



Seminario de Habilidades Prácticas – S5

Guía de Aprendizaje Práctico

INGENIERÍA DE SOFTWARE CON INTELIGENCIA ARTIFICIAL

TECNOLOGÍAS DE LA INFORMACIÓN

Presentación del Contenido de la Guía de Aprendizaje Práctico

La guía de aprendizaje práctico proporciona un recurso fundamental tanto para el instructor como para el estudiante, siendo un componente esencial durante el proceso formativo práctico. Su propósito radica en ofrecer pautas claras, que sirvan de guía, así como instrucciones técnicas, para llevar a cabo las diversas actividades prácticas en el taller o laboratorio.

Es importante señalar que los problemas están planteados en términos generales, puede el instructor complementar cada caso si lo considera, dosificándolo según las capacidades y recursos. Las propuestas y respuestas a las preguntas indicadas al final del caso deberán contemplar la mayor parte de las operaciones descritas en las tareas.



Desarrollador de aplicaciones web y móviles

Lista de Tareas Recomendadas

ÍTEM	MAT-CUR	DESCRIPCIÓN	CÓDIGO	TAREA
1	PIAD-525	DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES	HT-02	Diseña y crea interfaz gráfica de usuario
2	PIAD-525	DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES	HT-03	Crea programas con almacenamiento de datos en SQLite
3	PIAD-525	DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES	HT-04	Usa y crea aplicaciones con REST
4	PIAD-527	FULLSTACK DEVELOPER SOFTWARE	HT-02	Usa entorno de ejecución backend con JavaScript
5	PIAD-527	FULLSTACK DEVELOPER SOFTWARE	HT-03	Usa tecnología frontend con JavaScript
6	PIAD-527	FULLSTACK DEVELOPER SOFTWARE	HT-04	Diseña y crea servicios API RESTful
7	PIAD-528	TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING	HT-02	Crea y entrena modelos ML.
8	PIAD-528	TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING	HT-03	Exporta e integra modelos de Machine Learning.
9	PIAD-528	TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING	HT-04	Usa herramientas de IA para integrarlo al desarrollo de software.

Objetivos del Seminario de Habilidades Prácticas

Al finalizar el Seminario el participante estará en condiciones de:

- Diseñar, desarrollar y publicar aplicaciones móviles funcionales en la plataforma Android con una comprensión sólida de los conceptos clave y las prácticas recomendadas en el desarrollo de aplicaciones móviles.
- Contar con una sólida comprensión y habilidades prácticas en el desarrollo de software tanto en el front-end como en el back-end. Estarán preparados para crear y desplegar aplicaciones web de manera integral.
- Desarrollar aplicaciones prácticas y funcionales que emplean técnicas de Machine Learning e Inteligencia Artificial.

CURSO: PIAD-525_DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES

Tarea – HT-02

Diseña y crea interfaz gráfica de usuario.

Operaciones:

1. Diseña la GUI con XML (Layouts, vistas, recursos).
2. Revisa el ciclo de vida de un Activity.
3. Estudia principales elementos de interfaz de usuario.
4. Crea mensajes de alerta con Android Studio.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones diseñar una GUI funcional en Android Studio utilizando XML y programando la actividad principal de la aplicación, revisar el ciclo de vida de un Activity, conocer los elementos clave de la interfaz de usuario y aprender a crear mensajes de alerta para notificar al usuario sobre tareas pendientes.

Caso Práctico

La oficina de una empresa de servicios corporativos recibe visitas de clientes, proveedores y candidatos a diario. Para mejorar la eficiencia y seguridad en el control de accesos, la gerencia ha decidido desarrollar una aplicación de registro de visitas en Android que permita a los visitantes registrar su entrada y salida de manera sencilla. La aplicación debe incluir un diseño intuitivo y claro, que ayude a los visitantes a llenar los campos de nombre, empresa, y propósito de la visita, con mensajes de alerta en caso de que falte algún dato necesario. Además, el sistema deberá mostrar una lista de visitas actuales y notificar automáticamente la salida al finalizar la visita.

Por lo que se requiere que se desarrolle: El diseño de la GUI con XML (Layouts, vistas, recursos), revisa el ciclo de vida de un Activity, estudia principales elementos de interfaz de usuario y crea mensajes de alerta con Android Studio.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Documentación con lista de visitas	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora con 16gb de RAM y Android Studio instalado.	1
Dispositivo Android o emulador de Android.	1

Herramientas:	
Nombre	Cantidad
Android Studio.	1
SQLite Browser	1

Desarrollo de la Práctica

OPERACIÓN 01: Diseña la GUI con XML (Layouts, vistas, recursos).

Paso 1: Crear el proyecto en Android Studio:

Configura un nuevo proyecto con actividad en blanco (Blank Activity).

Paso 2:

Diseñar el layout principal en activity_main.xml, incluyendo elementos necesarios:

- EditText para nombre, empresa y propósito.
- Button para registrar entrada y salida.
- ListView para mostrar visitas activas.

Paso 3: Agregar recursos necesarios:

- **strings.xml**: Define los textos de etiquetas como “Nombre”, “Empresa”, “Propósito de Visita”, y mensajes de error para validación.
- **colors.xml**: Personaliza colores para la interfaz.
- **drawable**: Usa imágenes o iconos necesarios, como uno para representar visitantes activos.

Código para activity_main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView
        android:id="@+id/text_nombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/nombre" />

    <EditText
        android:id="@+id/edit_nombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/text_nombre"
        android:hint="Ingresa tu nombre" />

    <TextView
        android:id="@+id/text_empresa"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/edit_nombre"
        android:text="@string/empresa" />

    <EditText
        android:id="@+id/edit_empresa"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/text_empresa"
        android:hint="Ingresa la empresa" />

    <TextView
        android:id="@+id/text_proposito"
        android:layout_width="wrap_content"
```



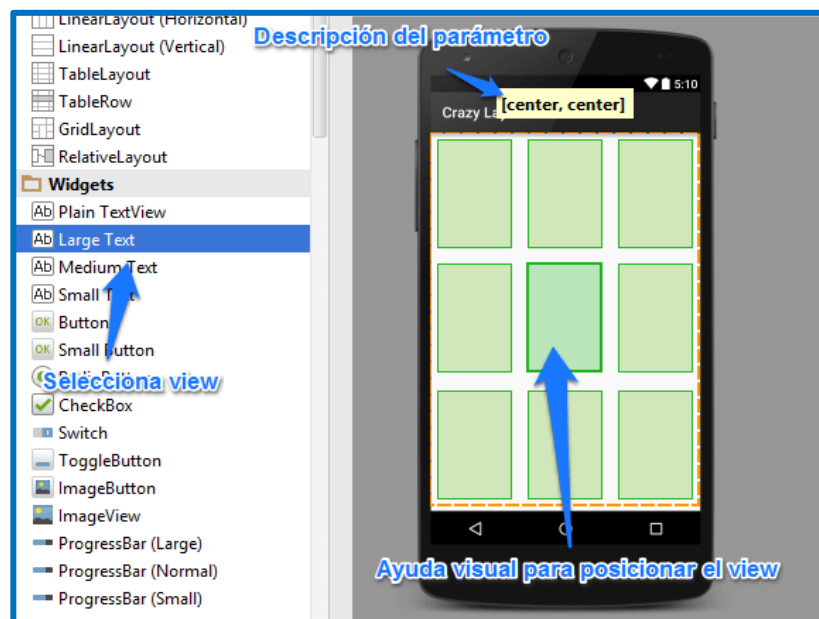
```
android:layout_height="wrap_content"
android:layout_below="@id/edit_empresa"
android:text="@string/proposito" />
```

```
<EditText
    android:id="@+id/edit_proposito"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/text_proposito"
    android:hint="Propósito de la visita" />
```

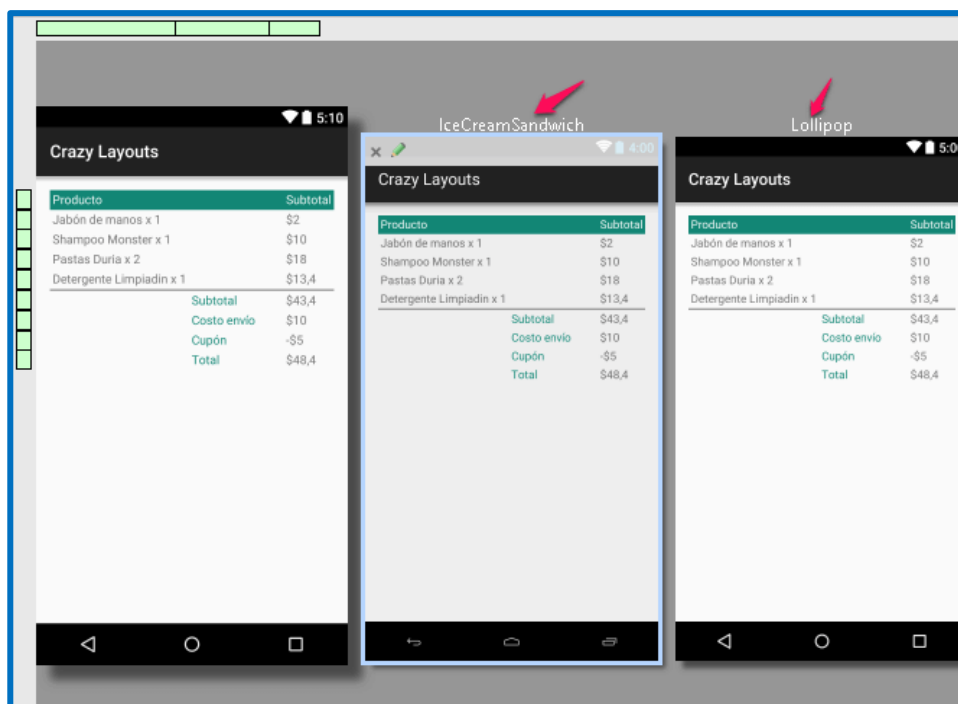
```
<Button
    android:id="@+id/btn_registrar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/edit_proposito"
    android:text="@string/registrar" />
```

```
<ListView
    android:id="@+id/list_visitas"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/btn_registrar"
    android:layout_alignParentBottom="true" />
```

```
</RelativeLayout>
```



Elementos del layout utilizando drag and drop



Layout en diferentes versiones de Android.

OPERACIÓN 02: Revisa el ciclo de vida de un Activity.

Paso 1:

Crear la clase MainActivity y configurar el método onCreate() para inicializar componentes de la UI y establecer el listener del botón btn_registrar.

Paso 2:

Implementar métodos onStart(), onResume(), onPause(), y onStop() para gestionar las interacciones de usuario y el listado de visitas.

Paso 3:

Guardar la lista de visitas en onSaveInstanceState() para mantener los datos en caso de cambios de configuración.

Código para MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

    private EditText editNombre, editEmpresa, editProposito;
    private Button btnRegistrar;
    private ListView listVisitas;
    private ArrayList<String> visitas;
    private ArrayAdapter<String> adapter;

    @Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Inicializar elementos de UI
    editNombre = findViewById(R.id.edit_nombre);
    editEmpresa = findViewById(R.id.edit_empresa);
    editProposito = findViewById(R.id.edit_proposito);
    btnRegistrar = findViewById(R.id.btn_registrar);
    listVisitas = findViewById(R.id.list_visitas);

    // Configuración de lista
    visitas = new ArrayList<>();
    adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, visitas);
    listVisitas.setAdapter(adapter);

    // Listener para botón registrar
    btnRegistrar.setOnClickListener(v -> registrarVisita());
}

private void registrarVisita() {
    String nombre = editNombre.getText().toString().trim();
    String empresa = editEmpresa.getText().toString().trim();
    String proposito = editProposito.getText().toString().trim();

    if (nombre.isEmpty() || empresa.isEmpty() || proposito.isEmpty()) {
        mostrarAlerta("Por favor, complete todos los campos.");
    } else {
        String visita = nombre + " - " + empresa + " - " + proposito;
        visitas.add(visita);
        adapter.notifyDataSetChanged();
        limpiarCampos();
    }
}

private void limpiarCampos() {
    editNombre.setText("");
    editEmpresa.setText("");
    editProposito.setText("");
}

private void mostrarAlerta(String mensaje) {
    Toast.makeText(this, mensaje, Toast.LENGTH_SHORT).show();
}

```

```

@Override
protected void onResume() {
    super.onResume();
    // Código para actualizar lista o configurar UI
}

@Override
protected void onPause() {
    super.onPause();
    // Guardar estado de la UI si es necesario
}
}

```

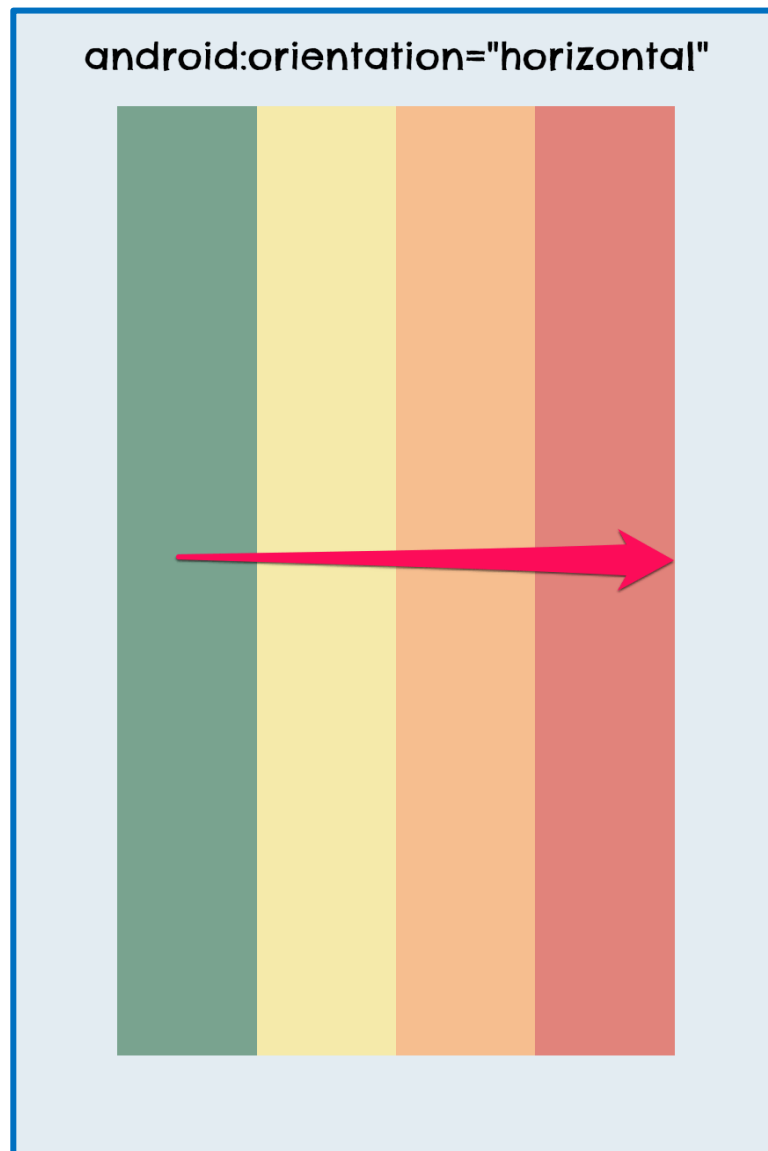
OPERACIÓN 03: Estudia principales elementos de interfaz de usuario.

