

MEMORIA PROYECTO INTEGRADO

“Declaración de la renta”



Realizado por

Jesús Cueto González

Tutorizado por

JOSE ANTONIO LARA SANCHEZ

Grado superior en desarrollo de aplicaciones multiplataforma

Cesur (Málaga Este)

Málaga, mayo 2021.

Índice.

1. Introducción.
 - 1.1 Introducción a Android Studio
 - 1.2 Interfaz del usuario
 - 1.3 Interfaz gráfica
 - 1.3.1 Elementos de la UI
 - 1.3.2 Posicionamientos de elementos en la UI
 - 1.4 Introducción a la declaración de la renta
 - 1.4.1 IRPF
 - 1.5 Código con Android StudioPlanificación del Proyecto.
 - 1.5.1 Creación de código en Android Studio.
2. Planificación del proyecto
 - 2.1 Elección de idea y entorno de desarrollo.
 - 2.2 Definición y desarrollo de clases.
 - 2.2.1 Shared Preferences
 - 2.2.2 Sql Server
 - 2.2.2.1 Funciones y Características de Microsoft Sql Server
 - 2.2.3 Toast
 - 2.2.4 JDBC
 - 2.2.4.1 Tipos de Drivers
 - 2.2.4.2 Métodos de la clase
 - 2.2.5 Java.mail
 - 2.2.5.1 Características
3. Fase de Pruebas.
 - 3.1 Pruebas durante el desarrollo
 - 3.2 Pruebas una vez finalizado el desarrollo.
4. Conclusiones y posibles mejoras
 - 4.1 Conclusiones personales.
 - 4.2 Conclusiones sobre el proyecto.
 - 4.3 Posibles mejoras a futuro.
5. Guía de instalación.

- 5.1 Guía de instalación del juego en un terminal Android.
- 6. Manual de usuario
 - 6.1 Manual de usuario
- 7. Bibliografía.

Resumen.

Este proyecto es enmarcado dentro del desarrollo de aplicaciones móviles, más concretamente dentro del Sistema operativo Android.

La idea desarrollada se trata de una aplicación para dispositivos móviles, en concreto para dispositivos Android. La temática propuesta es la declaración de la renta, y el desarrollo de la aplicación será poder saber el resultado de la declaración de la renta sí toca que devuelvan o a pagar.

Centraremos el proyecto en la utilización del entorno de desarrollo de aplicaciones en Android Studio, donde se demostrarán conocimientos adquiridos en el curso de “Desarrollo de Aplicaciones Multiplataforma”.

Las pruebas se hicieron dentro del mismo entorno de desarrollo y, además, en un terminal Android para mayor corrección de la aplicación.

Objetivos y características del proyecto.

El objetivo de este proyecto es demostrar conocimientos adquiridos sobre programación orientada a dispositivos móviles y el lenguaje de programación Java; así como el uso del entorno de desarrollo de Android Studio; además de concienciar al usuario a realizar la declaración de la renta de forma rápida y sencilla.

Finalidad.

Con el desarrollo de esta aplicación lo que buscamos es hacer ver que se puede hacer la declaración de la renta, una vez se tiene los datos necesarios, de forma sencilla e intuitiva a la vez.

Medios materiales utilizados.

Para el desarrollo de este proyecto hemos utilizado:

- Un equipo con Windows 10 y con una tarjeta gráfica integrada.
- La versión 2022 de Android Studio.
- La escala de gravamen de IRPF de 2021-2022.
- Emulador de móvil propio de Android Studio
- Un terminal con Android (versión 12).

1. Introducción.

1.1. Introducción a Android Studio.

A modo de introducción, pasaremos a explicar en concreto qué es Android Studio y qué aspectos de este entorno vamos a utilizar.

Android Studio es un entorno de desarrollo para desarrollo de aplicaciones para móviles en especial, como su propio nombre indica, Android, su enfoque es para un desarrollador que ya sabe o de java o de kotlin que es el otro lenguaje con el que trabaja y su curva de aprendizaje es parecido o igual a la que tiene java ya que no es exactamente lo mismo pero un 90% similar, ya que aunque trabaje con java, no es idéntico, tiene sus diferencias. Uno de los puntos fuertes de Android Studio para el desarrollo es que da buena visibilidad a la hora de crear el front que es lo que vera el cliente , y buen orden de los activities que son lo que vendría a ser como las clases de java, es lo que por lo general representa cada pantalla de la aplicación.



