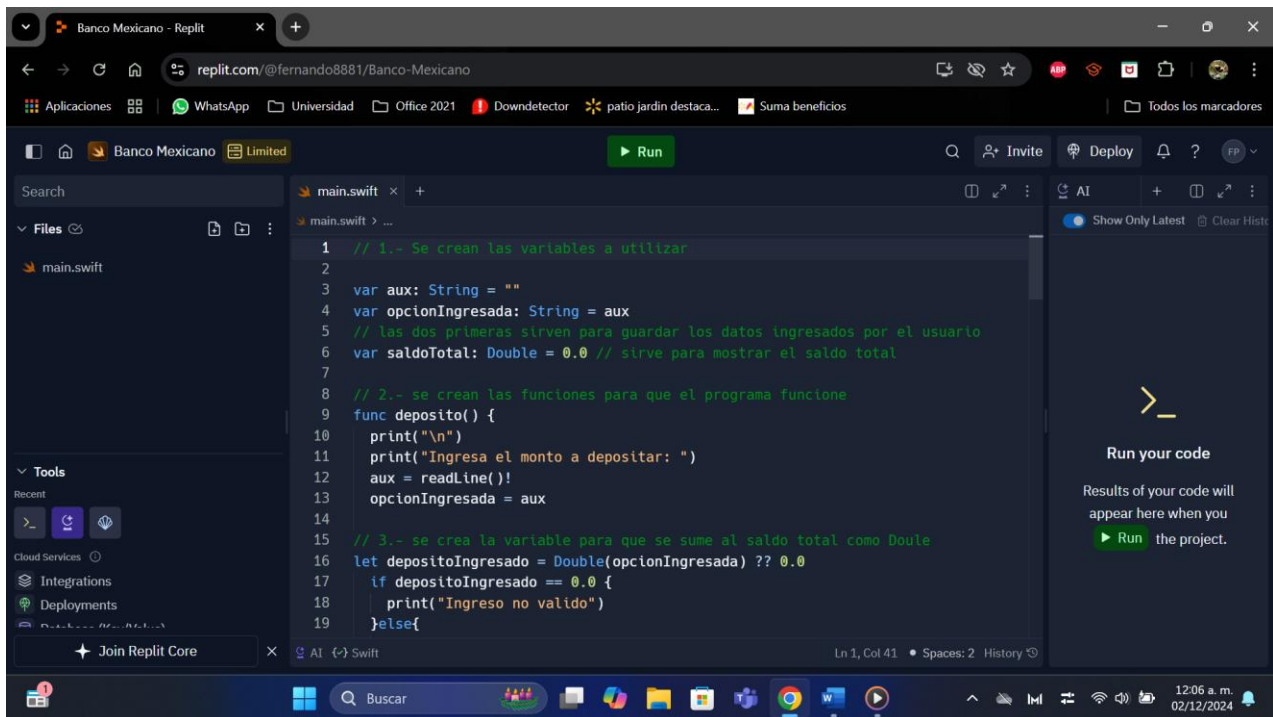
1.- Comprar artículo
2.- Salir
Seleccione una opción:
2
Gracias por visitarnos.

Y por ultimo se sale del programa y se detiene el programa agradeciendo la visita.

Etapa 2.- Banco Mexicano / Deposito y retiro.



Se ingresa a Replit y se crea el nuevo Repl con el nombre de Banco Mexicano



Empezamos por declarar las variables a utilizar, creando las funciones para que funcione el programa y la variable para que se sume el saldo total.

The screenshot shows a Replit IDE window titled "main.swift - Banco Mexicano". The code in the editor is as follows:

```
26 // función para el retiro de efectivo
27 func retiro() {
28     print("\n")
29     print("Ingresa el monto a retirar: ")
30     aux = readLine()!
31     opcionIngresada = aux
32
33     // Convertir el monto ingresado a Double
34     let retiroIngresado = Double(opcionIngresada) ?? 0.0
35
36     if retiroIngresado == 0.0 {
37         print("Ingreso no válido")
38     } else if retiroIngresado > saldoTotal {
39         print("No cuenta con suficiente saldo para realizar este retiro.")
40     } else {
41         saldoTotal -= retiroIngresado
42         print("Retiro exitoso de $ \(retiroIngresado)")
43     }
44     print("\n")
45 }
```

The interface includes a file explorer on the left showing "main.swift", a search bar, and a "Run" button at the top right. The status bar at the bottom indicates "Ln 95, Col 11" and "Spaces: 2".

