

Actividad 1 - Instalación de XCode/Aplicación 1.


Desarrollo de Aplicaciones Móviles 3.

Ingeniería en Desarrollo de Software

Tutor: Sandra Luz Lara Dévora.

Alumno: Homero Ramirez Hurtado.

Fecha: 17 de Julio del 2024.



Índice.

. Introducción.

. Descripción.

. Justificación.

. Desarrollo.

- Codificación.
- Prueba de la Aplicación.

. Conclusión.

. Referencias.



Introducción.

Swift es un lenguaje de programación moderno y versátil desarrollado por Apple. Aunque inicialmente se creó para construir aplicaciones en el ecosistema de Apple (como iOS, macOS, watchOS y tvOS), su popularidad ha crecido más allá de estas plataformas. Permíteme desglosarlo un poco:

1. **Potente e Intuitivo:** Swift combina la potencia de lenguajes como C y Objective-C con una sintaxis más amigable y expresiva. Esto lo hace ideal tanto para principiantes como para desarrolladores experimentados.
2. **Seguro y Rápido:** Swift está diseñado para minimizar errores comunes de programación. Su sistema de tipos es robusto, lo que ayuda a evitar problemas en tiempo de ejecución. Además, es altamente eficiente en términos de rendimiento.
3. **Interactivo y Divertido:** Puedes experimentar con Swift en Swift Playgrounds, una herramienta que te permite escribir y probar código de manera interactiva. Es como un patio de recreo para programadores.
4. **Sintaxis Concisa:** Swift utiliza una sintaxis limpia y concisa. Esto significa que puedes lograr más con menos líneas de código, lo que facilita la lectura y el mantenimiento.
5. **Código Abierto:** Apple liberó Swift como un proyecto de código abierto, lo que significa que cualquiera puede contribuir, aprender y usarlo sin restricciones.

En resumen, Swift es una excelente opción para desarrollar aplicaciones, desde pequeños proyectos hasta aplicaciones empresariales.

Descripción.

Xcode es un entorno de desarrollo integrado (IDE) creado por Apple. Se utiliza principalmente para desarrollar software en el ecosistema de Apple, incluyendo Mac, iPhone, iPad, Apple Watch y Apple TV. Aquí tienes algunos puntos clave sobre Xcode:

1. **Diseño y Desarrollo:** Xcode combina una variedad de funcionalidades en un flujo de trabajo unificado. Los desarrolladores pueden diseñar interfaces de usuario, escribir código, realizar pruebas y depurar sus aplicaciones, todo dentro de esta herramienta.
2. **Herramientas Integradas:** Xcode proporciona un conjunto completo de herramientas para el desarrollo de aplicaciones. Esto incluye editores de código, depuradores, analizadores de rendimiento y simuladores de dispositivos.
3. **Compatibilidad con Lenguajes:** Xcode es compatible con varios lenguajes de programación, como Swift, Objective-C y C++. Swift, en particular, se ha convertido en el lenguaje preferido para el desarrollo de aplicaciones en el ecosistema de Apple.
4. **Entorno de Pruebas:** Los desarrolladores pueden probar sus aplicaciones en simuladores de dispositivos o en dispositivos físicos conectados. Xcode facilita la creación de pruebas unitarias y funcionales.

5. **Despliegue a la App Store:** Una vez que la aplicación está lista, Xcode permite a los desarrolladores preparar y enviar sus aplicaciones a la App Store. Esto incluye la creación de archivos de distribución, la firma de código y la gestión de perfiles de aprovisionamiento.

En resumen, Xcode es una herramienta esencial para cualquier desarrollador que quiera crear aplicaciones para los productos de Apple. Su interfaz intuitiva y su conjunto de características hacen que el proceso de desarrollo sea más eficiente y efectivo.

Justificación.