A continuación, se explica cada componente y cómo se integra en el diseño general de la aplicación:

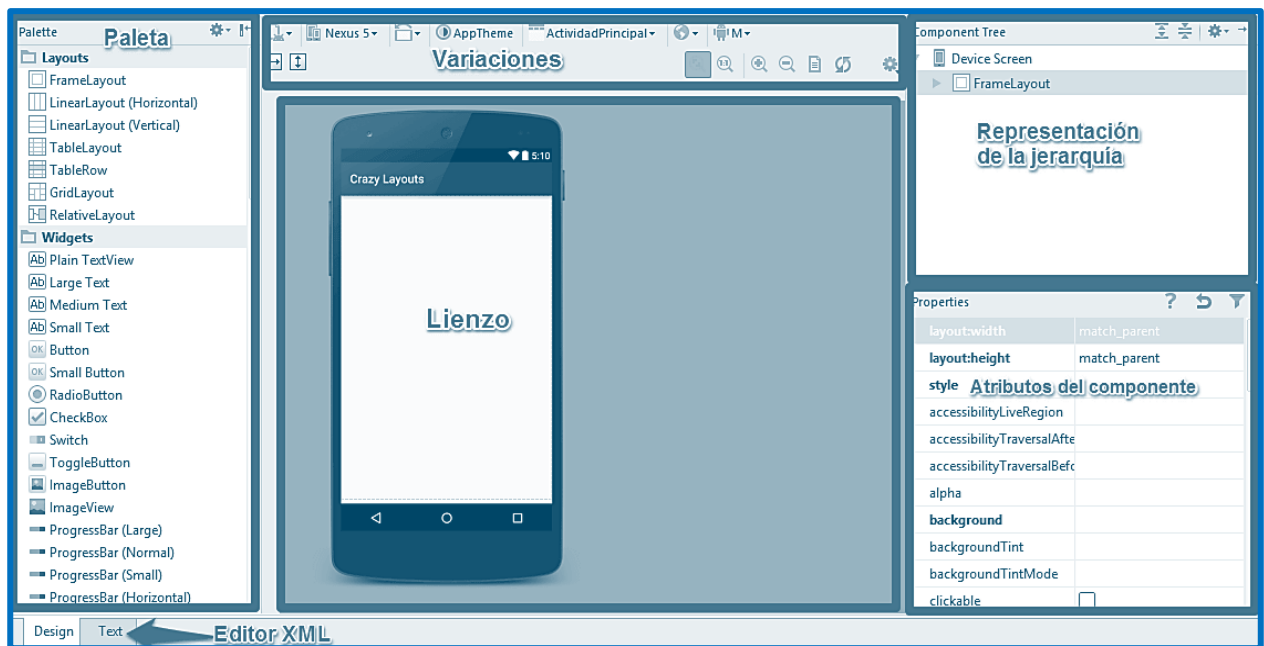
- **TextView:**
Este componente muestra etiquetas y textos explicativos en la interfaz, como “Nombre”, “Empresa”, y “Propósito de Visita”. Estos ayudan a guiar al usuario en el llenado de los campos de entrada.
- **EditText:**
Tres EditText reciben la información que el usuario debe ingresar: su nombre, la empresa a la que representa, y el propósito de la visita. Estos campos de entrada son críticos para capturar datos específicos que se registrarán.
- **Button:**
Un Button permite al usuario registrar su entrada a la oficina una vez ha completado todos los campos. Este elemento es interactivo y activa la función que verifica los datos ingresados y los agrega a la lista de visitas activas.
- **ListView:**
Un ListView se utiliza para mostrar una lista de visitas activas, permitiendo al usuario visualizar las entradas recientes registradas. Este componente es esencial para mantener un registro visual y actualizado de las visitas.
- **Toast:**
El Toast proporciona mensajes rápidos y temporales para alertar al usuario en caso de error o de una operación completada. Por ejemplo, muestra una alerta si algún campo de texto está vacío al intentar registrar la visita.

Código de Ejemplo:

```
// Configuración de los elementos en MainActivity.java
editNombre = findViewById(R.id.edit_nombre);
editEmpresa = findViewById(R.id.edit_empresa);
editProposito = findViewById(R.id.edit_proposito);
btnRegistrar = findViewById(R.id.btn_registrar);
listVisitas = findViewById(R.id.list_visitas);
```



LinearLayout en columnas.



Editor visual de Android Studio con una sección llamada "Palette"

OPERACIÓN 04: Crea mensajes de alerta con Android Studio.

Paso 1:

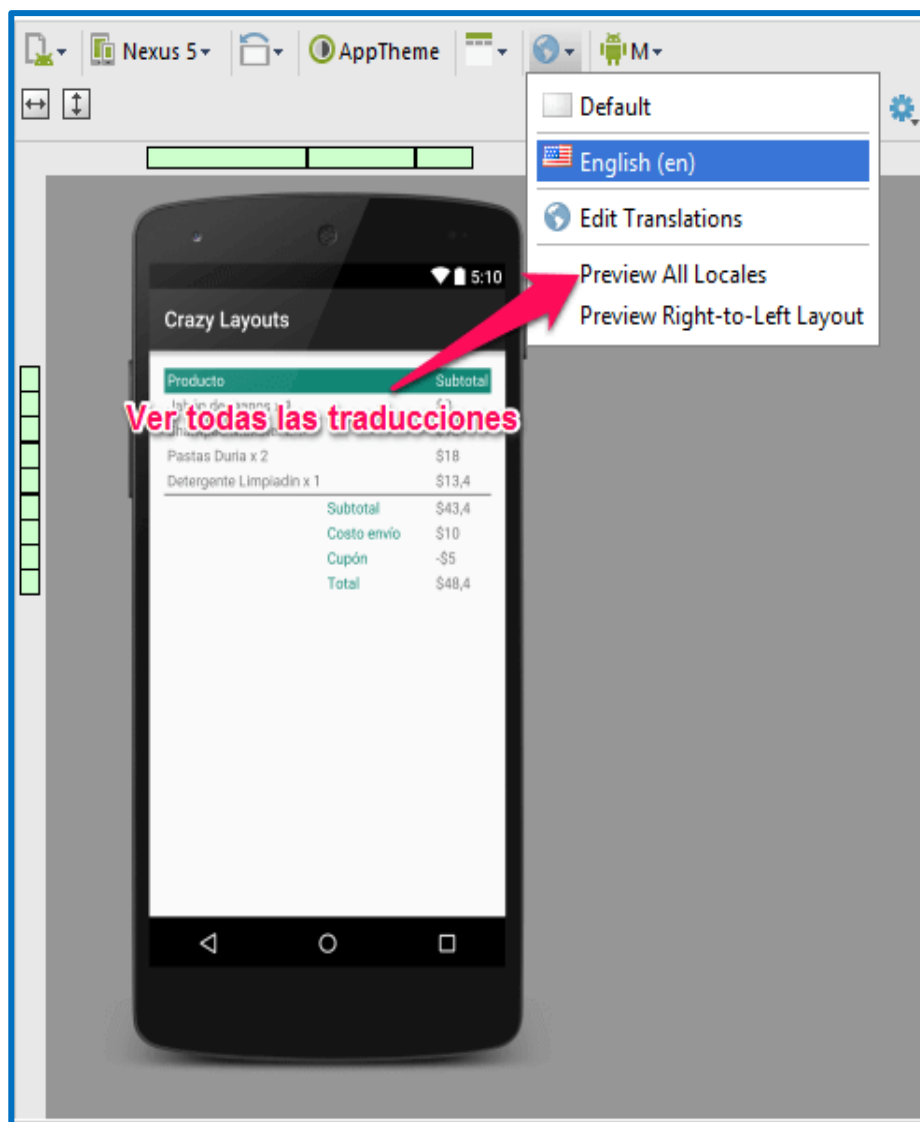
Usa el método `Toast.makeText()` para mostrar mensajes cuando algún campo está vacío.

Paso 2:

Configura mensajes de confirmación de registro, utilizando `Toast` para feedback instantáneo al usuario.

Código para mostrar alerta:

```
private void mostrarAlerta(String mensaje) {
    Toast.makeText(this, mensaje, Toast.LENGTH_SHORT).show();
}
```



Configuraciones de idiomas y otras propiedades visuales

Actividades para el Estudiante

1. ¿Qué ventaja ofrece el uso de layouts XML en comparación con la creación programática de la interfaz?
2. ¿Cuál es el propósito de implementar el ciclo de vida de una Activity en esta aplicación?
3. ¿Qué mejoras propondrías en el diseño de la interfaz?
4. ¿Cómo puedes mejorar la eficiencia de la aplicación si se agrega una base de datos para almacenar los registros de visitas?
5. ¿Cómo aplicarías los conocimientos obtenidos en un proyecto futuro?

CURSO: PIAD-525_DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES

Tarea – HT-03

Crea programas con almacenamiento de datos en SQLite

Operaciones:

1. Crea programas para comunicar actividades y fragmentos.
2. Desarrolla App considerando las funciones CRUD en SQLite.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de desarrollar una aplicación móvil que permita gestionar y acceder a la información de manera rápida y eficiente. A través de esta aplicación, se busca optimizar el almacenamiento y consulta de datos importantes, facilitando así la actualización y eliminación de registros en una base de datos local. Al emplear SQLite para la persistencia de los datos, esta solución garantiza que la información sea accesible aún sin conexión a Internet, promoviendo una gestión interna más ágil y segura. La aplicación debe ser intuitiva y permitir una navegación clara entre las diferentes opciones, para que el personal administrativo pueda operar y mantener la base de datos sin dificultad.

Caso Práctico

Un bufete de abogados quiere optimizar la gestión de su personal, facilitando el acceso a la información de sus abogados, la especialización de cada uno, sus datos de contacto, y su historial profesional. Este sistema se implementará en una aplicación Android que permita almacenar la información en una base de datos local usando SQLite, permitiendo la consulta rápida de abogados y la actualización de sus datos. El objetivo de esta aplicación es mejorar la eficiencia administrativa del bufete y asegurar el acceso seguro a la información del personal.

Por lo que se requiere que se desarrolle: Crear un programa para comunicar actividades y fragmentos y desarrolla App considerando las funciones CRUD en SQLite.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Documentación de SQLite en Android.	1
Esquema de la base de datos y diagrama de clases.	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora con 16gb de RAM y Android Studio instalado.	1
Dispositivo Android o emulador de Android.	1

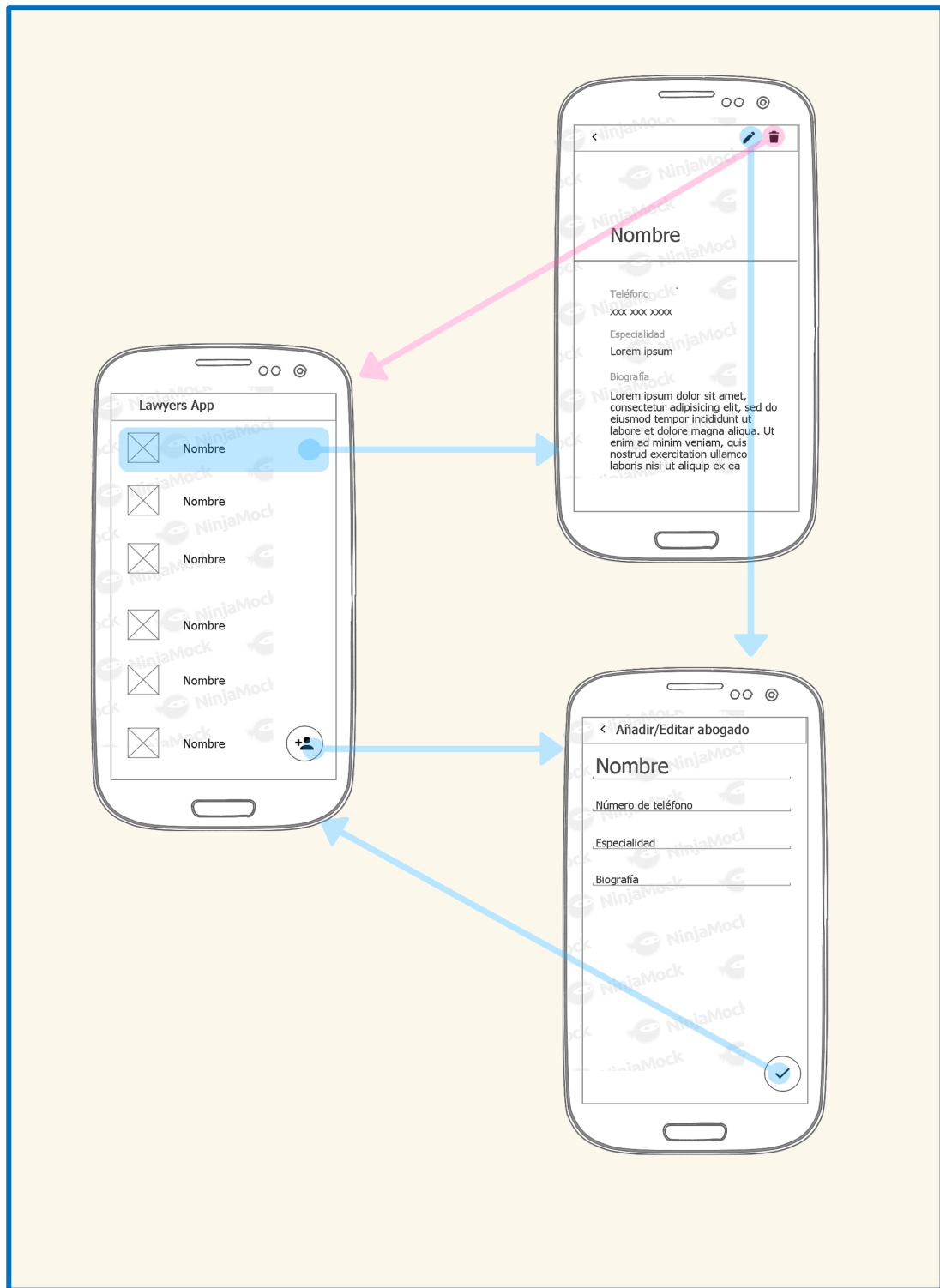
Herramientas:	
Nombre	Cantidad
Android Studio.	1
SQLite Browser	1

Desarrollo de la Práctica

OPERACIÓN 01: Crea programas para comunicar actividades y fragmentos.

En esta operación, crearemos dos actividades principales y configuraremos la comunicación entre ellas usando Intents.

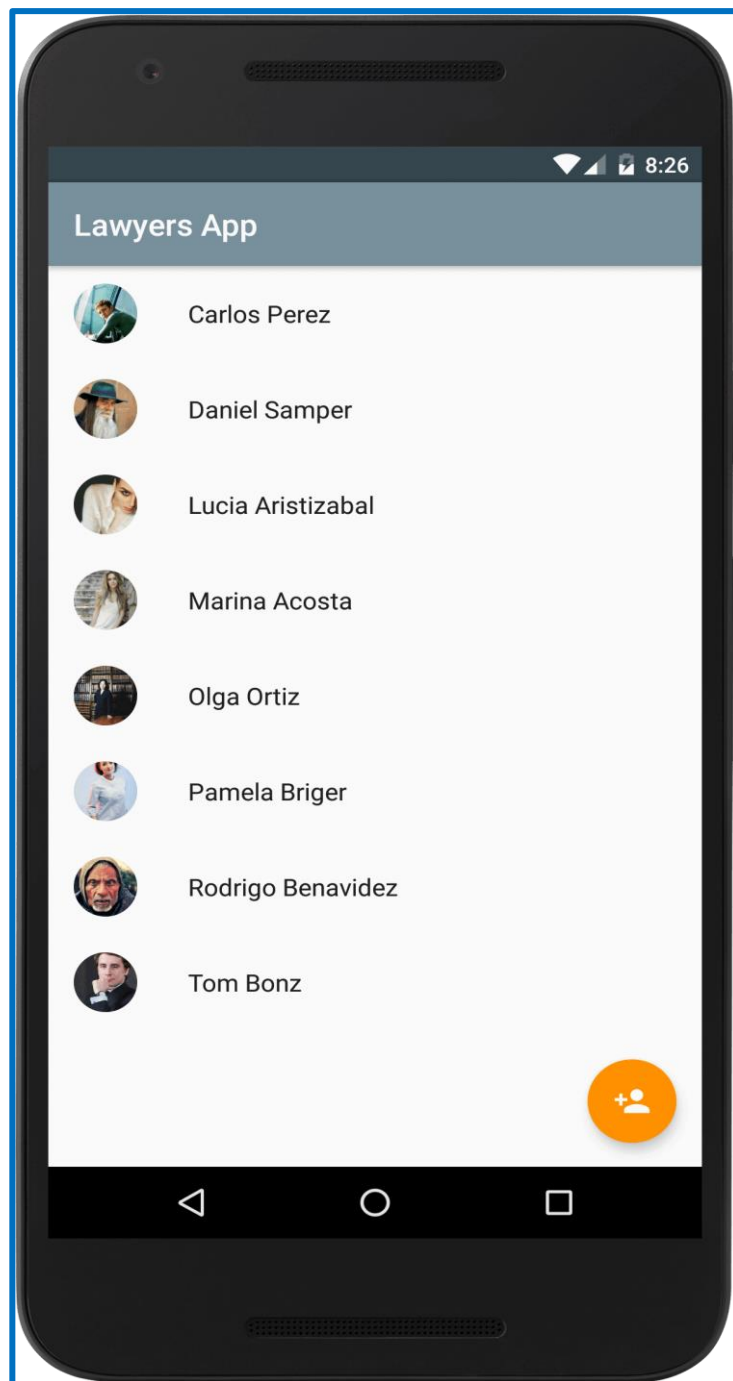
El siguiente es un wireframe que muestra los puntos de interacción:



Wireframe para la aplicación.