Imagen 1.1. Introducción a Android Studio.

1.2. Interfaz del usuario

A continuación, pasaremos a explicar los elementos que componen Entorno de desarrollo de Android Studio

- **La barra de herramientas:** te permite realizar una gran variedad de acciones, como ejecutar tu app e iniciar las herramientas de Android.
- **Barra de navegación:** te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana Project.
- **Ventana del editor:** es el área en la que puedes crear y modificar código. Según el tipo de actividad actual, el editor puede cambiar. Por ejemplo, cuando ves un archivo de diseño, el editor muestra el Editor de diseño.
- **Barra de la ventana de herramientas:** se encuentra afuera de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.
- **Venta de herramientas:** te brindan acceso a tareas específicas, como la administración de proyectos, la búsqueda, el control de versiones, entre otras. Puedes expandirlas y contraerlas.
- **Barra de estado:** se muestra el estado de tu proyecto y el IDE, además de advertencias o mensajes.

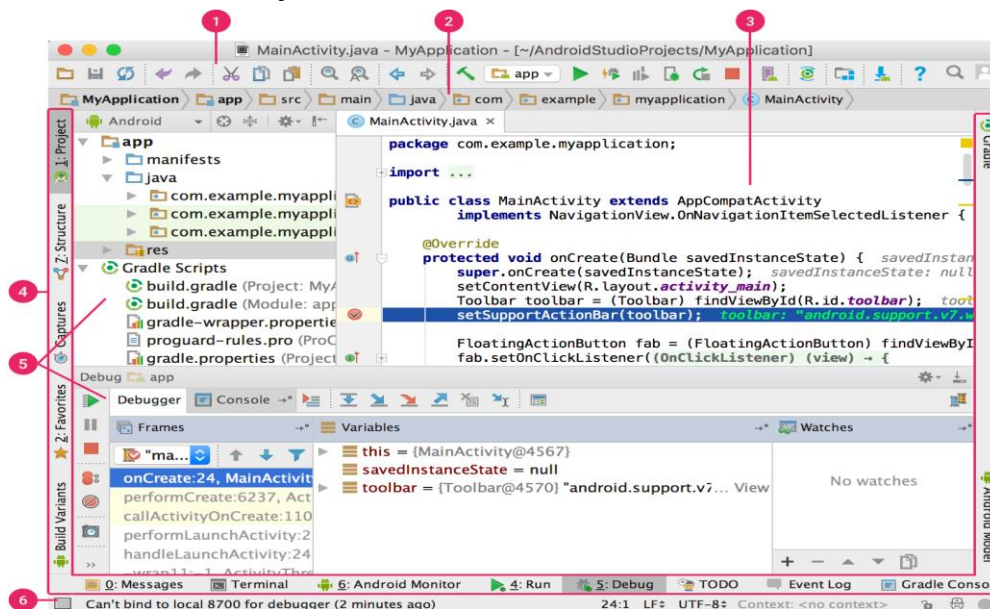


Imagen 1.2. Interfaz de usuario.

Este entorno de desarrollo se apoya en la programación orientada a objetos. Esto también es utilizado en java y es uno de los paradigmas de programación más utilizados, lo que se

desarrolla en este caso son las pantallas o actividades que son los objetos, cada pantalla es un objeto diferente con su funcionalidad, que puede ir ligada o no a la anterior.

En Android Studio, esta arquitectura se implementa mediante la carpeta *project*; en la escena todos los objetos están en dicha carpeta y estos objetos a su vez tendrán componentes y determinarán su comportamiento propio de forma individual o grupal. En la barra de herramientas esta dicha carpeta entre otras.

Esta forma de programar es muy intuitiva, ya que nos es muy fácil imaginar cómo diseñar cada objeto de forma individual, además la herramienta gráfica de Android Studio también nos ayuda a visualizar todos los elementos que estamos utilizando en todas las fases del proyecto.

