

## Contenido

MANUAL TÉCNICO Proyecto Persona .....	2
1. Descripción General.....	2
2. Arquitectura del Proyecto.....	3
3. Estructura del Proyecto .....	3
4. Documentación de Clases .....	4
Conexion.java.....	4
PersonaCRUD.java .....	5
App.java .....	6
Persona.java .....	7
PersonaService.java .....	8
ActualizarPersona.java .....	9
IngresarPersona.java .....	10
MenuPrincipal.java .....	11
MostrarPersona.java .....	12
Principal.java .....	12
5.Tecnologías utilizadas.....	13
6.Instalación y configuración .....	14
7.Mantenimiento .....	15
8.Recomendaciones .....	16

# MANUAL TÉCNICO

## Proyecto Persona

### 1. Descripción General

#### Objetivo del proyecto

El proyecto *Persona* tiene como propósito ofrecer una herramienta simple y accesible para gestionar información de personas. La idea es que cualquier usuario, incluso sin conocimientos técnicos, pueda registrar, consultar, actualizar o eliminar datos desde una interfaz clara y amigable. Todo esto se realiza conectándose a Firebase, una base de datos en la nube que permite almacenar la información de forma segura y sin necesidad de instalar un servidor local.

#### Qué problema resuelve?

Muchas aplicaciones pequeñas o educativas necesitan trabajar con datos, pero configurar una base de datos local puede ser complicado. Este sistema elimina esa barrera al usar Firebase, que se encarga de sincronizar y guardar la información automáticamente. Gracias a esto, el proyecto permite centrarse únicamente en la funcionalidad sin preocuparse por la infraestructura.

#### Alcance y funcionalidades principales

El sistema cubre las funciones esenciales para manejar información básica de personas. Entre las más importantes se encuentran:

- Registrar nuevas personas desde un formulario.
- Consultar la lista completa de personas guardadas.
- Editar la información de un registro existente.
- Eliminar datos que ya no se necesiten.
- Conectarse automáticamente a Firebase para leer y escribir en tiempo real.

Aunque es un sistema sencillo, está organizado de forma modular, lo que facilita añadir nuevas características más adelante.

## 2. Arquitectura del Proyecto

El proyecto sigue una arquitectura modular dividida en capas:

- **Capa de Conexión:** Manejo de Firebase.
- **Capa CRUD:** Acceso a datos.
- **Capa Servicio:** Lógica intermedia.
- **Capa UI:** Interfaces gráficas (Swing).
- **Capa Main:** Entrada principal del sistema.

## 3. Estructura del Proyecto

- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/conexion/Conexion.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/crud/PersonaCRUD.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/main/App.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/modelo/Persona.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/servicio/PersonaService.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/ActualizarPersona.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/IngresarPersona.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/MenuPrincipal.java

- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/MostrarPersona.java
- proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/Principal.java

## 4. Documentación de Clases

### Conexion.java

Ubicación:

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/conexion/Conexion.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

### Vista previa del código:

```
package proyectoPersona.conexion;

import com.google.auth.oauth2.GoogleCredentials;

import com.google.firebase.FirebaseApp;

import com.google.firebase.FirebaseOptions;

import com.google.firebase.cloud.FirestoreClient;

import com.google.cloud.firestore.Firestore;

import java.io.FileInputStream;

import java.io.IOException;

public class Conexion {

    private static Firestore db;
```

```

public static Firestore getConexion() {

    if (db == null) {

        try {

            FileInputStream serviceAccount = new
FileInputStream("firebase/serviceAccountKey.json");

```

### **PersonaCRUD.java**

#### **Ubicación:**

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/crud/  
PersonaCRUD.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

#### **Vista previa del código:**

```

package proyectoPersona.crud;

import com.google.api.core.ApiFuture;

import com.google.cloud.firestore.*;

import java.util.ArrayList;

import java.util.List;

import proyectoPersona.conexion.Conexion;

import proyectoPersona.modelo.Persona;

import java.util.concurrent.ExecutionException;

```

```
public class PersonaCRUD {
```

```
    private final Firestore db;
```

Aquí creo la conexión con Firestore y apunto a la colección "personas"

```
    public PersonaCRUD() {
```

```
        db = Conexion.getConnection();
```

```
    }
```