Se crea la función para el retiro de efectivo

The screenshot shows a Replit IDE window titled "Banco Mexicano - Replit". The code in the editor is as follows:

```
20     print("Abono realizado con exito por $ \(depositoIngresado)")
21 }
22 saldoTotal = saldoTotal + depositoIngresado
23 print("\n")
24 }
25
26 // 4.- Se crea el menú interactivo que se mostrara al inicio utilizando la
    sentencia while
27 while opcionIngresada != "4" {
28     print("*** Bienvenido a su Banco Mexicano**")
29     print("\n")
30     print("Elija una opción")
31     print("\n")
32     print("1.- Deposito")
33     print("2.- Retiro")
34     print("3.- Saldo")
35     print("4.- Salir")
36
37     aux = readLine()!
38 }
```

The interface includes a file explorer on the left showing "main.swift", a search bar, and a "Run" button at the top right. The status bar at the bottom indicates "Ln 1, Col 41" and "Spaces: 2".

Se crea el menú interactivo

```
38 opcionIngresada = aux
39
40 // 5.- Se valida la opción ingresada por el usuario utilizando la sentencia switch
41 switch opcionIngresada {
42     case "1":
43         deposito()
44         print("\n")
45         print("Desea realizar otro deposito? (S/N): ")
46         aux = readLine()!
47         opcionIngresada = aux
48
49 // Se utiliza el metodo if para validar si es afirmativa la respuesta del
50 usuario
51 if opcionIngresada == "S" || opcionIngresada == "s" ||
52    opcionIngresada == "Si" || opcionIngresada == "si" ||
53    opcionIngresada == "SI" {
54        deposito()
55    }
```

Se valida la opción ingresada por el usuario utilizando la sentencia switch

```
54 // Se utiliza el metodo if para validar si es negativa la respuesta del
55 usuario
56 if opcionIngresada == "N" || opcionIngresada == "n" ||
57    opcionIngresada == "No" || opcionIngresada == "no" ||
58    opcionIngresada == "NO" {
59     print("\n")
60     print("¿Desea realizar alguna otra operación? (S/N): ")
61     aux = readLine()!
62     opcionIngresada = aux
63
64 if opcionIngresada == "N" || opcionIngresada == "n" ||
65    opcionIngresada == "No" || opcionIngresada == "no" ||
66    opcionIngresada == "NO" {
67     print("\n")
68     print("Sesión terminada, \n gracias por utilizar
69 nuestra app.")
70     opcionIngresada = "4"
71 }
```

Se valida si la respuesta ingresada por el usuario es afirmativa o negativa

```
84
85     if opcionIngresada == "N" || opcionIngresada == "n" || opcionIngresada == "No" ||
opcionIngresada == "no" || opcionIngresada == "NO" {
86         print("\n")
87         print("¿Desea realizar alguna otra operación? (S/N): ")
88         aux = readLine()!
89         opcionIngresada = aux
90
91     if opcionIngresada == "N" || opcionIngresada == "n" || opcionIngresada == "No" ||
opcionIngresada == "no" || opcionIngresada == "NO">{
92         print("\n")
93         print("Sesión terminada, \n gracias por utilizar nuestra app.")
94         opcionIngresada = "4"
95     }
96 }
97
98 case "2": // Se crea la opción 2 para retirar dinero
99     retiro() // Se llama a la función retiro
100
```

Se establece la opción del case 2 para el retiro de efectivo.

