



**SENATI**

SERVICIO NACIONAL DE ADIESTRAMIENTO EN TRABAJO INDUSTRIAL

---

# **PLAN DE TRABAJO DEL ESTUDIANTE**

## 1. INFORMACIÓN GENERAL

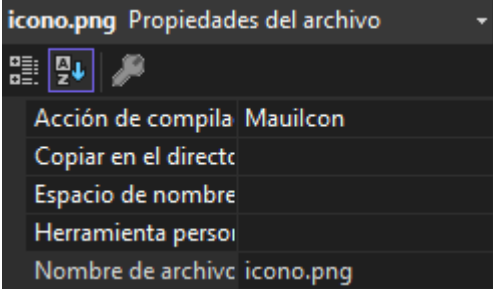
Apellidos y Nombres: Mullisaca Cahuana Dany ID: 001433727  
 Dirección Zonal/CFP: ETI – JULIACA  
 Carrera: Ingeniería de Software Con Inteligencia Artificial Semestre: VI  
 Curso/ Mód. Formativo Diseño y Desarrollo de Aplicaciones Móviles  
 Tema del Trabajo: Desarrollar estrategias comunes en el desarrollo para móviles.

## 2. PLANIFICACIÓN DEL TRABAJO

N°	ACTIVIDADES/ ENTREGABLES	CRONOGRAMA/ FECHA DE ENTREGA							
1	Configuración del proyecto	18/09							
2	Diseño de inicio de sesión (Login)	18/09							
3	Cambio del icono de la app y splash screen	18/09							
4	Diseño del TabBar		19/09						
5	Diseño de la barra lateral en pop-up		19/09						
6	Listado de los productos en la pantalla de inicio			20/09					
7	Entrega del entregable 01			20/09					
8	Desarrollo de una API				19/09				
9	Implementación de una API gratis					19/09			
10	Desarrollo de la aplicación y listar datos del JSON						19/09		
11	Entrega del entregable 02							29/09	
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									

### 3. ENTREGABLES:

Durante la investigación de estudio, deberán de dar solución a los planteamientos de cada entregable:

Nº	ENTREGABLE 1
1	<p>En esta oportunidad se desarrollara una aplicación móvil en .NET MAUI, una aplicación dedicada para la venta de prendas de vestir con ropas, zapatos y relojes.</p> <ol style="list-style-type: none"> <li>1) Creamos un proyecto en visual studio una aplicación en .NET MAUI.</li> <li>2) Diseño la pantalla de inicio de sesión, que es importante para acceder a cualquier aplicación en el cual se incluye un logo , 2 entradas de texto: usuario y contraseña, el botón para ingresar a la pagina de inicio.</li> <li>3) Cambiar el logo del Splash y el icono de la aplicación:             <ol style="list-style-type: none"> <li>a) Icono: Cargar el icono en nuestro proyecto, nuestro proyecto contiene carpetas dedicadas para los iconos. Existe la carpeta Resources/AppIcon cargar el icono dentro de la carpeta. En el icono cargado damos click derecho, seleccionamos las propiedades y seleccionamos lo que es la acción de compilación/Mauilcon.                  <p>En la carpeta Platforms/Android/AndroidManifest.xml configuramos el el icono de la aplicación.</p> <pre> android:icon="@mipmap/icono" android:roundIcon="@mipmap/icono" permission.ACCESS_NETWORK_STATE" /&gt; permission.INTERNET" /&gt; </pre> <p>También editamos la solución del proyecto el apartado del icono.</p> <pre> &lt;!-- App Icon --&gt; &lt;MauiIcon Include="Resources\AppIcon\icono.png" Color="#0FB600" ForegroundScale="0.65" /&gt; </pre> </li> <li>b) SplashScreen: Cargar el Logo en nuestro proyecto, nuestro proyecto contiene carpetas dedicadas para los iconos. Existe la carpeta Resources/AppIcon cargar el icono dentro de la carpeta. En el icono cargado damos click derecho, seleccionamos las propiedades y seleccionamos lo que es la acción de compilación/Mauil SplashScreen.                 <pre> &lt;ItemGroup&gt; &lt;MauiSplashScreen Include="Resources\Splash\splash.png" Color="#0FB600" BaseSize="128,128" /&gt; &lt;/ItemGroup&gt; </pre> </li> </ol> </li> <li>4) AppShell.xaml: Este archivo define la navegación principal en la aplicación, asegurándose de que los usuarios puedan moverse fácilmente entre las cinco secciones (Inicio, Ropa, Carrito, Zapatos y Reloj) usando una TabBar personalizada y visualmente atractiva.</li> <li>5) Menú Lateral: Que aparece como una ventana emergente en la aplicación. El fondo es semitransparente con un botón para cerrarlo. El menú está al lado izquierdo de la pantalla, inicialmente oculto, y se despliega cuando el usuario lo necesite. Dentro del menú tengo varios botones, como "Perfil", "Config", "Ayuda", "Pedidos", "Contraseña" y "Cerrar Sesión".</li> <li>6) Listado de Productos en la pantalla de inicio: Como prueba se agrega de forma de ejemplo ya que no se tiene conectado a una base de datos o una API REST donde se muestre todos los productos, en este caso solo son de ejemplo para la visualización del listado según lo requerido en el entregable 01.</li> </ol>