Xcode es un entorno de desarrollo integrado (IDE) diseñado específicamente para los desarrolladores que trabajan en el ecosistema de Apple. Permíteme explicarte por qué es tan relevante:

1. **Construcción de Aplicaciones:** Xcode es la piedra angular para crear aplicaciones en dispositivos iOS, macOS, watchOS, tvOS e incluso visionOS. Proporciona un conjunto completo de herramientas y recursos para escribir, depurar y compilar código. Sin Xcode, sería casi imposible desarrollar aplicaciones nativas para estos sistemas operativos.
2. **Editor de Código Avanzado:** El editor de código de Xcode es potente y versátil. Admite múltiples lenguajes de programación, como Swift y Objective-C. Ofrece características inteligentes como resaltado de sintaxis, completado automático y corrección de errores. Además, Apple está fomentando el uso exclusivo de Swift para mejorar la eficiencia de las aplicaciones.
3. **Simuladores y Depuradores:** Xcode proporciona simuladores virtuales para dispositivos iOS, macOS, watchOS y tvOS. Estos simuladores permiten probar y depurar aplicaciones antes de lanzarlas al mundo real. Es fundamental para garantizar que tus aplicaciones funcionen correctamente en diferentes dispositivos y escenarios.
4. **VisionOS y Realidad Mixta:** Desde 2023, Xcode también incluye herramientas para desarrollar aplicaciones para visionOS, el sistema operativo de las futuras gafas de realidad mixta de Apple. Esto abre un mundo de posibilidades para los desarrolladores interesados en la realidad aumentada y virtual.

En resumen, Xcode no solo es una herramienta esencial, sino también una ventana al vasto universo de desarrollo de aplicaciones Apple. Aprender a usar Xcode potencia tus habilidades y te prepara para una carrera en tecnología.

Desarrollo.

Codificación:

```
#include <iostream>
```

```
#include <stdio.h>
```

```
int main() {  
    int numero;
```

```
    // Se pide al usuario que ingrese un número
```

```
    std::cout << "Ingresa un numero entero: ";
```

```
    std::cin >> numero;
```

```
    // Se verifica si el número es par o impar
```

```
    if (numero % 2 == 0) {
```

```
        std::cout << numero << " es un numero par." << std::endl;
```

```
    } else {
```

```
        std::cout << numero << " es un numero impar." << std::endl;
```