Las actividades y fragmentos incluirán:

1. LawyersActivity: Muestra la lista de abogados.



[Imagen de referencia: lista de abogados.](#)

2. LawyerDetailActivity: Muestra los detalles de un abogado específico.



Imagen de referencia: información de un abogado.

Código para LawyersActivity:

Esta actividad muestra la lista de abogados en un ListView. Incluye un FloatingActionButton que permite al usuario agregar un nuevo abogado.

```
public class LawyersActivity extends AppCompatActivity {
    public static final String EXTRA_LAWYER_ID = "extra_lawyer_id";
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lawyers);

    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = findViewById(R.id.fab);
    fab.setOnClickListener(view -> {
        Intent intent = new Intent(this, AddEditLawyerActivity.class);
        startActivityForResult(intent, REQUEST_ADD_LAWYER);
    });

    loadLawyersList();
}

private void loadLawyersList() {
    ListView listView = findViewById(R.id.lawyers_list);
    // Configuración del adaptador para mostrar la lista
}
}

```

Código para LawyerDetailActivity:

Cuando el usuario selecciona un abogado en LawyersActivity, se abre esta actividad, la cual muestra los detalles completos del abogado.

```

public class LawyerDetailActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lawyer_detail);

        String lawyerId = getIntent().getStringExtra(LawyersActivity.EXTRA_LAWYER_ID);

        // Cargar detalles del abogado utilizando el ID
    }
}

```

OPERACIÓN 02: Desarrolla App considerando las funciones CRUD en SQLite.

Para implementar las funciones CRUD en SQLite, se utilizarán las siguientes clases y métodos:

1. **Creación de la Clase LawyerDbHelper:** Esta clase se encargará de la conexión y gestión de la base de datos.

```
public class LawyerDbHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Lawyers.db";
    private static final int DATABASE_VERSION = 1;

    public LawyerDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String SQL_CREATE_LAWYER_TABLE = "CREATE TABLE lawyer (" +
            "_id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "id TEXT NOT NULL," +
            "name TEXT NOT NULL," +
            "specialty TEXT NOT NULL," +
            "phone TEXT NOT NULL," +
            "bio TEXT," +
            "avatarUri TEXT);";
        db.execSQL(SQL_CREATE_LAWYER_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS lawyer");
        onCreate(db);
    }
}
```

2. Funciones CRUD:

- **Insertar Datos:** Método addLawyer() en LawyerDbHelper.

```
public long addLawyer(Lawyer lawyer) {
    SQLiteDatabase db = getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("name", lawyer.getName());
    values.put("specialty", lawyer.getSpecialty());
    values.put("phone", lawyer.getPhone());
    values.put("bio", lawyer.getBio());
    values.put("avatarUri", lawyer.getAvatarUri());
}
```

```

        return db.insert("lawyer", null, values);
    }

```

- **Leer Datos:** Método getAllLawyers() para leer todos los registros de la tabla lawyer.

```

public Cursor getAllLawyers() {
    SQLiteDatabase db = getReadableDatabase();
    return db.query("lawyer", null, null, null, null, null, null);
}

```

- **Actualizar Datos:** Método updateLawyer() en LawyerDbHelper.

```

public int updateLawyer(Lawyer lawyer, String lawyerId) {
    SQLiteDatabase db = getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("name", lawyer.getName());
    values.put("specialty", lawyer.getSpecialty());
    values.put("phone", lawyer.getPhone());
    values.put("bio", lawyer.getBio());

    return db.update("lawyer", values, "id = ?", new String[]{lawyerId});
}

```

[Editar la información de un abogado.](#)

- **Eliminar Datos:** Método deleteLawyer() en LawyerDbHelper.

```
public int deleteLawyer(String lawyerId) {  
    SQLiteDatabase db = getWritableDatabase();  
    return db.delete("lawyer", "id = ?", new String[]{lawyerId});  
}
```

Actividades para el Estudiante

1. ¿Cuáles son las principales ventajas de utilizar SQLite en aplicaciones Android?
2. ¿Cómo aseguraste que la base de datos solo crea registros únicos por cada abogado?
3. ¿Cómo te aseguraste de manejar las transacciones de la base de datos de forma segura?
4. ¿De qué forma verificaste el correcto funcionamiento de los métodos CRUD implementados?
5. ¿Qué cambios implementarías en la app para agregar una función de búsqueda de abogados por especialidad?

CURSO: PIAD-525_DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES

Tarea – HT-04

Usa y crea aplicaciones con API REST.

Operaciones:

1. Consume servicios de API RESTful en Android Studio.
2. Crea aplicación de caso práctico.
3. Identifica requerimientos para la publicación de una App.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de diseñar y desarrollar una aplicación en Android Studio que permita al usuario final consumir el servicio RESTful. Además, se definirán los requisitos necesarios para su futura publicación en Google Play.

Caso Práctico

Una empresa de servicios financieros, FinCo, necesita una aplicación móvil para Android que permita a sus empleados iniciar sesión de manera segura y acceder a su perfil y datos esenciales desde cualquier ubicación. La aplicación debe comunicarse con una API segura que gestione la autenticación y entregue datos personalizados para cada usuario.

Por lo que se requiere que se desarrolle: Consume servicios de API RESTful en Android Studio, crea aplicación de caso práctico e identifica requerimientos para la publicación de una App

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Documentación de Retrofit y API REST	1
Cuenta de Desarrollador de Google Play	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora/Laptop con 16gb de memoria RAM	1
Dispositivo Android físico (o emulador)	1
Conexión a Internet	1

Herramientas:	
Nombre	Cantidad
Android Studio	1
Java Development Kit (JDK)	1

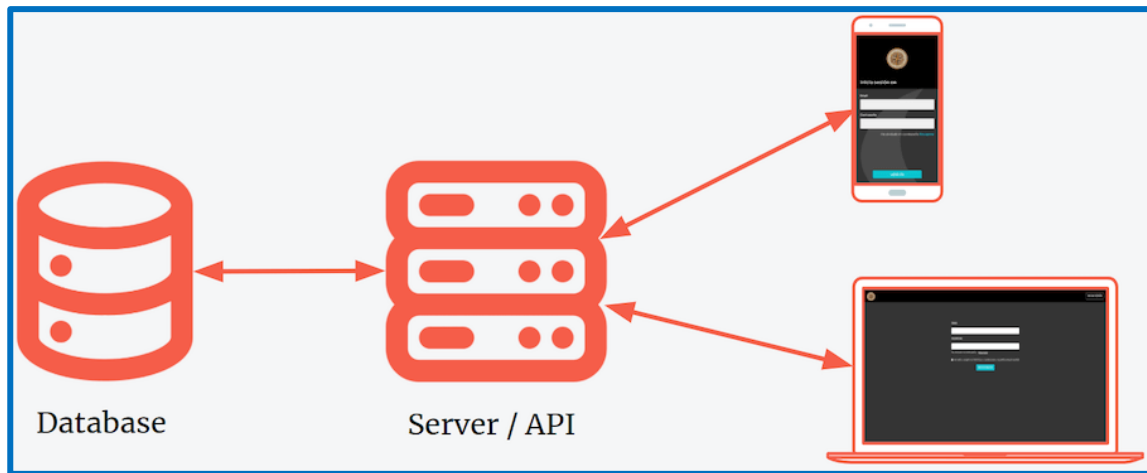
Desarrollo de la Práctica

OPERACIÓN 01: Consume servicios de API RESTful en Android Studio.

Paso 1: Configuración del Proyecto en Android Studio

- Crear un nuevo proyecto en Android Studio.
 - **Nombre del proyecto:** FinCoLoginApp.
 - **Actividad inicial:** Empty Activity.
- Agregar dependencias en build.gradle:

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'  
implementation 'com.squareup.okhttp3:logging-interceptor:4.9.0'
```



Diseño de inicio de sesión que se implementará en el proyecto.

Paso 2: Configuración de Retrofit para Consumo de API

- Crear una clase ApiClient para gestionar la conexión con la API:

```

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class ApiClient {
    private static Retrofit retrofit = null;

    public static Retrofit getClient(String baseUrl) {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                .baseUrl(baseUrl)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return retrofit;
    }
}
  
```

- Crear una clase ApiService que defina los endpoints:

```

import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.POST;

public interface ApiService {
    @POST("login")
    Call<LoginResponse> loginUser(@Body LoginRequest request);
}
  
```

Paso 3: Lógica para Autenticación

- Crear la clase LoginRequest para representar los datos de inicio de sesión:

```
public class LoginRequest {  
    private String email;  
    private String password;  
  
    public LoginRequest(String email, String password) {  
        this.email = email;  
        this.password = password;  
    }  
}
```

- Crear la clase LoginResponse para manejar la respuesta del servidor:

```
public class LoginResponse {  
    private String token;  
    private String message;  
    private boolean success;  
  
    // Getters y setters  
}
```

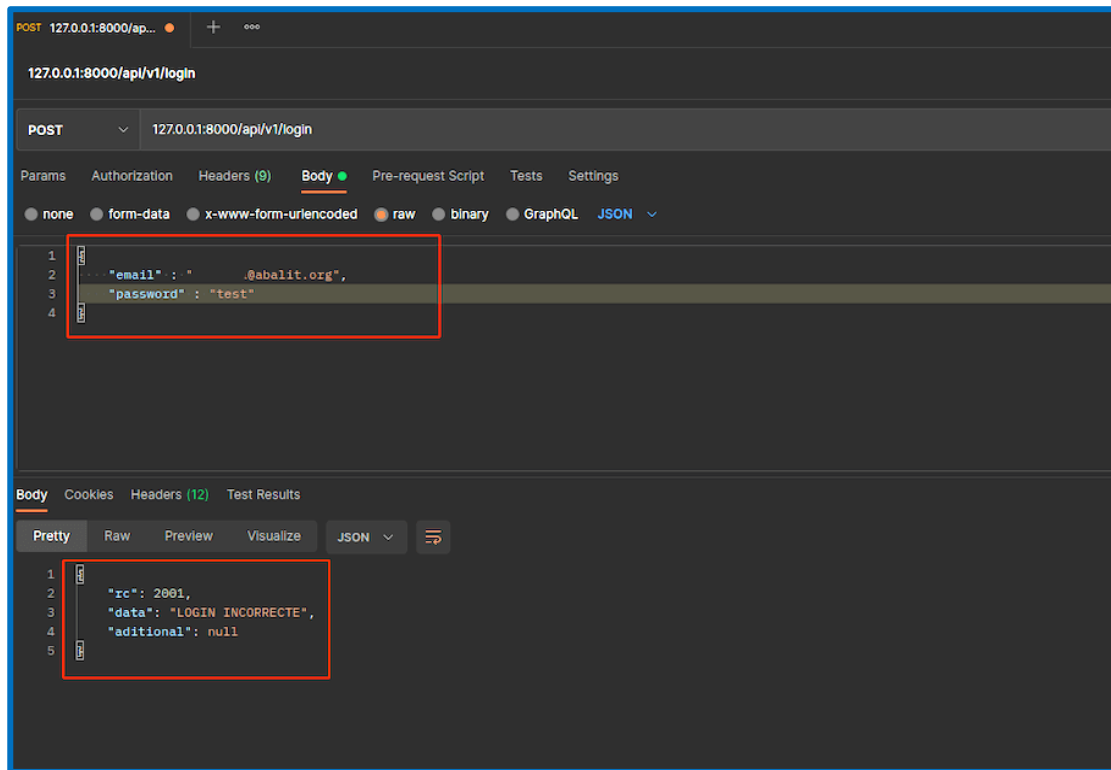
- Crear el método para el inicio de sesión en la actividad:

```
ApiService apiService =  
ApiClient.getClient("https://api.finco.com/").create(ApiService.class);  
LoginRequest request = new LoginRequest("email@example.com", "password123");  
  
Call<LoginResponse> call = apiService.loginUser(request);  
call.enqueue(new Callback<LoginResponse>() {  
    @Override  
    public void onResponse(Call<LoginResponse> call, Response<LoginResponse>  
response) {  
        if (response.isSuccessful()) {  
            // Procesar respuesta de éxito  
        } else {  
            // Procesar error  
        }  
    }  
})  
  
@Override  
public void onFailure(Call<LoginResponse> call, Throwable t) {
```

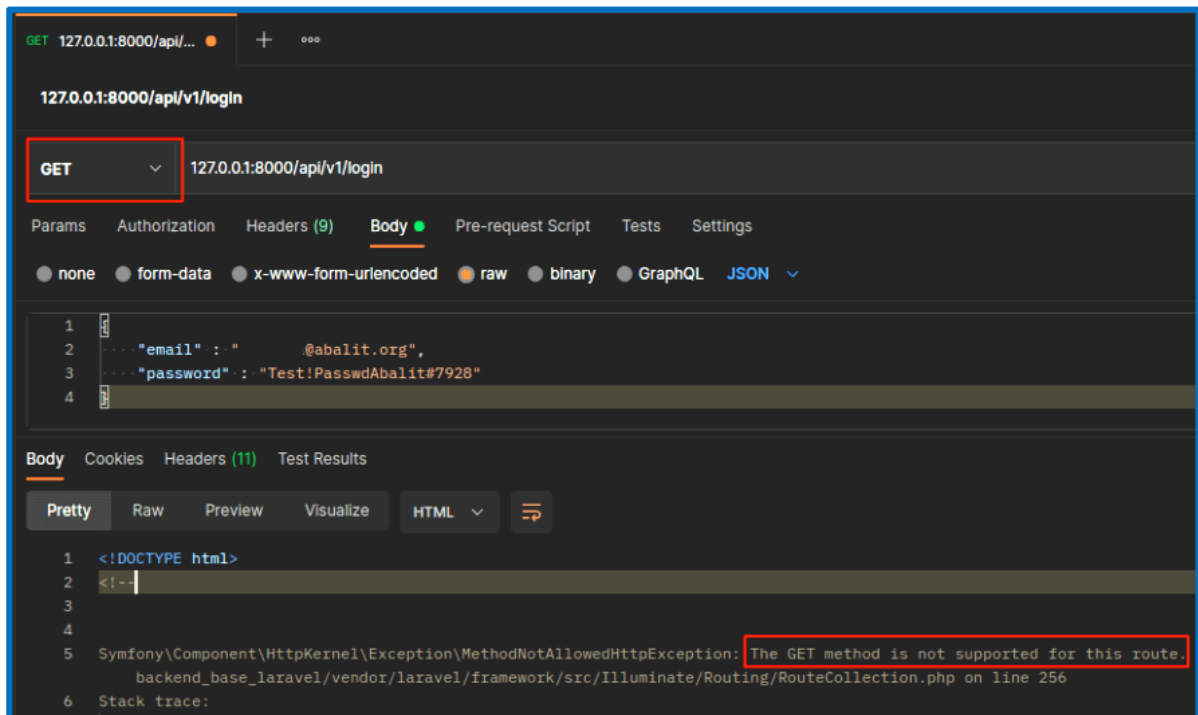
```