Nº	ENTREGABLE 2
2	<h2 style="text-align: center;">CREAR LA API</h2> <ol style="list-style-type: none"> <li> <b>Configuración del entorno</b>            Para comenzar, me aseguré de tener Visual Studio instalado en mi computadora, ya que es una de las herramientas más completas para desarrollar aplicaciones en .NET. Durante la instalación, seleccione las opciones para Desarrollo de ASP.NET y web , que incluyen las plantillas y herramientas necesarias para crear una API.         </li> <li> <b>Creación del Proyecto de API</b>            Después de abrir Visual Studio, procedió a crear un nuevo proyecto. Busqué la opción de API de ASP.NET Core y la seleccionada. En la siguiente ventana, elija un nombre significativo para el proyecto, como ViajesApi, y también seleccione una ubicación adecuada para guardarlo. Luego, seleccioné la plantilla de API y me aseguré de habilitar la opción de HTTPS , que es importante para la seguridad de la API.         </li> <li> <b>Configuración de los Modelos de Datos</b>            Con el proyecto creado, el siguiente paso fue definir los modelos de datos que representarán las entidades con las que estarán trabajando. Creé una carpeta llamada Models y dentro de ella agregué tres clases:           <ul style="list-style-type: none"> <li>❖ <b>Usuario:</b> Para almacenar información sobre los usuarios, como su nombre, apellido, correo electrónico, contraseña y foto.</li> <li>❖ <b>Destino:</b> Para almacenar información sobre los destinos, incluyendo el nombre, país e imagen.</li> <li>❖ <b>Viaje:</b> Para almacenar los detalles de los viajes, incluyendo el nombre del viaje, las fechas de inicio y fin, y las referencias al destino y al usuario.</li> </ul> </li> <li> <b>Configuración del Contexto de la Base de Datos</b>            Luego, creé una carpeta llamada Data, donde definí la clase ViajesDbContext, que hereda de DbContext. Esta clase es esencial porque se encargará de interactuar con la base de datos. En esta clase, definí las propiedades DbSet para cada uno de los modelos que cree, lo que permite a Entity Framework Core rastrear y realizar operaciones CRUD sobre ellos.             Además, configure la cadena de conexión a la base de datos en el archivo appsettings.json. Utilicé MySQL como base de datos, por lo que incluye los detalles de la conexión, como el servidor, el nombre de la base de datos, el usuario y la contraseña.         </li> <li> <b>Configuración de las Migraciones</b>            Con el contexto de la base de datos lista, utilicé las herramientas de Entity Framework Core para crear migraciones. Esto es importante para que la base de datos se genere de acuerdo con los modelos definidos. Las migraciones permiten que la estructura de la base de datos se sincronice con los cambios realizados en los modelos de datos.             Ejecute los comandos en la consola del administrador de paquetes para agregar y aplicar las migraciones. Esto creó las tablas correspondientes en la base de datos para las entidades Usuario, Destino y Viaje.         </li> <li> <b>Creación de controladores</b>            Después de tener la base de datos configurada, el siguiente paso fue crear controladores, que son las clases que manejarán las aplicaciones HTTP. Cree una carpeta de llamadas Controllers y dentro de ella agregué un controlador para manejar las operaciones relacionadas con los viajes. Este controlador, ViajesController se         </li> </ol>

encargará de responder a las solicitudes de tipo GET, POST, PUT y DELETE para la entidad Viaje.