Aquí agregué el método para crear una persona en Firestore

```
    public void...
```

### **App.java**

#### **Ubicación:**

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/main/  
App.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

#### **Vista previa del código:**

```
package proyectoPersona.main;
```

```
import proyectoPersona.crud.PersonaCRUD;
```

```
import proyectoPersona.modelo.Persona;
```

```
import proyectoPersona.ui.Principal;
```

```

public class App {

    public static void main(String[] args) {

        Principal menu = new Principal();

        Se visualiza el formulario principal

        menu.setVisible(true);

        menu.setLocationRelativeTo(null);

        menu.setTitle("MENÚ PRINCIPAL");

```

Aquí creo un objeto de la clase **PersonaCRUD** para acceder a los métodos de **Firebase**

```

PersonaCRUD crud = new PersonaCRUD();

```

### **Persona.java**

Ubicación:

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/modelo/Persona.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

Vista previa del código:

```

package proyectoPersona.modelo;

```

```

public class Persona {

    private String id;

```

```
private String nombre;
```

```
private String apellido;
```

```
private int edad;
```

Aquí agregué un constructor vacío para que Firebase pueda crear objetos automáticamente

```
public Persona() {  
  
}
```

Aquí agregué un constructor para inicializar los atributos de la persona

```
public Persona(String id, String nombre, String apellido, int edad) {  
  
    this.id = id;  
  
    this.nombre = nombre;  
  
    this.apellido = apellido;  
  
    this.edad = edad;  
  
}
```

Aquí agregué los ...

### **PersonaService.java**

Ubicación:

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/servicio/PersonaService.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

### Vista previa del código:

```
/*

 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license

 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

 */

package proyectoPersona.servicio;


/**

 *

 * @author Tony_Trrigue

 */

public class PersonaService {

}

...
```

### ActualizarPersona.java

#### Ubicación:

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/ActualizarPersona.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

### Vista previa del código:

```
/*

 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license

 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template

 */
```

```

package proyectoPersona.ui;
import javax.swing.JOptionPane;
import proyectoPersona.crud.PersonaCRUD;
import proyectoPersona.modelo.Persona;

/**
 *
 * @author Sumal
 */
public class ActualizarPersona extends javax.swing.JFrame {

    private static final java.util.logging.Logger logger =
        java.util.logging.Logger.getLogger(ActualizarPersona.class.getName());
    PersonaCRUD operaciones = new ...

```

### **IngresarPersona.java**

#### **Ubicación:**

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/IngresarPersona.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

#### **Vista previa del código:**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
 */
package proyectoPersona.ui;
import static java.lang.Integer.parseInt;
import javax.swing.JOptionPane;
import proyectoPersona.crud.PersonaCRUD;
import proyectoPersona.modelo.Persona;

/**
 *
 * @author Kimberly Sumala
 */
public class IngresarPersona extends javax.swing.JFrame {

    private static final java.util.logging.Logger logger =

```

```
java.util.logging.Logger.getLogger(IngresarPersona.clas...
```

### **MenuPrincipal.java**

#### **Ubicación:**

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/MenuPrincipal.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

#### **Vista previa del código:**

```
/*
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
```

```
*/
```

```
package proyectoPersona.ui;
```

```
/**
```

```
 *
```

```
 * @author Tony_Trrigue
```

```
*/
```

```
public class MenuPrincipal {
```

```
}
```

```
...
```

## MostrarPersona.java

### Ubicación:

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/MostrarPersona.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

### Vista previa del código:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
 */
package proyectoPersona.ui;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import proyectoPersona.crud.PersonaCRUD;
import proyectoPersona.modelo.Persona;
import proyectoPersona.ui.ActualizarPersona;

/**
 *
 * @author Kimberly Sumala
 */
public class MostrarPersona extends javax.swing.JFrame {

    private static final java.util.logging.Lo...
```

## Principal.java

### Ubicación:

proyectoPersona/proyectoPersona/proyectoPersona/src/main/java/proyectoPersona/ui/Principal.java

Descripción general pendiente de completar con mayor detalle si el usuario lo solicita.