Prueba del programa

```
1 // 1.- Se crean las variables a utilizar
2
3 var aux: String = ""
4 var opcionIngresada: String = aux
5 // las dos primeras sirven para guardar los datos ingresados por el usuario
6 var saldoTotal: Double = 0.0 // sirve para mostrar el saldo total
7
8 // 2.- se crean las funciones para que el programa funcione
9 func deposito() {
10     print("\n")
11     print("Ingresa el monto a depositar: ")
12     aux = readLine()!
13     opcionIngresada = aux
```

Menú principal

```
1 // 1.- Se crean las variables a utilizar
2
3 var aux: String = ""
4 var opcionIngresada: String = aux
5 // las dos primeras sirven para guardar los datos ingresados por el usuario
6 var saldoTotal: Double = 0.0 // sirve para mostrar el saldo total
7
8 // 2.- se crean las funciones para que el programa funcione
9 func deposito() {
10     print("\n")
11     print("Ingresa el monto a depositar: ")
12     aux = readLine()!
13     opcionIngresada = aux
14 }
```

Elige una opción

- 1.- Deposito
- 2.- Retiro
- 3.- Saldo
- 4.- Salir

Ingresa el monto a depositar:

100

Abono realizado con éxito por \$ 100.0

Desea realizar otro deposito? (S/N):

s

Ingresa el monto a depositar:

200

Depósitos ingresados de forma correcta

```
86 opcionIngresada == "NO" || opcionIngresada == "no" ||
87 opcionIngresada == "N0" {
88     print("\n")
89     print("¿Desea realizar alguna otra operación? (S/N): ")
90     aux = readLine()!
91     opcionIngresada = aux
92
93     if opcionIngresada == "N" || opcionIngresada == "n" ||
94     opcionIngresada == "No" || opcionIngresada == "no" ||
95     opcionIngresada == "N0" {
96         print("\n")
97         print("Sesión terminada, \n gracias por utilizar
98         nuestra app.")
99         opcionIngresada = "4"
100     }
101 }
```

Elige una opción

- 1.- Deposito
- 2.- Retiro
- 3.- Saldo
- 4.- Salir

Ingresa el monto a retirar:

200

Retiro exitoso de \$ 200.0

** Bienvenido a su Banco Mexicano **

Elige una opción

Prueba de retiro, funcionando de forma correcta


```
1 // 1.- Se crean las variables a utilizar
2
3 var aux: String = ""
4 var opcionIngresada: String = aux
5 // las dos primeras sirven para guardar los datos ingresados por el usuario
6 var saldoTotal: Double = 0.0 // sirve para mostrar el saldo total
7
8 // 2.- se crean las funciones para que el programa funcione
9 func deposito() {
10     print("\n")
11     print("Ingresa el monto a depositar: ")
12     aux = readLine()!
13     opcionIngresada = aux
```

Run

** Bienvenido a su Banco Mexicano**

Elija una opción

1.- Deposito
2.- Retiro
3.- Saldo
4.- Salir
fern

opcion no valida
** Bienvenido a su Banco Mexicano**

Elija una opción

1.- Deposito
2.- Retiro
3.- Saldo
4.- Salir

Mensaje de error al ingresar datos erróneos

```
1 // 1.- Se crean las variables a utilizar
2
3 var aux: String = ""
4 var opcionIngresada: String = aux
5 // las dos primeras sirven para guardar los datos ingresados por el usuario
6 var saldoTotal: Double = 0.0 // sirve para mostrar el saldo total
7
8 // 2.- se crean las funciones para que el programa funcione
9 func deposito() {
10     print("\n")
11     print("Ingresa el monto a depositar: ")
12     aux = readLine()!
13     opcionIngresada = aux
```

Run

1.- Deposito
2.- Retiro
3.- Saldo
4.- Salir
fern

opcion no valida
** Bienvenido a su Banco Mexicano**

Elija una opción

1.- Deposito
2.- Retiro
3.- Saldo
4.- Salir
4

Sesión terminada,
gracias por utilizar nuestra app.

Salida del programa

Etapa 3.- Banco Mexicano / Saldo y salir.