Defina métodos en el controlador que permitirán, por ejemplo, obtener la lista de viajes, crear un nuevo viaje, actualizar un viaje existente y eliminar uno.

#### 7. Probar la API

Una vez que la API estaba configurada, probé su funcionamiento utilizando herramientas como Postman o Swagger (que se incluyen automáticamente en los proyectos de API de ASP.NET Core). Estas herramientas me permiten realizar solicitudes a la API y verificar que todo funciona correctamente.

### IMPLEMENTACION DE UNA API GRATUITA EN UN APP MAUI

#### 1. Crear la aplicación MAUI

**Paso Inicial :** Primero, use un entorno de desarrollo como Visual Studio para crear un nuevo proyecto. Seleccionaste "Aplicación .NET MAUI" para comenzar a desarrollar una aplicación multiplataforma.

**Nombre del Proyecto :** Al crear el proyecto, le diste un nombre, por ejemplo, MC que será el nombre de tu aplicación.

#### 2. Definir el modelo

**¿Qué es un modelo? :** Un modelo es una clase que describe la estructura de los datos que manejarás en tu aplicación. En este caso, crea un modelo llamado **PersonajesResponse**.

- ❖ **Propiedades:** En este modelo, define tres propiedades:
- ❖ **name:** el nombre del personaje.
- ❖ **image:** la URL de la imagen del personaje.
- ❖ **species:** la especie del personaje.

#### 3. Crear el servicio

- ❖ **Interfaz del Servicio:** Crea una interfaz de llamada **IRickAndMortyService**. Esta interfaz es un acuerdo que dice que cualquier clase que la implemente tendrá un método llamado **Obtener**, que se encargará de traer la lista de personajes.
- ❖ **Implementación del Servicio:** Luego, crea una clase de llamada **RickAndMortyService** que implementa esta interfaz. En esta clase:

Definiste la URL de la API de Rick and Morty.

Usaste **HttpClient** para hacer una solicitud a la API y obtener los datos de los personajes.

Procesaste la respuesta de la API para convertirla en objetos de tipo **PersonajesResponse** que definiste antes.

#### 4. Configurar la aplicación

**Archivo MauiProgram:** En este archivo, configura cómo se comportará tu aplicación: Registre el servicio **RickAndMortyService** para que esté disponible en toda la aplicación. Esto significa que cuando necesites obtener personajes, puedes pedirle a la aplicación que te lo dé.

También registraste la página principal **MainPage** donde mostrarán los personajes.

#### 5. Crear la página principal

- ❖ **Definición de la página:** Crea una clase de llamada **MainPage** que represente la página principal de tu aplicación.
- ❖ **Constructor:** En el constructor de **MainPage**, inicializaste los componentes de la página y llamaste al método **CargarPersonaje**, que es responsable de cargar los personajes desde el servicio.