Por otro lado, también podemos llegar a tener muchas dificultades en el proceso de desarrollo; ya que, si no añadimos los componentes adecuados o no los configuramos bien, el objeto con el que estamos trabajando puede no comportarse como queremos independientemente de haber programado bien su comportamiento vía *Código*. De hecho, una de las mayores fuentes de errores es cometer algún error en la configuración de los componentes dentro del proyecto, por lo que a la hora de trabajar con este tipo de entornos tendremos que estar muy atentos a los posibles errores que podemos cometer, así como tener muy claro cómo queremos que se comporte cada elemento de la escena y saber que componentes debe tener para que funcione como es debido.

➤ Todos los objetos tienen una serie de propiedades que podremos configurar.

Toda la programación de estos objetos se basa en la creación de Código que controlarán el comportamiento de los métodos de dicho objeto. Estos Códigos son muy similares a los que se harían propiamente en java por ejemplo.

Para añadir objetos al proyecto en Android Studio, lo podemos hacer a través de la misma carpeta del proyecto *App*.

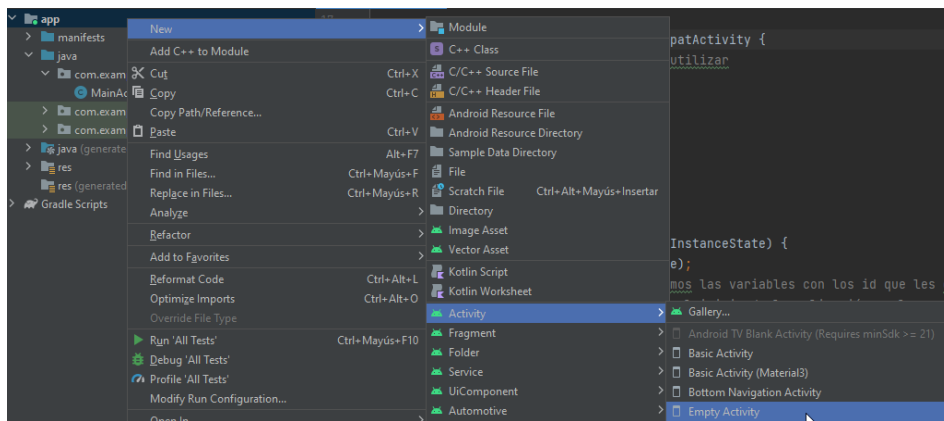


Imagen 1.3. Introducción de elementos en la Jerarquía.

1.3. Interfaz Gráfica.

Android Studio cuenta con una interfaz gráfica que se puede ver tanto en xml como en gráfico, ya que se puede arrastrar de esta forma los elementos al entorno gráfico. Y cuenta a parte con varias posibilidades de modificación para dichos elementos. Esta parte de la aplicación se almacena en un componente llamado *Main Activity.xml*, este es el nombre que viene por defecto, en él ubicaremos todos los elemento referentes a la interfaz de usuario (por ejemplo, las imágenes, etiquetas de texto, o botones).