    // Manejar error de conexión
}
});

```



Captura de pantalla del formulario de inicio de sesión en la aplicación



Error por método HTTP incorrecto

OPERACIÓN 03: Identifica requerimientos para la publicación de una App.

Para publicar la aplicación en Google Play, se deben seguir estos pasos:

- **Crear una cuenta de desarrollador de Google** (pago único de \$25 USD).
- **Configurar la seguridad:** asegurarse de manejar adecuadamente los datos de usuarios y tokens de API.
- **Preparar el APK/AAB:** generar el archivo AAB desde Android Studio.
- **Configurar la descripción:** proporcionar información sobre las características y requisitos de la aplicación.
- **Crear gráficos de la app:** imágenes de la aplicación para la galería en Play Store.

Actividades para el Estudiante

1. ¿Por qué es importante validar las credenciales de usuario en el lado del servidor?
2. ¿Qué conocimientos de programación y diseño fueron más útiles para desarrollar la aplicación?
3. ¿Qué problemas técnicos encontraste en la conexión con la API y cómo los resolviste?
4. ¿Qué habilidades nuevas o conocimientos obtuviste al trabajar con Retrofit y la API REST?
5. ¿Qué nuevas funcionalidades considerarías en una futura actualización de la aplicación para mejorar el seguimiento de envíos?

CURSO: PIAD-527_FULLSTACK DEVELOPER SOFTWARE

Tarea – HT-02

Usa entorno de ejecución backend con JavaScript

Operaciones:

1. Instala y configura entorno de desarrollo con Node JS.
2. Crea programas de operaciones CRUD con Node JS y MySQL.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de crear un sistema CRUD en Node.js para gestionar un inventario, con una interfaz web donde se puedan realizar operaciones de creación, lectura, actualización y eliminación (CRUD) de los productos en la base de datos.

Caso Práctico

Una tienda de productos electrónicos necesita una aplicación para gestionar su inventario de productos. La tienda desea que el sistema permita registrar productos, consultar el stock, actualizar la información de los productos y eliminar registros cuando un artículo ya no esté disponible. Se utilizarán Node.js y MySQL para desarrollar la solución.

Por lo que se requiere que se desarrolle: Instala y configura entorno de desarrollo con Node JS y crea programas de operaciones CRUD con Node JS y MySQL.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Documentación de Node.js y MySQL	1
Manual de Postman o Insomnia	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora/Laptop	1
Conexión a Internet	1

Herramientas:	
Nombre	Cantidad
Node.js	1
MySQL2	1

Desarrollo de la Práctica

OPERACIÓN 01: Instala y configura entorno de desarrollo con Node JS.

Paso 1:

Crear un nuevo directorio para el proyecto:

```
mkdir crud-inventario-electronicos  
cd crud-inventario-electronicos  
npm init
```

Completa las preguntas de configuración y genera el archivo package.json.

Paso 2:

Instalar las dependencias necesarias:

```
npm install express mysql body-parser nodemon --save
```

```
npm install --save-dev nodemon
```

Nombres	Tipo	Longitud/Valores	Predefinido	Colapso	Atributos	Nota	Índice	Comentarios	Virtualidad	Mover columna	Tipo de medio	Visualización en el navegador	Transferir visualización
ID <small>Requerido desde las columnas con datos</small>	int	11	String					PRIMARY KEY	<input checked="" type="checkbox"/>				
nombre <small>Requerido desde las columnas con datos</small>	varchar	255	String										
sexo <small>Requerido desde las columnas con datos</small>	varchar	255	String										
edad <small>Requerido desde las columnas con datos</small>	varchar	255	String										
email <small>Requerido desde las columnas con datos</small>	varchar	255	String										
img <small>Requerido desde las columnas con datos</small>	varchar	255	String										
nombre_m <small>Requerido desde las columnas con datos</small>	text		JSON										
nombre_f <small>Requerido desde las columnas con datos</small>	text		JSON										

Consistencia de la tabla:

Colapso: ☐ Múltiple de documentación: ☐

Definición de la partición:

Dividir por: (Expresión o lista de columnas)

Columnas:

Previsualizar SQL:

Generación de campos en MySQL.

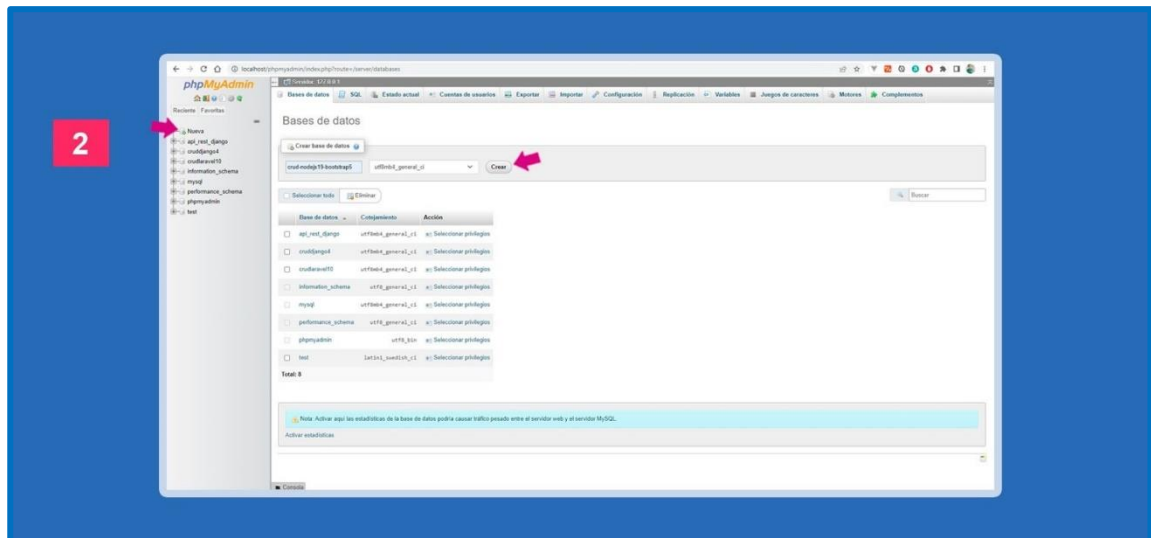
5

Tabla creada.

Paso 3:

Configurar el archivo package.json para agregar un script de inicio:

```
"scripts": {
  "start": "nodemon app.js"
}
```



Contenido del archivo package.json después de ejecutar npm init.

Paso 4:

Crear el archivo principal app.js para configurar el servidor de Node.js:

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const port = 3000;
app.listen(port, () => {
  console.log(`Servidor ejecutándose en http://localhost:${port}`);
});
```

Paso 5:

Verificar la instalación ejecutando:

```
npm start
```

Navega a <http://localhost:3000> para confirmar que el servidor está activo.

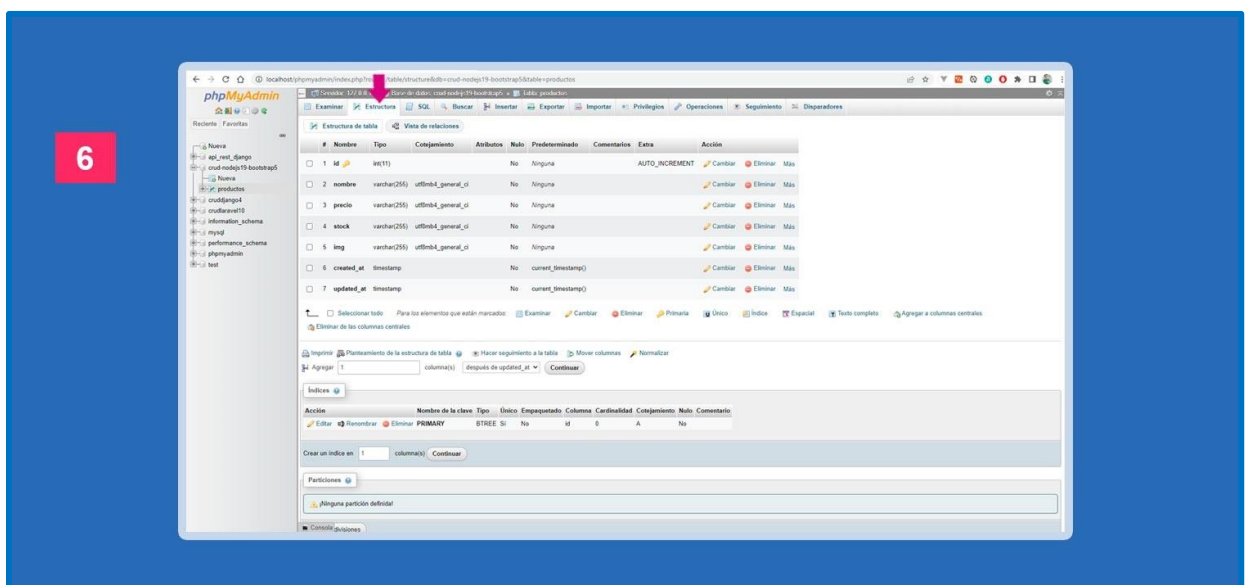
OPERACIÓN 02: Crea programas de operaciones CRUD con Node JS y MySQL.

Paso 1: Configuración de la Base de Datos en MySQL.

- Utiliza phpMyAdmin o MySQL Workbench para crear una base de datos llamada inventario_tienda.

- Dentro de esta base de datos, crea una tabla llamada productos:

```
CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(255),
  precio DECIMAL(10, 2),
  stock INT,
  imagen VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
  CURRENT_TIMESTAMP
);
```



Base de datos con nombre inventario tienda en phpMyAdmin.

Paso 2: Conexión a la Base de Datos.

- Crea una carpeta config y dentro un archivo db.js para gestionar la conexión:

```
const mysql = require('mysql');
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'inventario_tienda'
});

connection.connect(err => {
  if (err) throw err;
```

```
console.log("Conectado a la base de datos MySQL");
});
```

```
module.exports = connection;
```


Paso 3: Desarrollar el Controlador CRUD.

- Crea una carpeta controllers y un archivo productosController.js para implementar las funciones CRUD.

- Operación de Creación (Create):

```
exports.crearProducto = (req, res) => {
  const producto = {
    nombre: req.body.nombre,
    precio: req.body.precio,
    stock: req.body.stock,
    imagen: req.file ? req.file.filename : ""
  };
  const query = "INSERT INTO productos SET ?";
  connection.query(query, producto, (err, result) => {
    if (err) throw err;
    res.send('Producto creado exitosamente');
  });
};
```

2



Aplicación puede interactuar [CREAR] con la base de datos inventario_tienda.

- Operación de Lectura (Read):

```
exports.listarProductos = (req, res) => {
  connection.query("SELECT * FROM productos", (err, results) => {
    if (err) throw err;
    res.json(results);
  });
};

exports.obtenerProductoPorId = (req, res) => {
  const id = req.params.id;
  connection.query("SELECT * FROM productos WHERE id = ?", [id], (err, result) => {
    if (err) throw err;
    res.json(result[0]);
  });
};
```



Aplicación puede interactuar [LEER] con la base de datos inventario tienda.

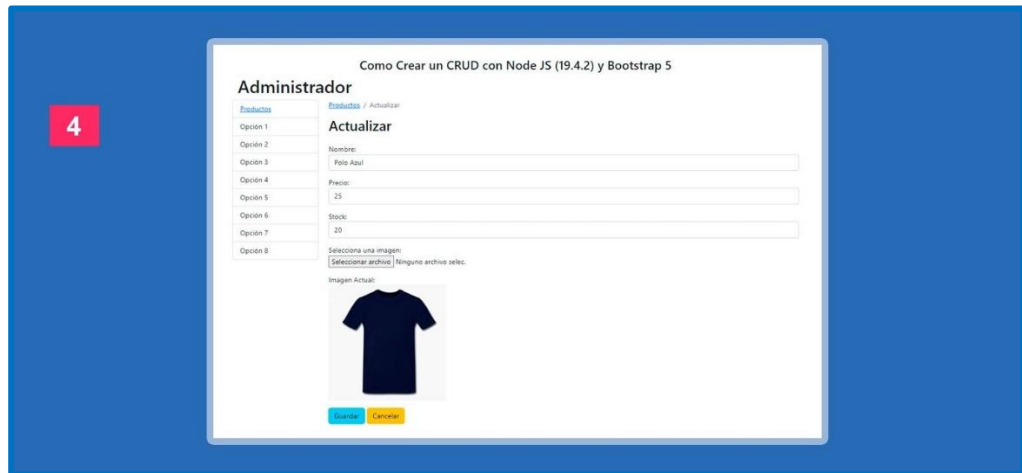
- Operación de Actualización (Update):

```
exports.actualizarProducto = (req, res) => {
  const id = req.params.id;
  const productoActualizado = {
    nombre: req.body.nombre,
    precio: req.body.precio,
    stock: req.body.stock,
    imagen: req.file ? req.file.filename : req.body.imagenActual
  };
};
```

```

connection.query("UPDATE productos SET ? WHERE id = ?", [productoActualizado,
id], (err, result) => {
  if (err) throw err;
  res.send('Producto actualizado exitosamente');
});
};

```



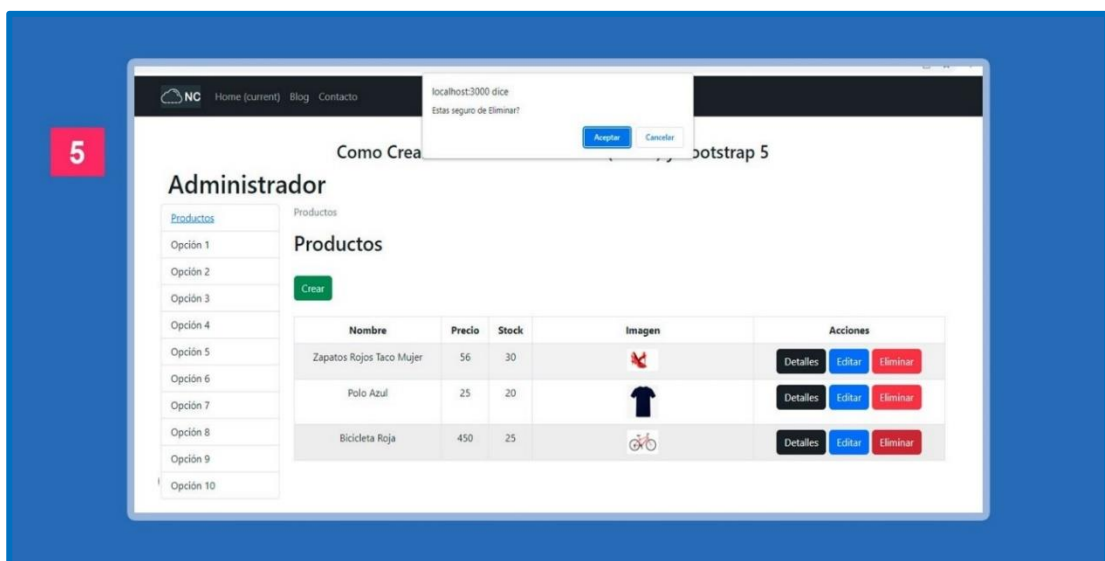
Aplicación puede interactuar [ACTUALIZAR] con la base de datos inventario, tienda.

- Operación de Eliminación (Delete):

```

exports.eliminarProducto = (req, res) => {
  const id = req.params.id;
  connection.query("DELETE FROM productos WHERE id = ?", [id], (err, result) => {
    if (err) throw err;
    res.send('Producto eliminado exitosamente');
  });
};

```



Aplicación puede interactuar [ELIMINAR] con la base de datos inventario, tienda.

Paso 4: Configuración de las Rutas.

- Crea una carpeta routes y un archivo productos.js para definir las rutas:

```
const express = require('express');
const router = express.Router();
const productosController = require('../controllers/productosController');

router.get('/productos', productosController.listarProductos);
router.post('/productos', productosController.crearProducto);
router.get('/productos/:id', productosController.obtenerProductoPorId);
router.put('/productos/:id', productosController.actualizarProducto);
router.delete('/productos/:id', productosController.eliminarProducto);

module.exports = router;
```

Paso 5: Integración de las Rutas en app.js

```
const productosRoutes = require('./routes/productos');
app.use('/api', productosRoutes);
```

Actividades para el Estudiante

1. ¿Qué operaciones CRUD consideraste más desafiantes y por qué?
2. ¿Cómo validas que las operaciones CRUD funcionen correctamente en el sistema?
3. ¿Cuál fue el aspecto más complicado al crear la funcionalidad de actualización (Update) en el servidor y cómo lo resolviste?
4. ¿Qué decisiones tomaste en cuanto al diseño del sistema y por qué?
5. ¿Qué ajustes o mejoras aplicarías al servidor backend si tuvieras más tiempo o recursos?