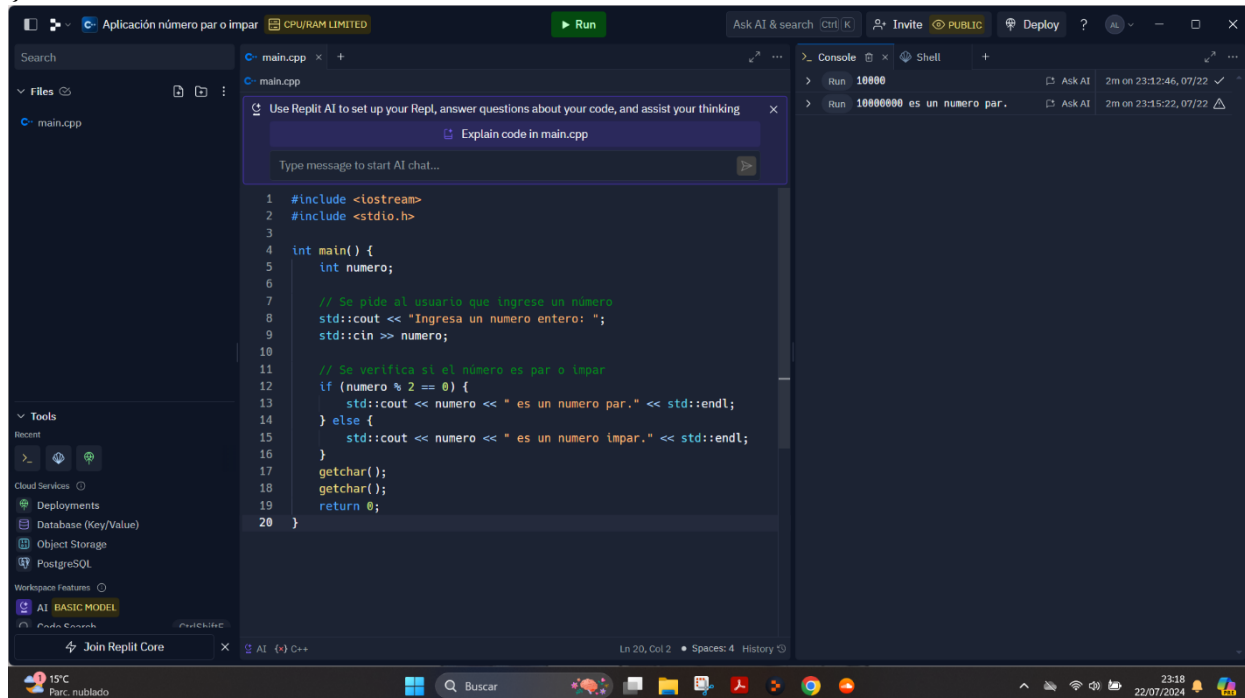
```
    }
```

```
    getchar();
```

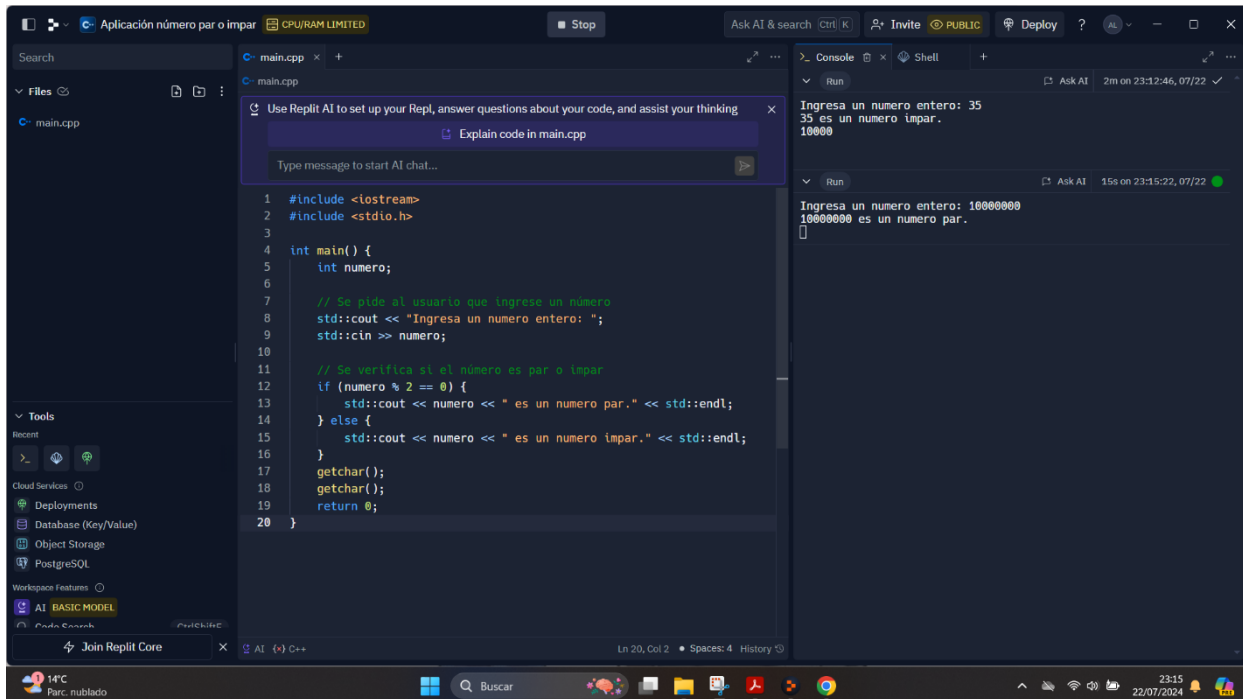
```
    getchar();
```

```
    return 0;
```

```
}
```



Prueba de la Aplicación:



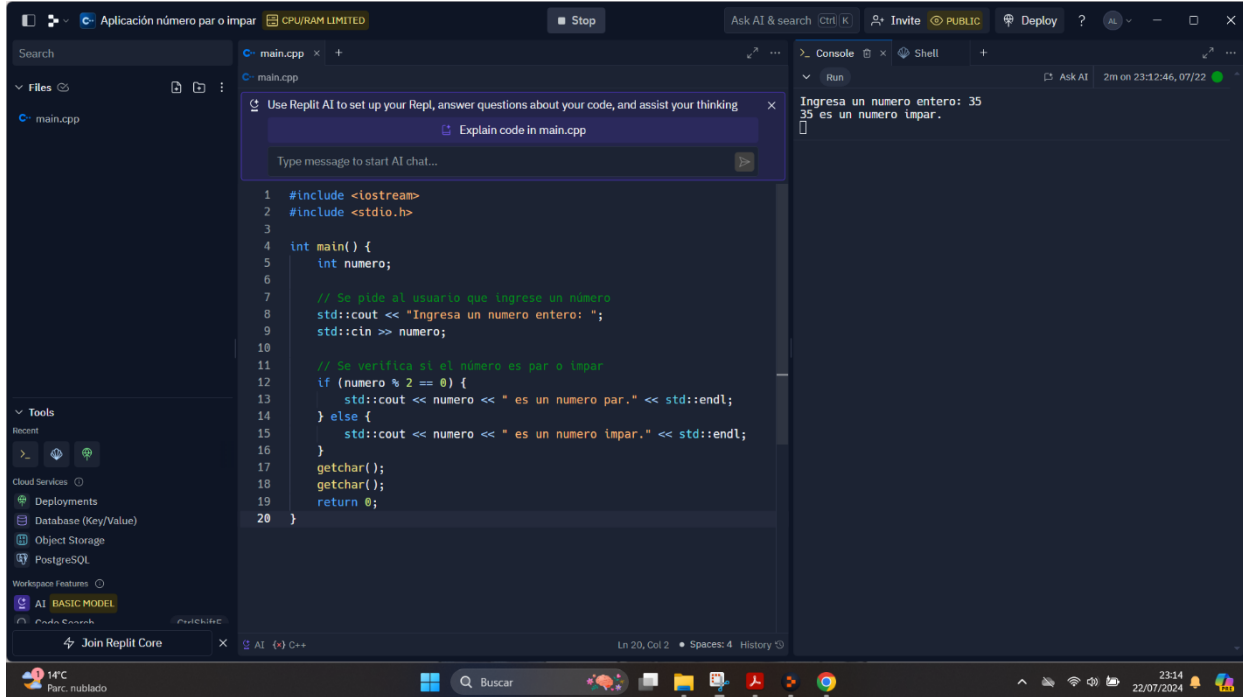
The screenshot shows a Replit IDE window titled "Aplicación número par o impar". The code in `main.cpp` is as follows:

```
1 #include <iostream>
2 #include <stdio.h>
3
4 int main() {
5     int numero;
6
7     // Se pide al usuario que ingrese un número
8     std::cout << "Ingresa un numero entero: ";
9     std::cin >> numero;
10
11     // Se verifica si el número es par o impar
12     if (numero % 2 == 0) {
13         std::cout << numero << " es un numero par." << std::endl;
14     } else {
15         std::cout << numero << " es un numero impar." << std::endl;
16     }
17     getchar();
18     getchar();
19     return 0;
20 }
```

The console output shows two test cases:

```
Ingresa un numero entero: 35
35 es un numero impar.
1000000
Ingresa un numero entero: 1000000
1000000 es un numero par.
```

Numero Par.



This screenshot is identical to the one above, showing the same C++ code and console output for the "Numero Par" application.

Numero Impar.

Conclusion.

Durante mi tiempo explorando C++ en Replit, he descubierto la belleza de la simplicidad. Mi modesta aplicación, que decide si un número es par o impar, me ha enseñado más de lo que imaginaba. Aquí están mis principales aprendizajes:

1. Fundamentos de C++: Aprendí sobre variables, operadores y estructuras de control. La sintaxis de C++ puede ser intimidante al principio, pero con práctica, se vuelve más natural.
2. Lógica Básica: La lógica detrás de determinar si un número es par o impar es fundamental. Comprender cómo usar el operador de módulo (%) para verificar la paridad fue un gran paso.
3. Replit como Entorno de Desarrollo: Descubrí cómo configurar proyectos, compilar y ejecutar código en Replit. Además, la comunidad de Replit es increíblemente útil.

En resumen, mi pequeña aplicación me mostró que incluso las tareas aparentemente triviales pueden enseñarnos mucho.

Referencias.

Video Tutoria 1