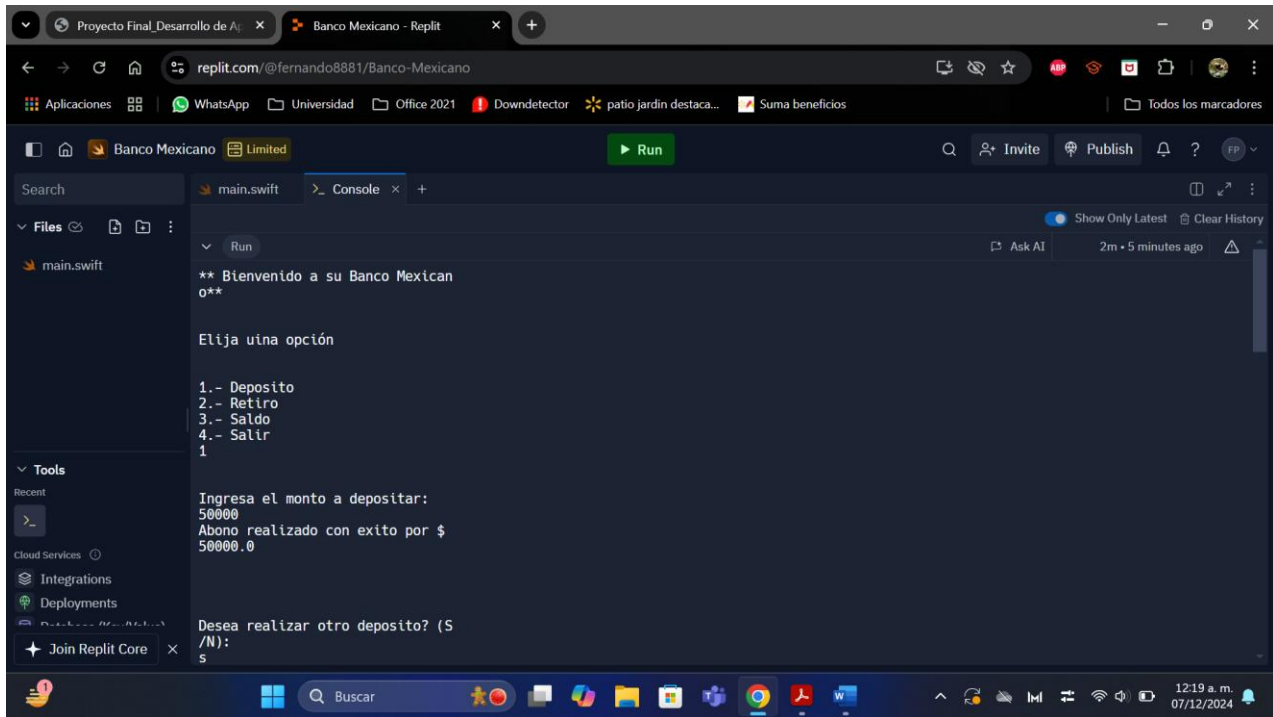
```
47 // función para la consulta de saldo
48 func varSaldo() {
49     print("\n")
50     print("Tu saldo actual es de: $ \(saldoTotal)")
51     print("\n")
52 }
53
54
55 // 4.- Se crea el menú interactivo que se mostrara al inicio utilizando la sentencia while
56 while opcionIngresada != "4" {
57     print("*** Bienvenido a su Banco Mexicano**")
58     print("\n")
59     print("Elija una opción")
60     print("\n")
61     print("1.- Deposito")
62     print("2.- Retiro")
63     print("3.- Saldo")
64     print("4.- Salir")
65 }
```

Se crea la función para la consulta de saldo y la opción salir

```
100
101 case "3": // Se crea la opción 3 para mostrar el saldo
102     varSaldo() // Se llama a la función varSaldo
103
104 // Siempre que se utilice switch se debe utilizar la opción default para que el programa no se caiga
105 default:
106
107     if opcionIngresada != "4"{
108         print("\n")
109         print("opcion no valida")
110     } else {
111         print("\n")
112         print("Sesión terminada, \ngracias por utilizar nuestra app.")
113     }
114 }
115 }
```