CURSO: PIAD-527_FULLSTACK DEVELOPER SOFTWARE

Tarea – HT-03

Usa tecnología frontend con JavaScript.

Operaciones:

1. Define conceptos de web SPA y configura el entorno para React.
2. Crea proyecto con React.
3. Usa componentes, Routing y Navegación en React.
4. Eventos y formularios con React.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de crear una aplicación básica en React, cubriendo conceptos clave como el uso de componentes, la configuración de rutas y el manejo de eventos y formularios. Además, se podrá configurar el entorno, estructurar el proyecto y comprender cómo funcionan las aplicaciones React y las SPA.

Caso Práctico

Una pequeña tienda de artesanías desea tener presencia en línea y permitir que sus clientes visualicen sus productos, agreguen artículos al carrito de compras y realicen pedidos. Quieren una aplicación de una sola página (SPA) que permita una navegación rápida sin recargar la página, mejorando la experiencia del usuario.

Por lo que se requiere que se desarrolle: Define conceptos de web SPA y configura el entorno para React, crea proyecto con React, usa componentes, Routing y Navegación en React y Eventos y formularios con React.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Documentación de React y SPA.	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora/Laptop con GPU	1
Conexión a Internet	1

Herramientas:	
Nombre	Cantidad
Visual Studio Code	1

Desarrollo de la Práctica

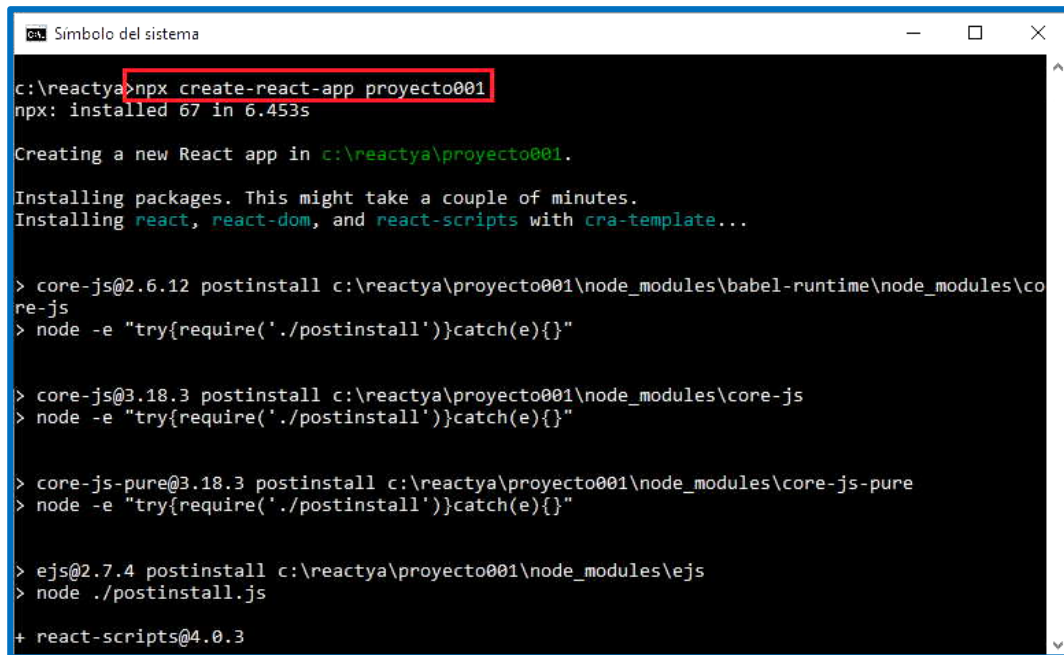
OPERACIÓN 01: Define conceptos de web SPA y configura el entorno para React.

Paso 1: Concepto de SPA:

- **Single Page Application (SPA):** Una SPA (Single Page Application) es una aplicación web que interactúa con el usuario cargando solo una página HTML. Todo el contenido adicional o nuevo se carga dinámicamente mediante JavaScript sin recargar la página completa. Este enfoque mejora la velocidad de navegación y la experiencia del usuario.
- **React:** Es una biblioteca de JavaScript utilizada para construir interfaces de usuario de forma declarativa y eficiente. Al ser una biblioteca basada en componentes, React permite dividir la interfaz en partes reutilizables.

Paso 2: Configuración del Entorno:

- **Requisitos Previos:** Tener instalado [Node.js](#) y un editor de código como [Visual Studio Code](#).



```
c:\reactya>npx create-react-app proyecto001
npx: installed 67 in 6.453s

Creating a new React app in c:\reactya\proyecto001.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

> core-js@2.6.12 postinstall c:\reactya\proyecto001\node_modules\babel-runtime\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"

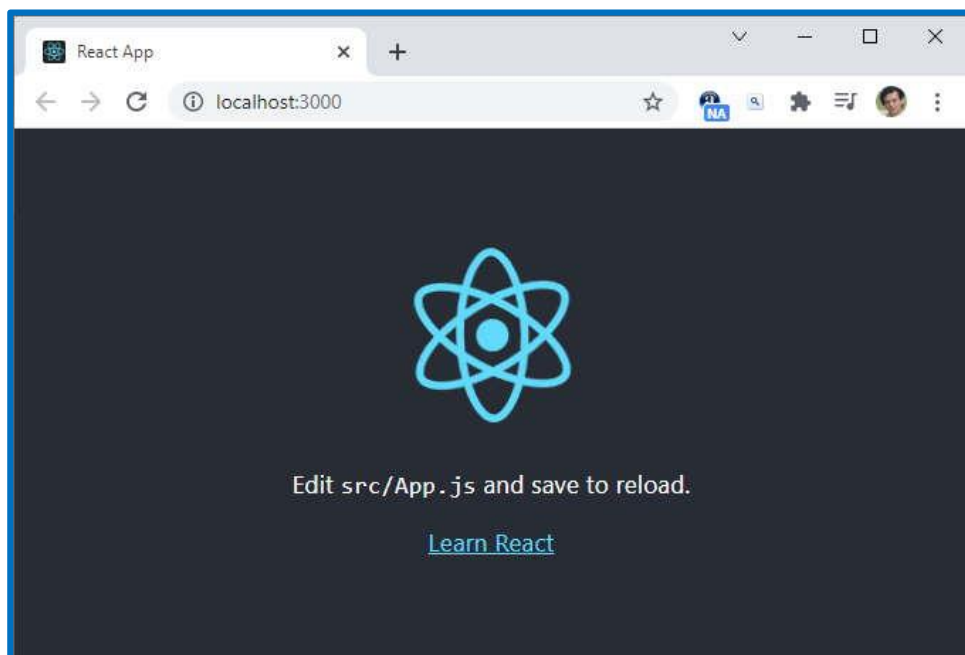
> core-js@3.18.3 postinstall c:\reactya\proyecto001\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"

> core-js-pure@3.18.3 postinstall c:\reactya\proyecto001\node_modules\core-js-pure
> node -e "try{require('./postinstall')}catch(e){}"

> ejs@2.7.4 postinstall c:\reactya\proyecto001\node_modules\ejs
> node ./postinstall.js

+ react-scripts@4.0.3
```

La instalación de create-react-app con npx create-react-app



Ejecución npm start permite visualizar la aplicación en el navegador

Paso 3: Instalación de Create-React-App:

- Abre la terminal y ejecuta:

`npx create-react-app tienda-online`

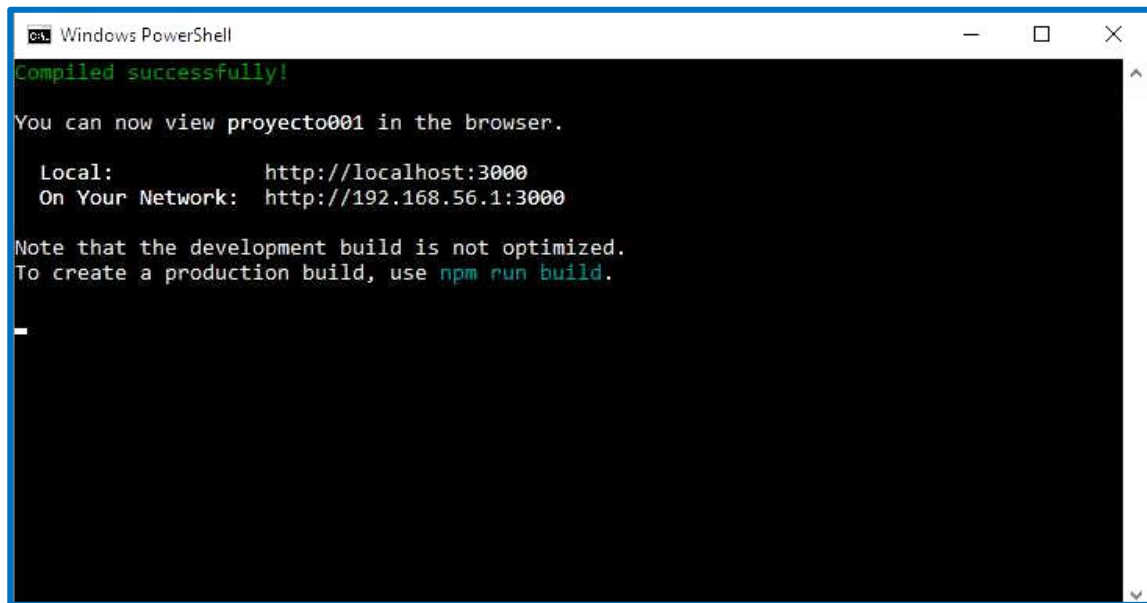
- Una vez creado el proyecto, navega a la carpeta:

`cd tienda-online`

- Para iniciar la aplicación, ejecuta:

`npm start`

- Visualiza la aplicación en el navegador en <http://localhost:3000>.



NO cerrar la consola de Node.js donde está en ejecución la aplicación

OPERACIÓN 02: Crea proyecto con React.

Paso 1: Personalización Inicial.

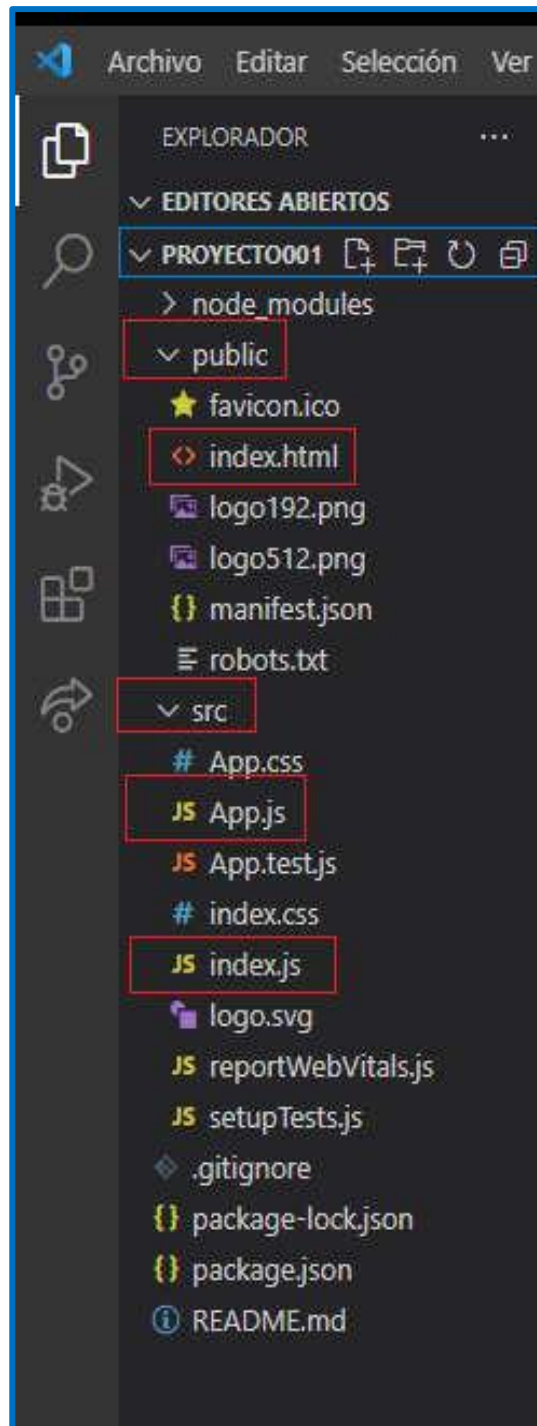
- Abre el archivo App.js y reemplaza el contenido básico por un saludo inicial:

```
function App() {  
  return (  
    <div>  
      <h1>Bienvenidos a la Tienda de Artesanías</h1>  
    </div>  
  );  
}  
export default App;
```

- Guarda y verifica los cambios en el navegador.

Paso 2: Estructura de Carpeta Básica.

- **public/:** Contiene el archivo index.html que sirve como base de la aplicación.



Explorador en Visual Studio Code

- **src/**: Incluye archivos clave como index.js y App.js, donde se desarrollará la lógica y estructura de la aplicación.



Los archivos App.js, index.js y index.html se relacionan.

OPERACIÓN 03: Usa componentes, Routing y Navegación en React.

Paso 1: Creación de Componentes:

- Crea una estructura de carpetas en src/components/ para organizar los componentes.
- Crea los siguientes componentes básicos:

- **Header (src/components/Header.js):**

```
function Header() {  
  return <h1>Tienda de Artesanías</h1>;  
}  
export default Header;
```

- **ProductList (src/components/ProductList.js):**

```
function ProductList() {  
  return (  
    <div>  
      <h2>Lista de Productos</h2>  
      <ul>  
        <li>Producto 1</li>  
        <li>Producto 2</li>  
      </ul>  
    </div>  
  );  
}  
export default ProductList;
```

Paso 2: Configuración de Rutas con React Router:

- Instala React Router:

```
npm install react-router-dom
```

- En App.js, configura el routing básico:

```
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Header from './components/Header';
import ProductList from './components/ProductList';

function App() {
  return (
    <Router>
      <Header />
      <Routes>
        <Route path="/" element={<ProductList />} />
        <Route path="/product/:id" element={<h2>Detalle del Producto</h2>} />
      </Routes>
    </Router>
  );
}

export default App;
```

OPERACIÓN 04: Eventos y formularios con React.

Paso 1: Creación del Componente de Carrito (src/components/Cart.js):

```
import { useState } from 'react';

function Cart() {
  const [cart, setCart] = useState([]);

  const handleAddToCart = (product) => {
    setCart([...cart, product]);
  };

  return (
    <div>
      <h2>Carrito de Compras</h2>
      <button onClick={() => handleAddToCart('Producto 1')}>Agregar Producto 1</button>
      <ul>
        {cart.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
    </div>
  );
}
```

```
}  
export default Cart;
```

Paso 2: Formulario de Contacto (src/components/ContactForm.js):

```
import { useState } from 'react';  
  
function ContactForm() {  
  const [formData, setFormData] = useState({ name: "", email: "" });  
  
  const handleChange = (e) => {  
    const { name, value } = e.target;  
    setFormData({ ...formData, [name]: value });  
  };  
  
  const handleSubmit = (e) => {  
    e.preventDefault();  
    alert(` Gracias por contactarnos, ${formData.name}!`);  
  };  
  
  return (  
    <form onSubmit={handleSubmit}>  
      <input type="text" name="name" placeholder="Nombre" onChange={handleChange}/>  
      <input type="email" name="email" placeholder="Email" onChange={handleChange}/>  
      <button type="submit">Enviar</button>  
    </form>  
  );  
}
```

export default ContactForm;

Actividades para el Estudiante

1. ¿Por qué es importante tener un entorno de desarrollo configurado correctamente?
2. ¿Qué diferencia hay entre componentes funcionales y componentes de clase en React?
3. ¿Qué técnicas puedes usar para administrar el estado en React?
4. ¿Cómo validarías los datos ingresados en un formulario de React?
5. ¿Qué aprendiste sobre la estructura de un proyecto en React que aplicarías en proyectos futuros?