### Vista previa del código:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
```

\* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template

```
*/  
package proyectoPersona.ui;  
  
/**  
 *  
 * @author Kimberly Sumala  
 */  
public class Principal extends javax.swing.JFrame {  
  
    private static final java.util.logging.Logger logger =  
java.util.logging.Logger.getLogger(Principal.class.getName());  
  
    /**  
     * Creates new form Principal  
     */  
    public Principal() {  
        initComponents();  
    }  
  
    /**  
     * This method is called from w...
```

## 5. Tecnologías utilizadas

### Lenguajes de programación

- **Java**, utilizado tanto para la lógica del programa como para la interfaz gráfica.

### Librerías, frameworks y versiones

- **Java Swing**, encargado de la interfaz de usuario.
- **Firebase Admin SDK**, que permite conectarse y trabajar con Firebase.
- **Google Gson**, usado para convertir objetos Java a formato JSON y viceversa.

### Bases de datos

- **Firebase Realtime Database**
  - Una base de datos en la nube muy flexible.

- No requiere instalación ni mantenimiento del usuario.
- Se actualiza en tiempo real.

## **Dependencias**

El proyecto utiliza varias dependencias que facilitan la comunicación con Firebase y el manejo de datos:

- firebase-admin
- gson
- google-auth-library

Estas se descargan automáticamente mediante Maven, lo cual simplifica la configuración.

## **6.Instalación y configuración**

### **Requisitos del sistema**

Para usar o modificar el proyecto se necesita:

- **Java JDK 8 o superior**
- **Conexión a Internet**
- Un entorno de desarrollo (IDE) como IntelliJ, NetBeans o Eclipse
- El archivo JSON de credenciales de Firebase

### **Pasos para instalar o ejecutar el proyecto**

1. Descargar o clonar el proyecto.
2. Abrirlo con un IDE compatible con Maven.
3. Configurar la ruta del archivo de credenciales de Firebase en Conexion.java.
4. Asegurarse de que Maven descargue todas las dependencias.
5. Ejecutar la clase App.java para iniciar el sistema.
6. Ya dentro del menú principal, se puede acceder a todas las funciones desde la interfaz gráfica.

### **Variables de entorno y configuraciones**

- Es importante que el archivo de credenciales de Firebase esté en una ruta válida.

- La URL de la base de datos debe coincidir con la del proyecto configurado en Firebase.
- No se requieren variables de entorno especiales ni configuraciones adicionales.

## **7.Mantenimiento**

### **Cómo actualizar el sistema**

Actualizar el sistema es bastante sencillo:

- Verificar si existen nuevas versiones de las dependencias, especialmente del SDK de Firebase.
- Mantener el proyecto organizado en sus capas para evitar problemas al hacer cambios.
- Probar cada actualización para asegurarse de que la comunicación con Firebase siga funcionando correctamente.

### **Cómo agregar nuevas funcionalidades**

Si se desea expandir el sistema, se puede hacer de forma ordenada siguiendo estos pasos:

1. Añadir nuevas propiedades en la clase Persona.
2. Ajustar las operaciones CRUD para trabajar con los nuevos datos.
3. Crear o modificar formularios en la interfaz gráfica.
4. Implementar la lógica necesaria en la capa de servicios.

La estructura modular del proyecto permite añadir funciones sin afectar lo que ya existe.

### **Consideraciones futuras**

Algunas ideas para futuras mejoras pueden ser:

- Añadir validaciones más completas en los formularios.
- Migrar la interfaz a JavaFX para lograr un diseño más moderno.
- Incluir autenticación de usuarios.
- Permitir exportar la información a formatos como PDF o Excel.

- Usar Firebase Firestore si se desea una estructura de datos más compleja.

## 8.Recomendaciones

El proyecto *Persona* demuestra que es posible crear un sistema funcional y bien estructurado sin necesidad de infraestructura compleja. Su arquitectura modular facilita tanto el mantenimiento como la expansión del proyecto. Además, al basarse en Firebase, se obtiene una solución escalable y confiable desde el primer momento.

Para continuar mejorando el sistema, se recomienda mantener buenas prácticas de programación, documentar las nuevas funciones que se agreguen y considerar migraciones tecnológicas según lo requiera el crecimiento del proyecto.