Y se establece la variable en el case 3 para llamar a la función

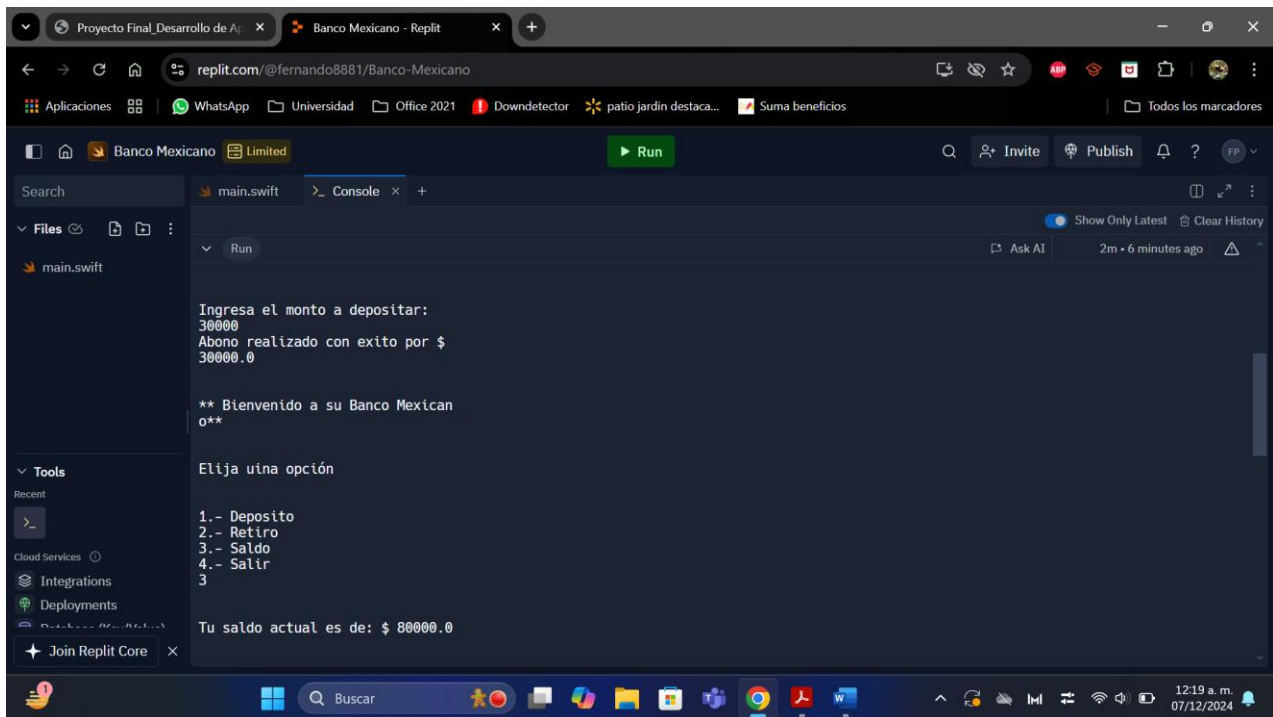
Prueba del programa



The screenshot shows a Replit terminal window for a project named "Banco Mexicano". The terminal output displays a menu with four options: 1.- Deposito, 2.- Retiro, 3.- Saldo, and 4.- Salir. Option 1 is selected. The user is prompted to enter the amount to deposit, and "50000" is entered. The terminal confirms the deposit with the message "Abono realizado con exito por \$ 50000.0". It then asks if the user wants to perform another deposit, to which "N" is entered.

```
main.swift | Console | +  
Run  
** Bienvenido a su Banco Mexican  
o**  
  
Elija uina opción  
  
1.- Deposito  
2.- Retiro  
3.- Saldo  
4.- Salir  
1  
  
Ingresa el monto a depositar:  
50000  
Abono realizado con exito por $  
50000.0  
  
Desea realizar otro deposito? (S  
/N):  
N
```