CURSO: PIAD-527_FULLSTACK DEVELOPER SOFTWARE

Tarea – HT-04

Diseña y crea servicios API RESTful.

Operaciones:

1. Define los fundamentos de REST y SOAP.
2. Crea API REST con Node JS.
3. Define los fundamentos de contenedores y Docker.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de crear una API REST que permita la consulta y administración de productos en un entorno controlado y contenedorizado.

Caso Práctico

La empresa "AtenciónTotal" se dedica a la prestación de servicios de soporte al cliente, gestionando una base de datos de clientes y contactos en diferentes plataformas. Sin embargo, su sistema de administración de contactos está obsoleto y no permite integración con nuevas aplicaciones. Se requiere modernizar este sistema utilizando una API REST para interactuar con una base de datos de contactos y desplegar el sistema en contenedores Docker para facilitar su integración y escalabilidad.

Por lo que se requiere que se desarrolle: •Define los fundamentos de REST y SOAP, crea API REST con Node JS y define los fundamentos de contenedores y Docker.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Documentación de Dockerfile	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora/Laptop con GPU	1
Conexión a Internet	1

Herramientas:	
Nombre	Cantidad
Visual Studio Code o Editor de Código	1

Desarrollo de la Práctica

OPERACIÓN 01: Define los fundamentos de REST y SOAP.

- **REST (Representational State Transfer):** REST es un estilo arquitectónico basado en recursos y utiliza métodos HTTP como GET, POST, PUT, y DELETE para interactuar con estos recursos. Se enfoca en una comunicación sin estado y es ampliamente utilizado en aplicaciones web.
- **SOAP (Simple Object Access Protocol):** SOAP es un protocolo de mensajería estándar que depende de XML para estructurar la información. Requiere una definición formal en formato WSDL (Web Services Description Language) y se usa frecuentemente en sistemas de integración complejos.

OPERACIÓN 02: Crea API REST con Node JS.

Paso 1: Instalación de Node.js y Configuración del Proyecto.

`mkdir gestion-contactos`

```
cd gestion-contactos
npm init -y
npm install express body-parser mongoose
```

Paso 2: Estructura de Archivos del Proyecto.

- Crear las carpetas y archivos:

```
gestion-contactos/
├── app.js
├── models/
│   └── Contact.js
├── routes/
│   └── contactRoutes.js
└── package.json
```

Paso 3: Configuración de Express y Conexión a la Base de Datos (MongoDB).

- **app.js:**

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const contactRoutes = require('./routes/contactRoutes');

const app = express();
app.use(bodyParser.json());

mongoose.connect('mongodb://localhost/contactsDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

app.use('/api/contacts', contactRoutes);

app.listen(3000, () => console.log('API REST corriendo en http://localhost:3000'));
```

Paso 4: Definir el Modelo Contact en models/Contact.js.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const ContactSchema = new Schema({
  name: String,
  email: String,
```

```
    phone: String  
  });
```

```
module.exports = mongoose.model('Contact', ContactSchema);
```

Paso 5: Implementar Rutas de la API en routes/contactRoutes.js:

```
const express = require('express');  
const router = express.Router();  
const Contact = require('../models/Contact');
```

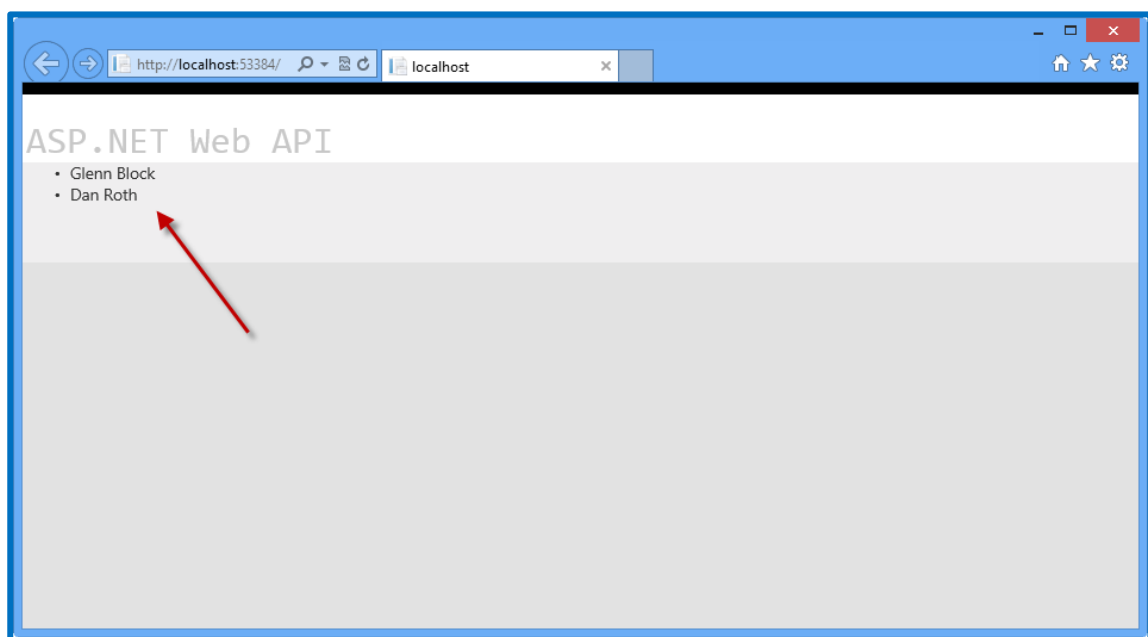
```
// Obtener todos los contactos  
router.get('/', async (req, res) => {  
  const contacts = await Contact.find();  
  res.json(contacts);  
});
```

```
// Crear un nuevo contacto  
router.post('/', async (req, res) => {  
  const newContact = new Contact(req.body);  
  await newContact.save();  
  res.status(201).json(newContact);  
});
```

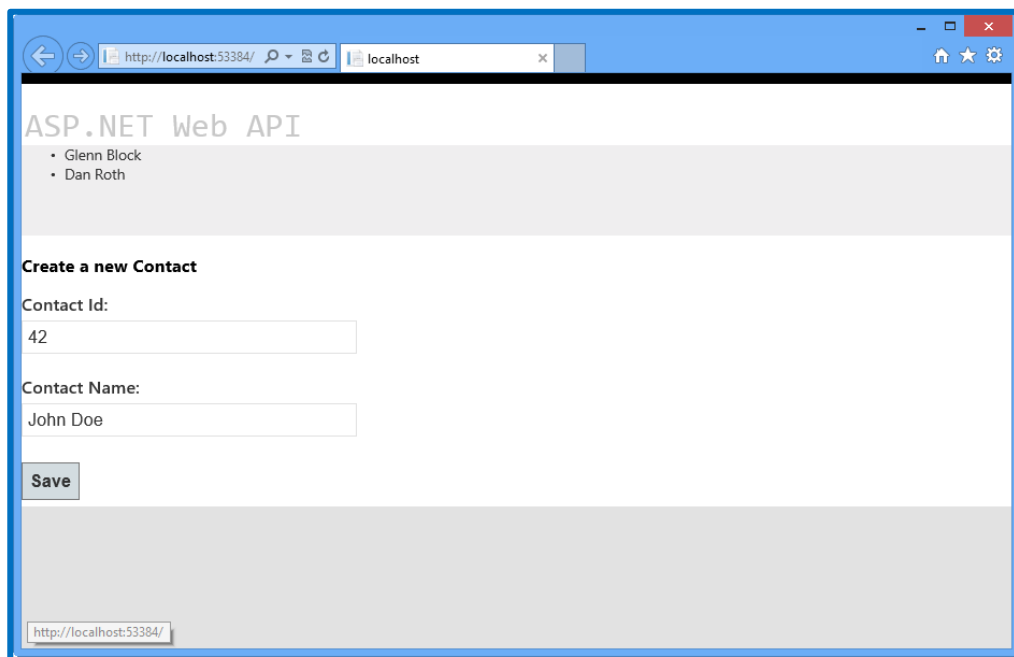
```
module.exports = router;
```

Paso 6: Prueba de la API.

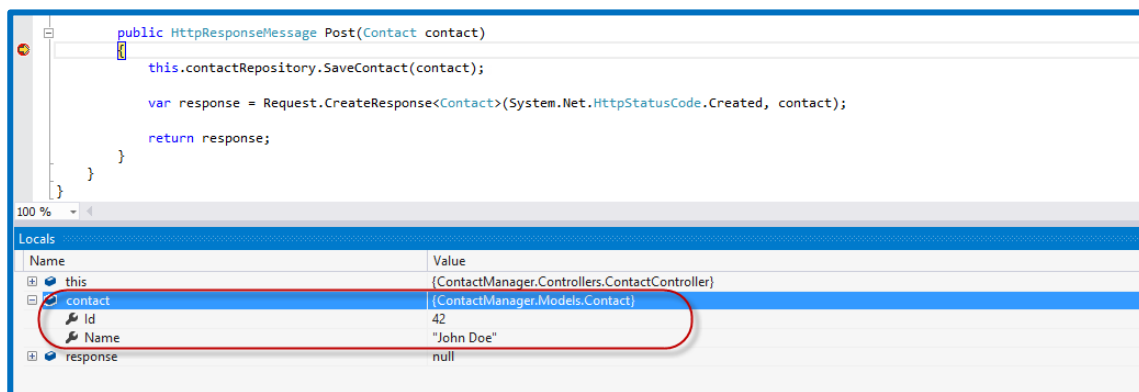
- Ejecutar la API: node app.js.
- Usar herramientas como Postman o curl para probar las solicitudes GET y POST.



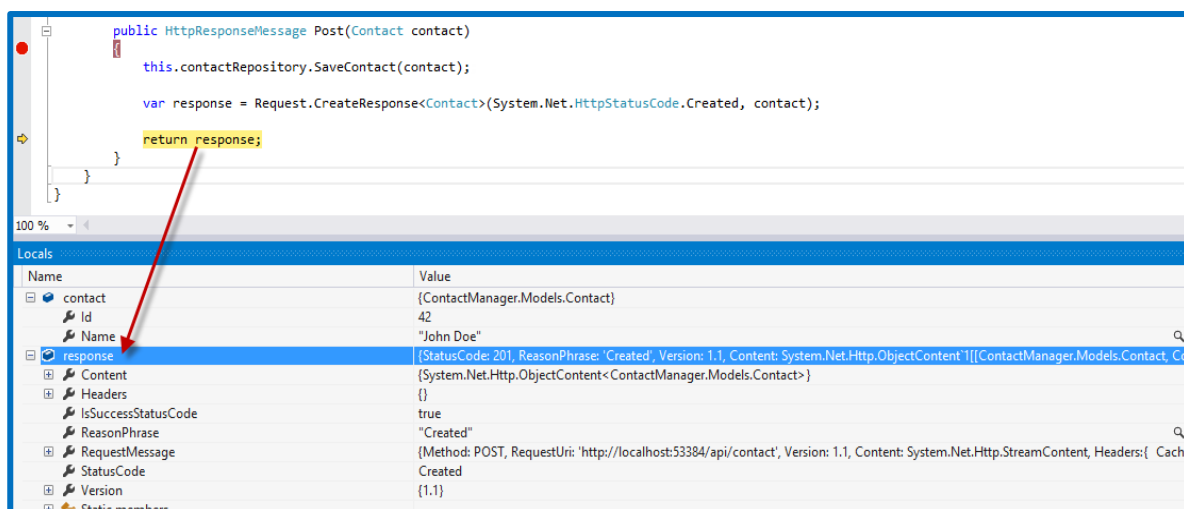
Resultados de la llamada a la API mostrados en el navegador



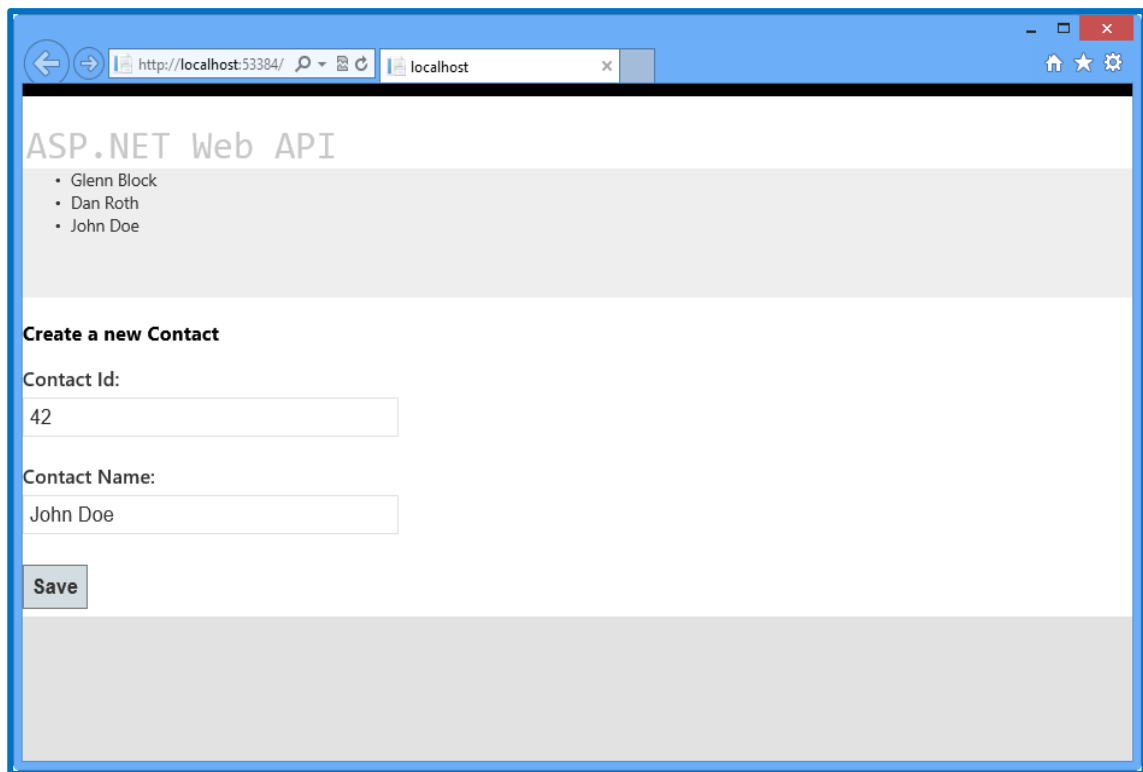
El documento HTML del cliente cargado en el navegador



El objeto Contacto que se envía a la Web API desde el cliente



La respuesta tras la creación en el depurador



El navegador refleja la creación exitosa de la nueva instancia de contacto

OPERACIÓN 03: Define los fundamentos de contenedores y Docker.

- **Contenedores:** Los contenedores son entornos aislados que empaquetan una aplicación junto con sus dependencias para asegurar que funcione de manera uniforme en cualquier entorno.
- **Docker:** Docker es una plataforma de contenedorización que permite a los desarrolladores crear, desplegar y ejecutar aplicaciones en contenedores.

Paso 1: Instalación de Docker.

Descargar e instalar Docker desde la [página oficial](#).

Paso 2: Crear Dockerfile en el Proyecto.

```
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "app.js"]
```

Paso 3: Construcción de la Imagen Docker.

- En la raíz del proyecto:

```
docker build -t gestion-contactos-api .
```

Paso 4: Ejecutar el Contenedor Docker:

- Para iniciar el contenedor:

```
docker run -p 3000:3000 gestion-contactos-api
```

Paso 5: Pruebas de la API en el Contenedor.

Abrir <http://localhost:3000/api/contacts> en el navegador o en Postman para verificar que la API funciona dentro del contenedor Docker.

Actividades para el Estudiante

1. ¿Qué diferencias existen entre REST y SOAP, y por qué fue mejor utilizar REST en este caso?
2. ¿Cómo resolviste cualquier error que surgió durante la instalación de los paquetes de Node.js?
3. ¿Cómo manejaste los errores en las rutas de la API para mejorar la experiencia del usuario?
4. ¿Qué pasos consideraste importantes al estructurar los archivos del proyecto?
5. ¿Qué has aprendido sobre el ciclo de vida del desarrollo de una API REST y cómo se aplica a otros proyectos futuros?