#### 6. Cargar personajes

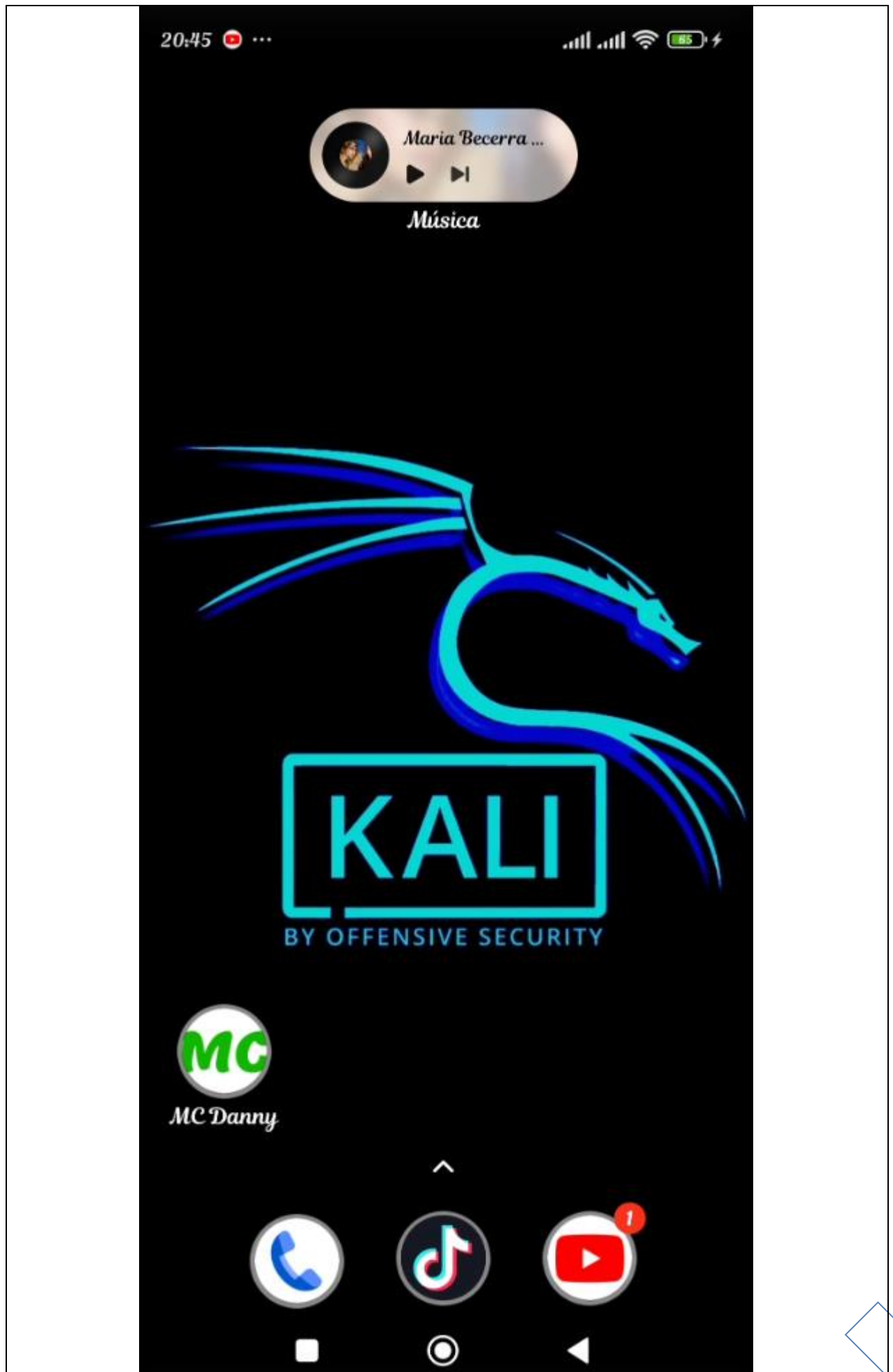
**Método CargaPersonaje:** Dentro de este método:

	<p>Intenta obtener la lista de personajes llamando al método Obtenerdel servicio. Si la obtención es exitosa, asignas la lista de personajes a un componente en la interfaz que se llama listViewPersonajes, donde se mostrarán los personajes. Si ocurre un error (por ejemplo, si no puedes acceder a la API), muestra un mensaje al usuario indicando que no se pudieron cargar los personajes.</p> <p><b>7. Diseñar la interfaz de usuario</b> Archivo XAML : Utilice XAML para diseñar la interfaz de la página principal. Estructura visual : Agregaste un ScrollView para que los usuarios puedan desplazarse hacia abajo si hay muchos personajes. Usaste un CollectionView para mostrar la lista de personajes. Cada personaje se presenta en un cuadro que incluye su imagen, nombre y especie.</p> <p><b>8. Ejecutar la aplicación</b> Finalmente, ejecutaste la aplicación en un emulador o en un dispositivo físico para ver cómo se ve y funciona. Cuando inicias la aplicación, los personajes se cargan y aparecen en la pantalla.</p>
--	--