Se generan los depósitos



This screenshot shows the continuation of the program execution. The user enters "30000" for the deposit amount, and the terminal confirms it with "Abono realizado con exito por \$ 30000.0". The menu is shown again, and option 3 (Saldo) is selected. The terminal then displays the current balance: "Tu saldo actual es de: \$ 80000.0".

```
main.swift | Console | +  
Run  
  
Ingresa el monto a depositar:  
30000  
Abono realizado con exito por $  
30000.0  
  
** Bienvenido a su Banco Mexican  
o**  
  
Elija uina opción  
  
1.- Deposito  
2.- Retiro  
3.- Saldo  
4.- Salir  
3  
  
Tu saldo actual es de: $ 80000.0
```

```
replit.com/@fernando8881/Banco-Mexicano

main.swift Console x +

Files
main.swift

Tools
Recent
>_

Cloud Services
Integrations
Deployments
Join Replit Core x

Run

** Bienvenido a su Banco Mexican
o**

Elija una opción

1.- Deposito
2.- Retiro
3.- Saldo
4.- Salir
2

Ingresa el monto a retirar:
10000
Retiro exitoso de $ 10000.0

** Bienvenido a su Banco Mexican
o**

Elija una opción
```

Se genera un retiro y se consulta el saldo

```
replit.com/@fernando8881/Banco-Mexicano

main.swift Console x +

Files
main.swift

Tools
Recent
>_

Cloud Services
Integrations
Deployments
Join Replit Core x

Run

Elija una opción

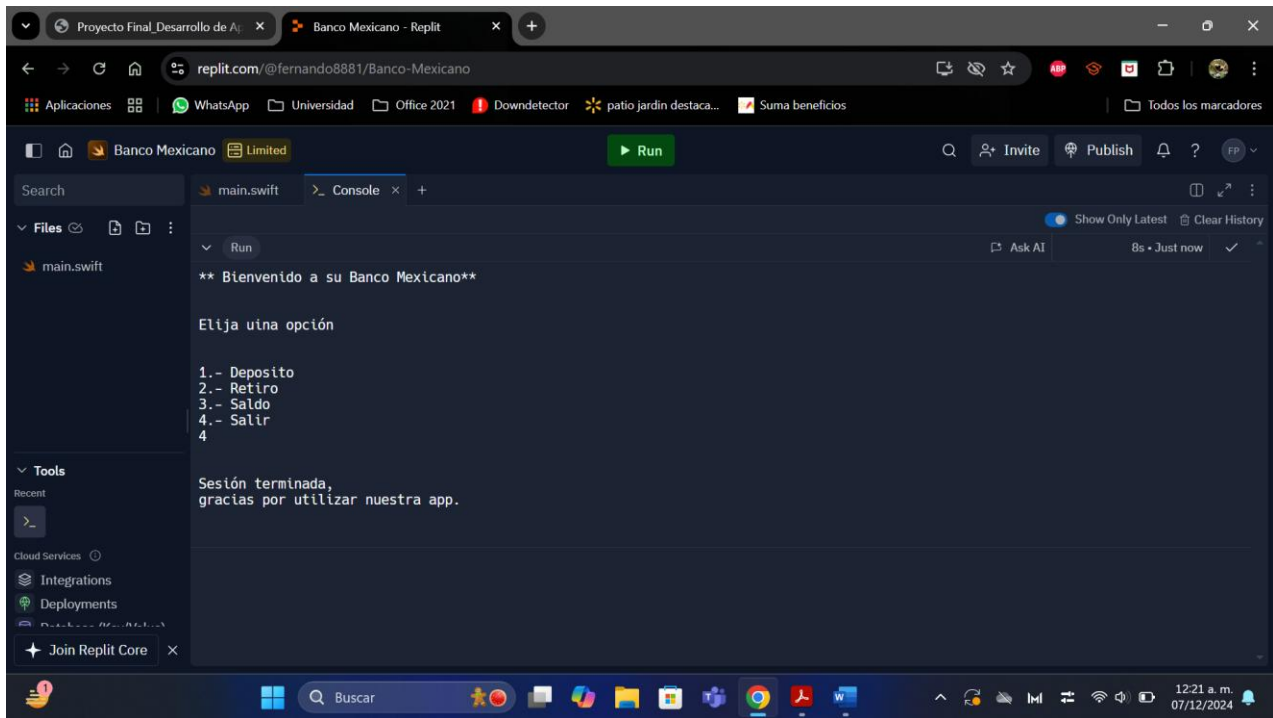
1.- Deposito
2.- Retiro
3.- Saldo
4.- Salir
3

Tu saldo actual es de: $ 70000.0

** Bienvenido a su Banco Mexican
o**

Elija una opción

1.- Deposito
2.- Retiro
3.- Saldo
4.- Salir
```