CURSO: PIAD-528_TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING

Tarea – HT-02

Crea y entrena modelos ML

Operaciones:

1. Crea y entrena una red neuronal con Python y TensorFlow.
2. Implementa y gráfica regresión lineal simple.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de desarrollar una red neuronal que clasifique prendas de ropa y probar la relación entre dos variables a través de una regresión lineal simple. Esta guía permitirá que otros equipos dentro de la empresa puedan implementar y replicar estos procesos en sus almacenes para mejorar la eficiencia operativa.

Caso Práctico

Una empresa de moda está interesada en implementar un sistema automatizado de clasificación de ropa en su almacén. Este sistema debe ser capaz de identificar automáticamente diferentes tipos de prendas, como camisetas, pantalones y abrigos, basándose en imágenes. La clasificación rápida y precisa de los artículos ayuda a organizar el inventario de manera eficiente y mejora los tiempos de respuesta para la reposición de productos en las tiendas. El equipo de desarrollo ha decidido utilizar redes neuronales y técnicas de machine learning para esta tarea, basándose en el conjunto de datos Fashion MNIST.

Por lo que se requiere que se desarrolle: Crea y entrena una red neuronal con Python y TensorFlow e implementa y gráfica regresión lineal simple.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Datos de Ventas y Temperatura	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora/Laptop con GPU	1
Conexión a Internet	1

Herramientas:	
Nombre	Cantidad
Visual Studio Code	1

Desarrollo de la Práctica

OPERACIÓN 01: Crea y entrena una red neuronal con Python y TensorFlow.

Paso 1: Instalación de TensorFlow

Para trabajar con redes neuronales, el documento sugiere instalar TensorFlow, una biblioteca de código abierto desarrollada por Google. Si tienes un entorno Python instalado, puedes ejecutar el siguiente comando en tu terminal:

```
pip install --upgrade tensorflow
```

Paso 2: Cargar y Explorar el Conjunto de Datos Fashion MNIST

El dataset Fashion MNIST, incluido en Keras, es una colección de imágenes de ropa divididas en categorías como camisetas, pantalones y abrigos.



[Colección de imágenes de ropa.](#)

Puedes cargar y explorar el conjunto de datos con este código:

```
from tensorflow.keras.datasets import fashion_mnist
import matplotlib.pyplot as plt
import numpy as np

# Cargar el dataset Fashion MNIST
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# Explorar el dataset
labels = ["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker",
          "Bag", "Ankle boot"]
plt.figure(figsize=(14, 8))
for i in range(20):
    plt.subplot(4, 5, i+1)
    plt.imshow(X_train[i], cmap="binary")
    plt.title(labels[y_train[i]])
    plt.axis("off")
plt.show()
```

Paso 3: Normalización de Datos

Para mejorar el rendimiento de la red, es importante normalizar los datos, escalando los valores de píxeles (0-255) a un rango entre 0 y 1:

```
# Normalizar los datos
X_train, X_test = X_train / 255.0, X_test / 255.0
```

Paso 4: Definición de la Arquitectura de la Red Neuronal

Se usará una red neuronal con tres capas:

- Una capa Flatten que convierte la imagen de 28x28 píxeles en un vector de 784 entradas.
- Una capa oculta Dense con 128 neuronas y la función de activación relu.
- Una capa de salida Dense con 10 neuronas y la función de activación softmax para clasificar cada prenda en una de las 10 categorías.

```
from tensorflow import keras

# Definimos la arquitectura de la red neuronal
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

Paso 5: Compilar el Modelo

Para entrenar la red, es necesario compilar el modelo especificando la función de pérdida, el optimizador y las métricas de evaluación:

```
# Compilar el modelo
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Paso 6: Entrenar el Modelo

Configura los parámetros de entrenamiento. Entrenaremos el modelo por 10 épocas usando el conjunto de datos de entrenamiento y validación:

```
# Entrenar el modelo
model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
```

Paso 7: Evaluar el Modelo

Finalmente, evalúa el modelo en el conjunto de prueba para obtener la precisión del modelo:

```
# Evaluar el modelo
test_loss, test_acc = model.evaluate(X_test, y_test)
print("\nPrecisión en el conjunto de prueba:", test_acc)
```

OPERACIÓN 02: Implementa y gráfica regresión lineal simple.

Para evaluar cómo las ventas de una tienda de ropa pueden depender de la temperatura (ejemplo ficticio), se implementará una regresión lineal simple para ver si existe una correlación entre estas dos variables.

```
# Importamos las bibliotecas necesarias
import numpy as np
import matplotlib.pyplot as plt
```

```

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Datos de ejemplo (ficticios)
temperatura = np.array([15, 16, 18, 20, 21, 23, 25, 27, 30, 32])
ventas = np.array([500, 520, 560, 580, 600, 640, 680, 700, 760, 800])

# Dividimos los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(temperatura.reshape(-1, 1), ventas,
test_size=0.2, random_state=42)

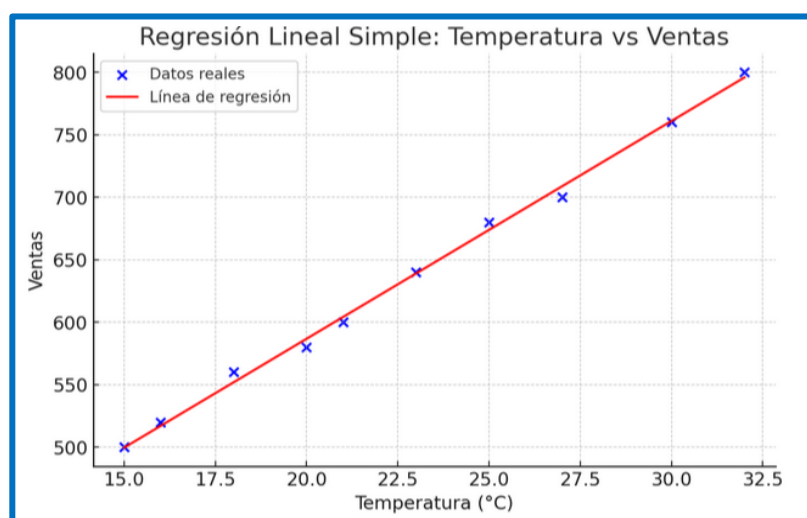
# Creamos y entrenamos el modelo de regresión lineal
model = LinearRegression()
model.fit(X_train, y_train)

# Predicciones y gráfica de resultados
plt.scatter(temperatura, ventas, color="blue", label="Datos reales")
plt.plot(temperatura, model.predict(temperatura.reshape(-1, 1)), color="red", label="Línea de regresión")
plt.xlabel("Temperatura (°C)")
plt.ylabel("Ventas")
plt.legend()
plt.show()

# Evaluación del modelo
r_sq = model.score(X_test, y_test)
print('Coeficiente de determinación:', r_sq)

```

A continuación, mostraremos la gráfica de la regresión lineal simple entre la temperatura y las ventas.



Regresión lineal simple.

Donde la línea roja representa el modelo de regresión lineal ajustado a los datos reales, que están indicados con puntos azules.

Actividades para el Estudiante

1. ¿Qué función cumple la capa Flatten en el modelo de red neuronal?
2. ¿Qué descubriste sobre la relación entre temperatura y ventas en el ejercicio de regresión?
3. ¿Qué mejoras le harías a la red neuronal para aumentar la precisión?
4. ¿Cómo puede mejorar la precisión de la clasificación al aumentar el número de épocas?
5. ¿Qué otras aplicaciones prácticas puedes imaginar para esta red neuronal en la industria de la moda?

CURSO: PIAD-528_TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING

Tarea – HT-03

Exporta e integra modelos de Machine Learning

Operaciones:

1. Conoce y usa “Teachable Machine”.
2. Exporta Modelo de Machine Learning con Python.
3. Usa modelo ML exportado y lee con TensorflowJS para web.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de desarrollar un modelo de *Machine Learning* accesible con *Teachable Machine* para clasificar imágenes de aves en categorías, exportar el modelo para aplicaciones web y desplegarlo utilizando *TensorFlowJS*.

Caso Práctico

Un equipo de biólogos y analistas de datos trabaja en un proyecto de conservación de especies en un área protegida. El objetivo es identificar especies de aves endémicas a partir de imágenes de cámaras trampa para monitorear sus poblaciones y mejorar la toma de decisiones en cuanto a conservación. Dado que el equipo no cuenta con especialistas en *Machine Learning*, optan por usar herramientas accesibles para crear un modelo de clasificación de imágenes.

Por lo que se requiere que se desarrolle: Conoce y usa “Teachable Machine”, exporta Modelo de Machine Learning con Python y usa modelo ML exportado y lee con TensorflowJS para web.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Imágenes de especies de aves.	1
Archivos de documentación.	1

Instrumentos y Equipos:	
Nombre	Cantidad
Computadora/Laptop con 8gb de ram.	1
Cámara de fotos o cámara de celular.	1

Herramientas:	
Nombre	Cantidad
Teachable Machine.	1
TensorFlow y TensorFlowJS.	1
Visual Studio Code.	1

Desarrollo de la Práctica

OPERACIÓN 01: Conoce y usa "Teachable Machine".

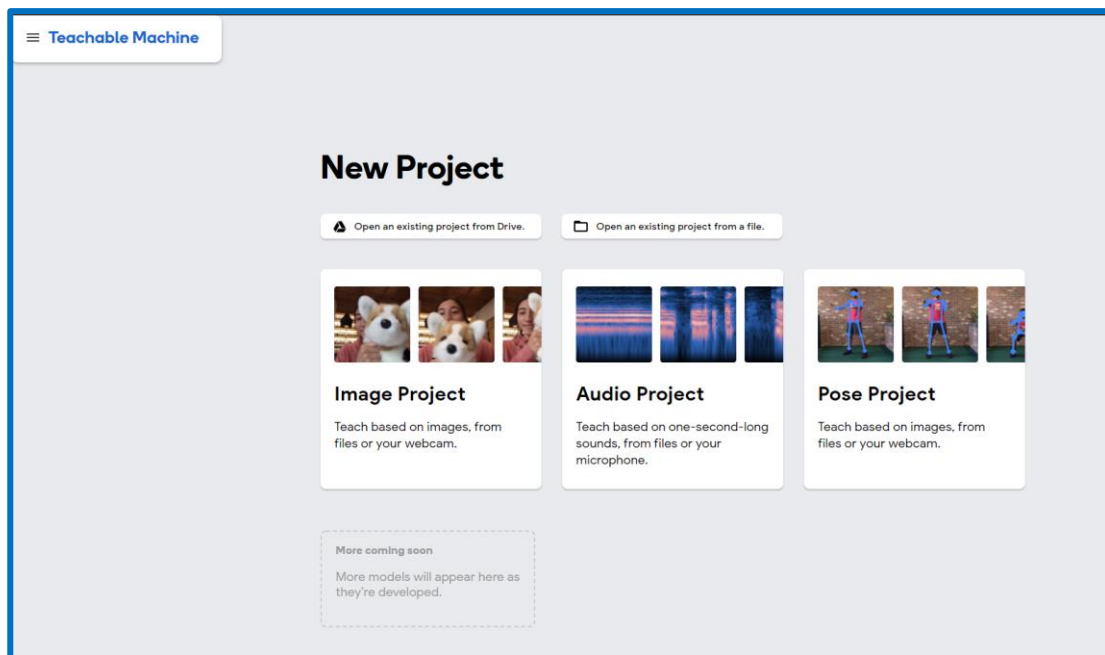
Paso 1: Acceder a [Teachable Machine](#).

Paso 2:

Seleccionar "Image Project" y elegir "Standard Image Model" para comenzar.

Paso 3: Preparación de Imágenes:

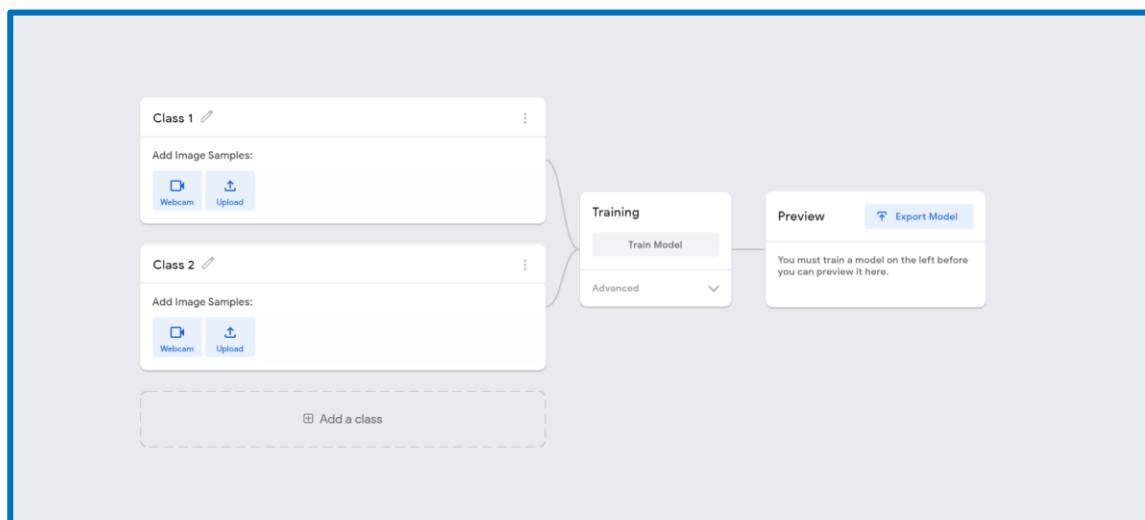
- Recolectar imágenes de las especies de aves que se quieren clasificar. Usar un mínimo de 100 imágenes por especie para mayor precisión.
- Cargar las imágenes en el proyecto, organizándolas por carpetas o directamente en la plataforma, una carpeta para cada especie.



[Teachable Machine webpage](#)

Paso 4: Entrenamiento del Modelo:

- Iniciar el entrenamiento en Teachable Machine después de cargar todas las imágenes y ajustar los parámetros básicos según el rendimiento deseado (ej., número de épocas).



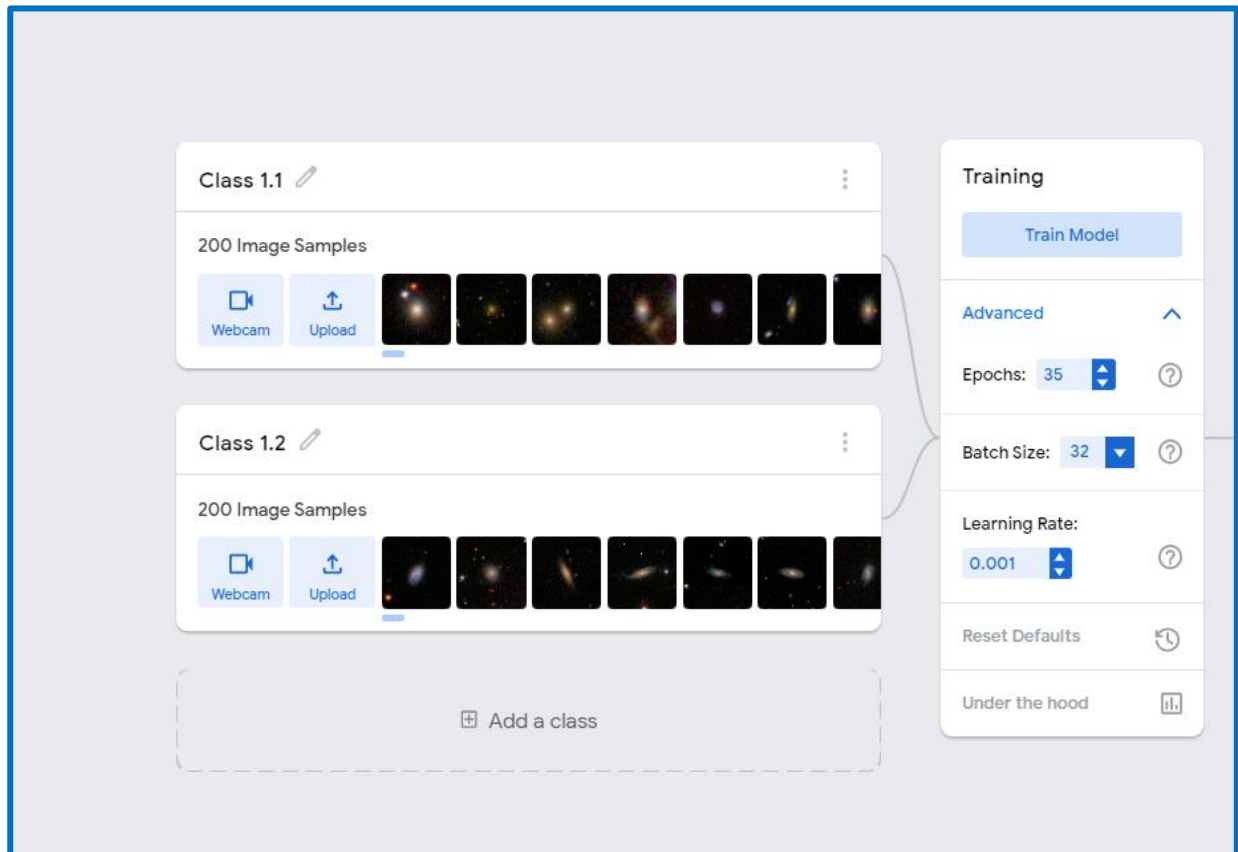
[Vista del comienzo de un proyecto de clasificación de imágenes.](#)

Paso 5:

Revisar los resultados del entrenamiento, asegurándose de que el modelo muestra buenos niveles de precisión para cada clase de aves.