## PROTOTIPO DISEÑADO EN FIGMA



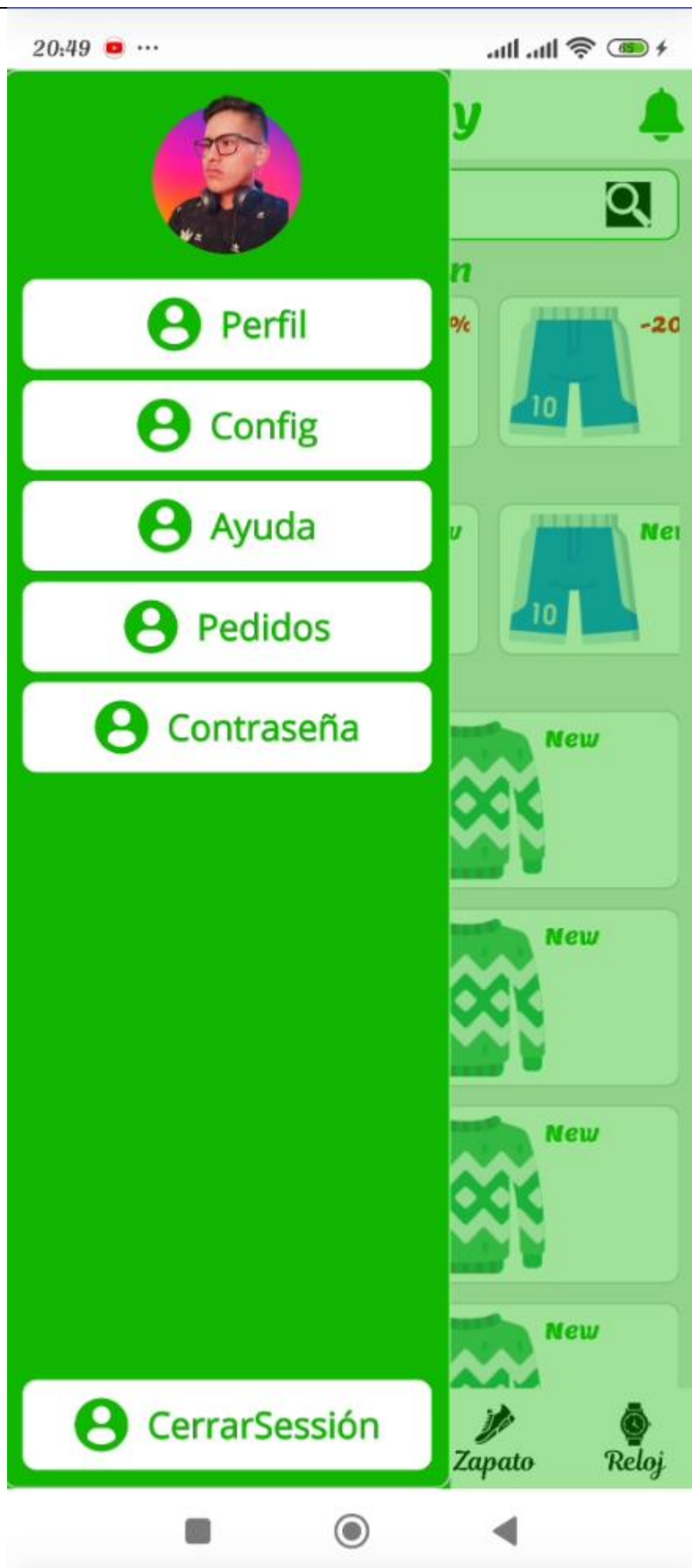
## VISTA DE LA APLICACIÓN DESARROLLADA EN .NET MAUI



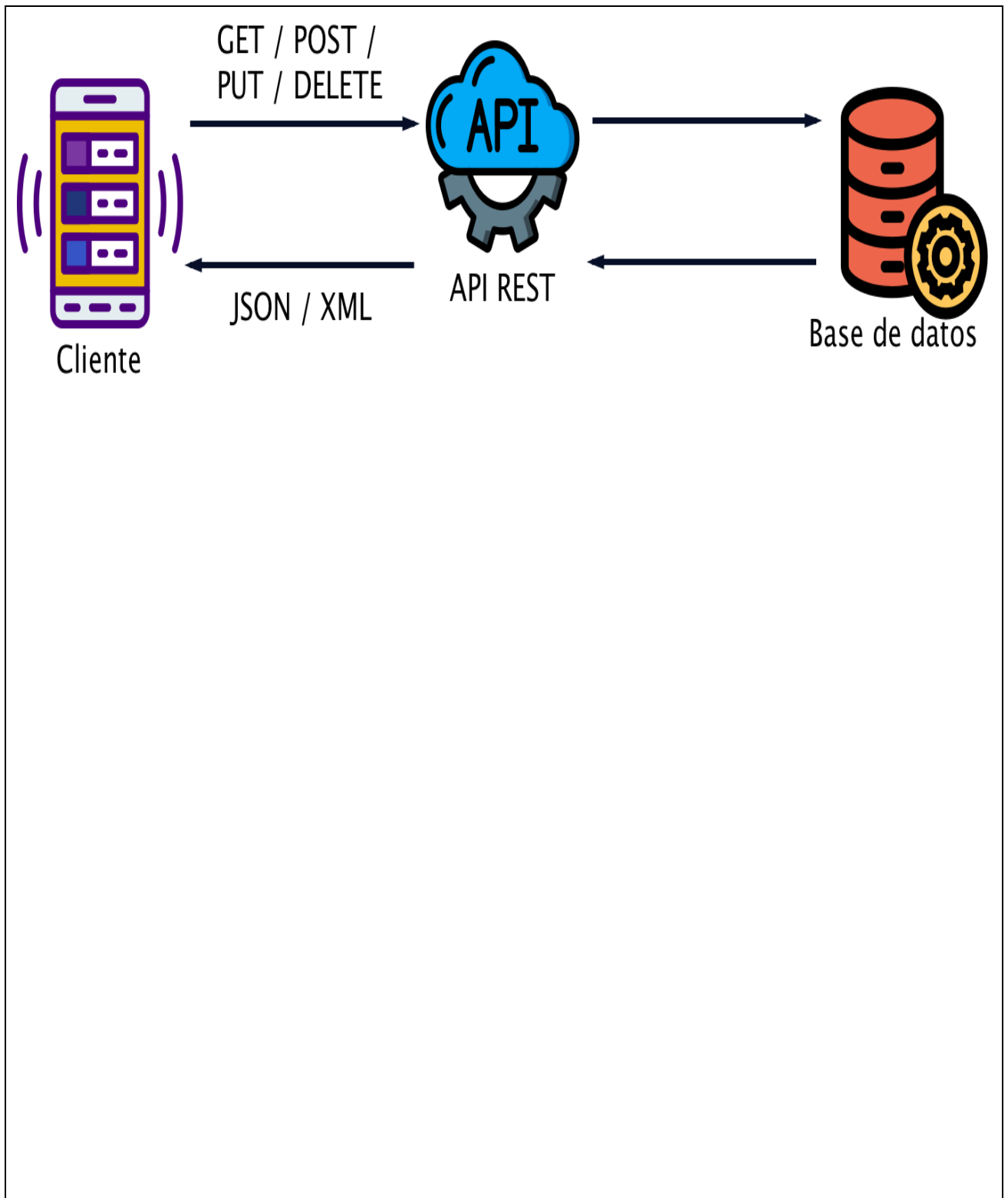














21:23



# Lista de personajes



**Rick Sanchez**  
Human



**Morty Smith**  
Human



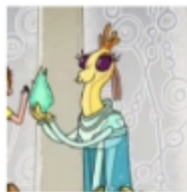
**Summer Smith**  
Human



**Beth Smith**  
Human



**Jerry Smith**  
Human



**Abadango Cluster Prir**  
Alien

## HOJA DE PLANIFICACIÓN (Entregable 2)

[illegible]

**INSTRUCCIONES:** debes ser lo más explícito posible. Los gráficos ayudan a transmitir mejor las ideas. No olvides los aspectos de calidad, medio ambiente y SHI.



## LISTA DE RECURSOS

**INSTRUCCIONES:** completa la lista de recursos necesarios para la ejecución del trabajo.

### 1. MÁQUINAS Y EQUIPOS

---

---

---

---

---

---

---

---

---

---

### 3. HERRAMIENTAS E INSTRUMENTOS

---

---

---

---

---

---

---

---

---

---

### 5. MATERIALES E INSUMOS

---

---

---

---

---

---

---

---

---

---