Y por ultimo se sale de la aplicación cerrando sesión de forma correcta

Enlace de GitHub: <https://github.com/Chifer888/desarrollo-de-apps-moviles-4.git>

Conclusión.

Xcode es esencial para desarrolladores que quieran publicar aplicaciones en la App Store, ya que incluye las herramientas necesarias para compilar y firmar los paquetes requeridos, y **los compiladores online**, son una gran opción para aprendizaje, experimentación y desarrollo rápido, sin sustituir completamente a un entorno de desarrollo local en proyectos más avanzados, como las aplicaciones bancarias en el ecosistema Apple, donde **Swift** combina rendimiento, seguridad y escalabilidad, que lo posiciona como una excelente elección.

¿Qué aprendo?

Que Xcode desde mi punto de vista tiene similitudes con Android, solo que enfocado a los sistemas operativos mencionados (**iOS, macOS, watchOS y tvOS**), los cuales se utilizan para el desarrollo de aplicaciones, pero con un lenguaje mas sencillo de entender al momento de programar, evitando re trabajo, y muy predictivo, en el caso de los compiladores en línea son muy útiles para poder aprender mas acerca de este lenguaje.

Referencias

ChatGPT. (n.d.). <https://chatgpt.com/c/6746b3b6-9754-8003-8af0-ee6595749f25>

Videoconferencias, conferencias web, seminario web, uso compartido de pantalla. (n.d.). Zoom.

https://academiaglobal-mx.zoom.us/rec/play/f5QMWaCWfJB4Uy0VupE-GLUxae7SwX9KqHvg0Ok32_jaS9gdRogobbfXOkqbrZ9FFHCcHAGvHJVM5TyK.Tu62DJVU67cd4Bg_?canPlayFromShare=true&from=share_recording_detail&continueMode=true&componentName=rec-play&originRequestUrl=https%3A%2F%2Facademiaglobal-mx.zoom.us%2Frec%2Fshare%2F0xbGbjl0HJL5J43u5Jx0UmUi0y7uH1GWVilRPJrrbLKANDh3tPRw15ubM8mwig18.xdF78u4lqBmTxZRU

ChatGPT. (n.d.). <https://chatgpt.com/c/674d2a24-c904-8003-9bd6-27220cfeae9d>

Videoconferencias, conferencias web, seminario web, uso compartido de pantalla. (n.d.). Zoom.

https://academiaglobal-mx.zoom.us/rec/play/AEoa155GiyW_bRNGCEBxI_8Gaav8GhXnokG5f1PoLopVBZSKYNsLsTtQeOdR2CMQms9UbfhAS9mNhOps.fEQz0QEQVF_mR92i?canPlayFromShare=true&from=share_recording_detail&continueMode=true&componentName=rec-play&originRequestUrl=https%3A%2F%2Facademiaglobal-mx.zoom.us%2Frec%2Fshare%2F0c4lfqDxFY-FuUVndC6nuVh7c27r7CZp6cfuldk4RF7kSn1bM__IE8d64YE_IVnbl.UFcC-I44Z1OeeTql