OPERACIÓN 02: Exporta Modelo de Machine Learning con Python.

Paso 1: Exportar el modelo de Teachable Machine en formato de *TensorFlow* o *TensorFlow Lite*.



Una vez subidas las imágenes para comenzar el entrenamiento.

Paso 2: Descargar el modelo y cargarlo en un entorno Python para preprocesamiento adicional si es necesario. A continuación, se muestra un ejemplo para cargar el modelo en Python.

Código para cargar el modelo:

```
import tensorflow as tf

# Cargar el modelo exportado de Teachable Machine
model = tf.keras.models.load_model('/ruta/del/modelo/teachable_machine_model.h5')

# Proceso de prueba con una imagen
import numpy as np
from tensorflow.keras.preprocessing import image

def predict_species(image_path):
    img = image.load_img(image_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) # Aumentar dimensión para predicción
    img_array /= 255.0 # Normalizar
```

```
prediction = model.predict(img_array)
species = np.argmax(prediction) # Obtener clase predicha
return species
```

```
# Ejemplo de predicción
image_path = '/ruta/a/una/imagen.jpg'
print(f"La especie predicha es: {predict_species(image_path)}")
```

OPERACIÓN 03: Usa modelo ML exportado y lee con TensorFlowJS para web.

Paso 1: Exportar el modelo desde Teachable Machine en el formato de *TensorFlow.js*. Teachable Machine proporciona una opción para exportar directamente al formato de *TensorFlowJS*.

Paso 2: Crear un archivo HTML simple con *JavaScript* y *TensorFlowJS* para cargar y utilizar el modelo.

Código HTML y JavaScript para el sitio web:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Clasificación de Especies de Aves</title>
  <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
  <script
src="https://teachablemachine.withgoogle.com/models/tu_modelo_ya_exportado/model.j
son"></script>
</head>
<body>
  <h1>Clasificación de Especies de Aves</h1>
  <input type="file" id="file-input" onchange="predictImage()" />
  <p id="prediction-result"></p>

  <script>
    let model;

    async function loadModel() {
      model = await tf.loadLayersModel('model.json'); // Cargar el modelo
    }

    async function predictImage() {
```

```

const fileInput = document.getElementById('file-input');
const img = document.createElement('img');
img.src = URL.createObjectURL(fileInput.files[0]);
img.onload = async () => {
    const imgTensor = tf.browser.fromPixels(img).resizeNearestNeighbor([224,
224]).toFloat().expandDims();
    const prediction = await model.predict(imgTensor).data();
    const classIndex = prediction.indexOf(Math.max(...prediction));
    document.getElementById('prediction-result').innerText = `Predicción: Especie
${classIndex}`;
};
}

window.onload = loadModel;
</script>
</body>
</html>

```

Actividades para el Estudiante

1. ¿Cuáles fueron los desafíos más significativos al usar *Teachable Machine*?
2. ¿Qué ventajas y limitaciones encontraste en *Teachable Machine* para este tipo de proyecto?
3. ¿Qué cambios realizarías en el flujo de trabajo para mejorar la precisión del modelo?
4. ¿Qué limitaciones podrías encontrar al usar este modelo para clasificar especies desconocidas?
5. ¿Qué habilidades consideras que has desarrollado a través de este caso práctico?

CURSO: PIAD-528_TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING

Tarea – HT-04

Usa herramientas de IA para integrarlo al desarrollo de software.

Operaciones:

1. Conoce y crea aplicaciones Python usando OpenCV.
2. Conoce la utilidad de Tensorflow Lite.
3. Conoce la utilidad de TinyML.

Objetivo de la Tarea

Al concluir la tarea el participante estará en condiciones de crear una aplicación en Python que capture y visualice datos en tiempo real, configure notificaciones y aplique modelos ligeros de Machine Learning para predicciones rápidas en dispositivos de baja potencia.

Caso Práctico

Se plantea la implementación de un sistema de monitoreo de temperatura para un almacén de productos perecederos. El objetivo es predecir en tiempo real si la temperatura en el ambiente supera un límite, permitiendo a los supervisores del almacén tomar decisiones inmediatas para ajustar el clima y asegurar la conservación óptima de los productos. El sistema utiliza un sensor de temperatura conectado a un microcontrolador, aplicando modelos de Machine Learning (ML) ligeros que pueden ejecutarse en dispositivos de baja potencia, como un microcontrolador.

Por lo que se requiere que se desarrolle: Conoce y crea aplicaciones Python usando OpenCV, conoce la utilidad de Tensorflow Lite y conoce la utilidad de TinyML.

Materiales/ Instrumentos/ Equipos/Herramientas/ Reactivos/ Insumos/ Colorantes.

Las siguientes listas son de referencia.

El instructor puede variar los requerimientos, con fin de desarrollar la tarea.

Materiales:	
Nombre	Cantidad
Sensor de temperatura	1
Cableado para conexiones	2

Instrumentos y Equipos:	
Nombre	Cantidad
Microcontrolador (Arduino o ESP32).	1
Computadora/Laptop con Python y TensorFlow instalado	1

Herramientas:	
Nombre	Cantidad
IDE Arduino	1
Software TensorFlow Lite Converter	1

Desarrollo de la Práctica

OPERACIÓN 01: Conoce y crea aplicaciones Python usando OpenCV.

Paso 1: Instalar OpenCV y Librerías Necesarias

```
pip install opencv-python pandas matplotlib
```

Paso 2: Captura de Temperatura

Simular datos de temperatura con valores aleatorios para esta práctica. En un caso real, los datos vendrían de un sensor.

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

```
# Generamos datos de temperatura aleatorios para simular la entrada del sensor
temperatura = np.random.normal(loc=20, scale=5, size=100)
```



```
# Visualización en gráfico
plt.plot(temperatura, label='Temperatura (°C)')
plt.xlabel('Tiempo')
plt.ylabel('Temperatura')
plt.legend()
plt.show()
```

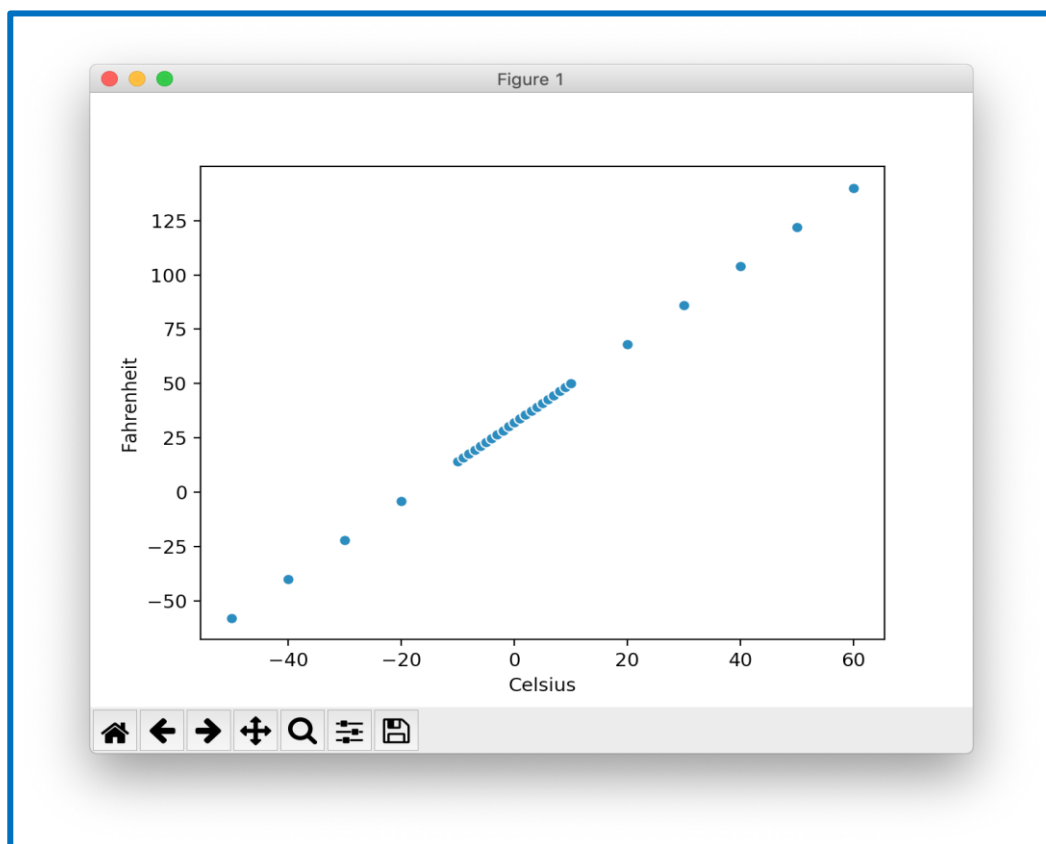
Paso 3: Mostrar Alertas en Tiempo Real

Visualizar alertas en OpenCV cuando la temperatura exceda los límites.

```
# Visualización usando OpenCV
for temp in temperatura:
    alert_msg = "Alerta: Temperatura Excesiva!" if temp > 25 else "Temperatura Normal"
    print(f"Temperatura actual: {temp}°C - {alert_msg}")
```

```
# Visualización usando OpenCV
for temp in temperatura:
    alert_msg = "Alerta: Temperatura Excesiva!" if temp > 25 else "Temperatura Normal"
    print(f"Temperatura actual: {temp}°C - {alert_msg}")
```

Paso 4:



[Visualización de gráfico de temperatura, usando scatterplot.](#)

OPERACIÓN 02: Conoce la utilidad de Tensorflow Lite.

Paso 1: Configurar TensorFlow Lite

Instalar TensorFlow:

```
pip install tensorflow
```

Paso 2: Entrenamiento del Modelo

Entrenar el modelo en una computadora antes de convertirlo a TensorFlow Lite.

```
import tensorflow as tf
import pandas as pd

# Crear datos de ejemplo
datos = {'Celsius': np.arange(0, 100, 10), 'Fahrenheit': np.arange(32, 212, 18)}
df = pd.DataFrame(datos)

X_train = df['Celsius']
y_train = df['Fahrenheit']

# Crear el modelo
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(units=1, input_shape=[1]))

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=500)
```

Paso 3: Convertir a TensorFlow Lite

Guardar y convertir el modelo para TensorFlow Lite.

```
# Guardar el modelo
model.save("modelo_temperatura.h5")

# Convertir a TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Guardar el modelo convertido
with open('modelo_temperatura.tflite', 'wb') as f:
    f.write(tflite_model)
```

Paso 4:

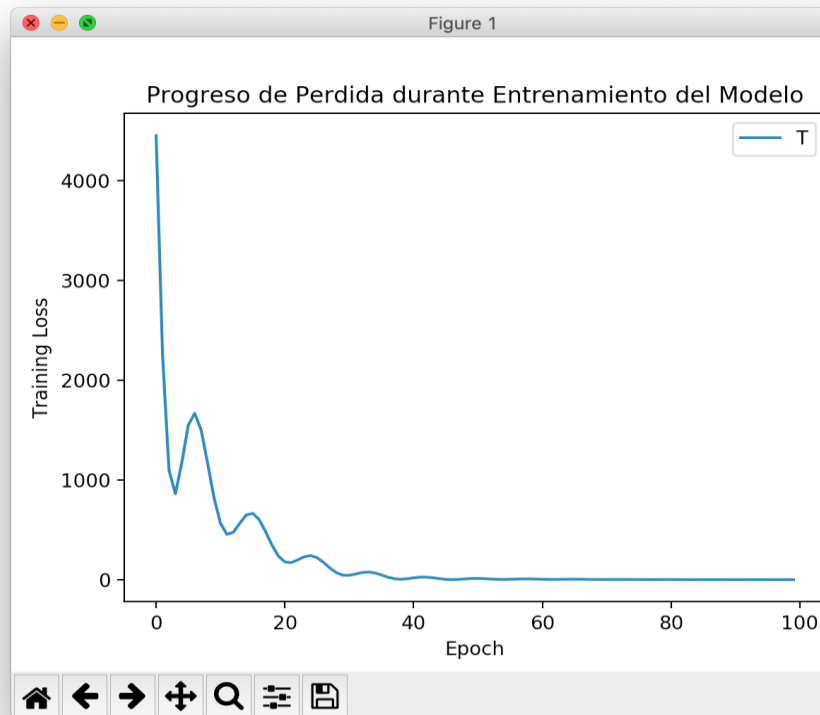


Gráfico de pérdida de entrenamiento

```
Epoch 92/100
1/1 [=====] - 0s 255us/step - loss: 0.0592
Epoch 93/100
1/1 [=====] - 0s 259us/step - loss: 0.0439
Epoch 94/100
1/1 [=====] - 0s 1ms/step - loss: 0.0282
Epoch 95/100
1/1 [=====] - 0s 289us/step - loss: 0.0383
Epoch 96/100
1/1 [=====] - 0s 265us/step - loss: 0.0585
Epoch 97/100
1/1 [=====] - 0s 262us/step - loss: 0.0603
Epoch 98/100
1/1 [=====] - 0s 252us/step - loss: 0.0461
Epoch 99/100
1/1 [=====] - 0s 248us/step - loss: 0.0400
Epoch 100/100
1/1 [=====] - 0s 249us/step - loss: 0.0490
Evaluando el modelo entrenado
Keys:
dict_keys(['loss'])
Temperatura de Prediccion: [[31.801525]]
Temperatura de Ecuacion: 32.0
gibgarcia@MacBook-Pro-de-Gibran-2 Proyecto 1 %
```

Ejemplo de salida predicha y comparación con valores reales.

OPERACIÓN 03: Conoce la utilidad de TinyML.

Paso 1: Configurar TinyML

Usar TensorFlow Lite para microcontroladores y transferir el modelo a un microcontrolador como Arduino.

Paso 2: Desplegar el Modelo en el Microcontrolador

Utilizar el archivo modelo_temperatura.tflite en el microcontrolador para leer datos de temperatura y realizar predicciones en tiempo real.

Paso 3: Ejemplo de Configuración en Arduino

Conectar el sensor de temperatura, cargar el modelo y crear alertas cuando la temperatura sea alta.

Actividades para el Estudiante

1. ¿Qué papel juega OpenCV en la visualización de los datos de temperatura?
2. ¿Qué diferencia existe entre el modelo original de TensorFlow y el convertido para TinyML?
3. ¿Qué acciones se toman cuando la temperatura sobrepasa los niveles de seguridad?
4. ¿Cómo puede mejorar el modelo de predicción con más datos?
5. ¿Cómo evalúas el desempeño de este sistema en un contexto real de almacenamiento?



RDA
RECURSO DIDÁCTICO PARA EL APRENDIZAJE