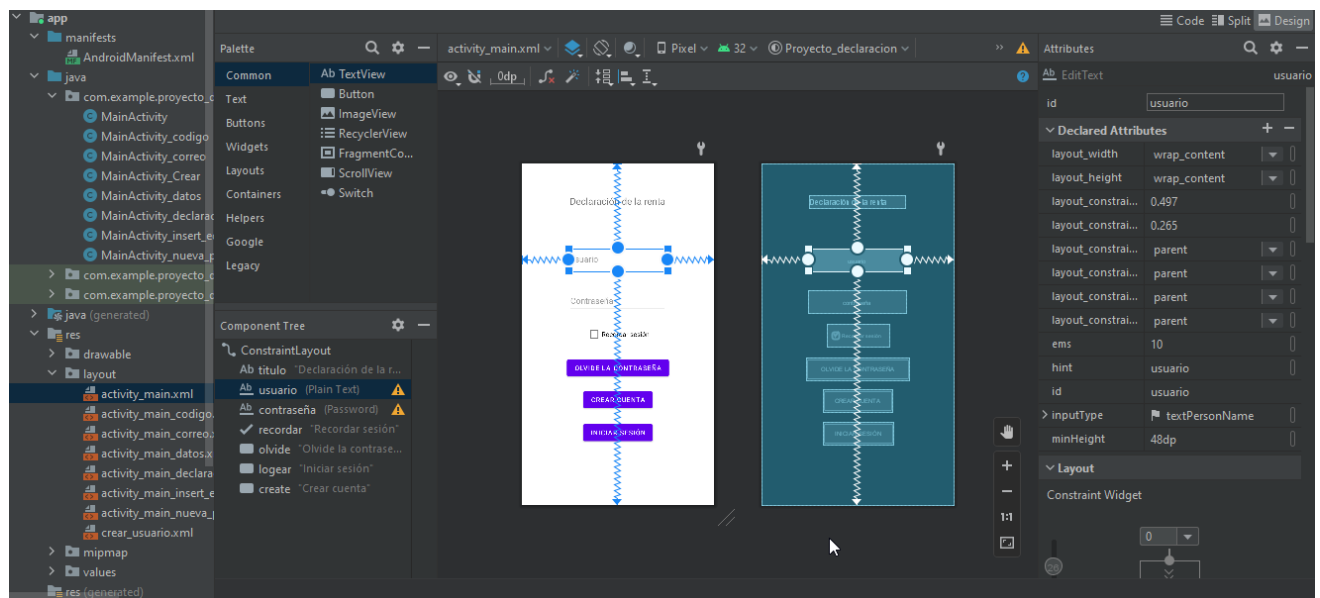


Imagen 1.4. Visualización del xml en el entorno gráfico.

La interfaz gráfica es un panel gráfico que incluye todos los componentes de la interfaz de usuario. Dichos componentes deben estar dentro del `xml` y a su vez dentro del `androidmanifest` para que vaya enlazado al objeto correspondiente.

Una propiedad importante de este componente es la propiedad de la vista y se refiere a de qué manera se va a mostrar el `xml`; tendremos tres opciones:

- **Code:** El `xml` se dibujará como un `xml` clásico.
- **Split:** Se mostrará de las dos maneras, como *Code* y como *Design*.
- **Design:** Esta opción es la que más se utiliza ya que es la más rápida e intuitiva a la hora de realizar las pantallas ya que los estás viendo en directo y puedes moverlo de forma directa sin código extra ni tienes que saber `html` ni nada parecido ya que lo crea solo, sería como la otra herramienta que se dio en el curso, Scene Builder que era para `java`.

1.3.1. Elementos de la UI

- a. **Etiquetas de Texto:** Son objetos de un componente tipo *Text*, y heredarán todas las características del mismo. Este objeto nos permite mostrar un texto en la interfaz, además podremos ajustar las propiedades típicas como color, fuente, tamaño...
- b. **Imágenes:** Del mismo modo que las etiquetas de texto, las imágenes heredan del componente *Image*. En una imagen deberemos introducir una

imagen o un gif para mostrarlo como imagen. Lo ideal es usar imágenes en formato PNG.

- c. **Botones:** Es un objeto que hereda de *Button*, además contendrá en su interior un texto para mostrar una etiqueta de texto a modo de descripción del botón.

Además, uno de los parámetros del botón será introducir una instrucción al ser pulsado (evento *OnClick*), es decir, hará una llamada a un método de nuestro código.

```
btncrear.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        Intent siguiente = new Intent(view.getContext(), MainActivity_Crear.class);  
        startActivity(siguiente);  
    }  
});
```

Imágenes 1.5. Configuración de un botón vía Código.

En este ejemplo podremos cambiar de una pantalla a otra.

1.3.2. Posicionamiento de elementos en la UI

Ahora que conocemos las propiedades de los elementos de la Interfaz de usuario, es importante colocar éstos correctamente. Para trabajar con esto, Android Studio incluye una serie de propiedades de posicionamiento algo más concretas que las que tenemos en el posicionamiento normal de la pantalla.

El componente *Layout* nos permite elegir también una serie de posiciones predeterminadas para colocar los elementos

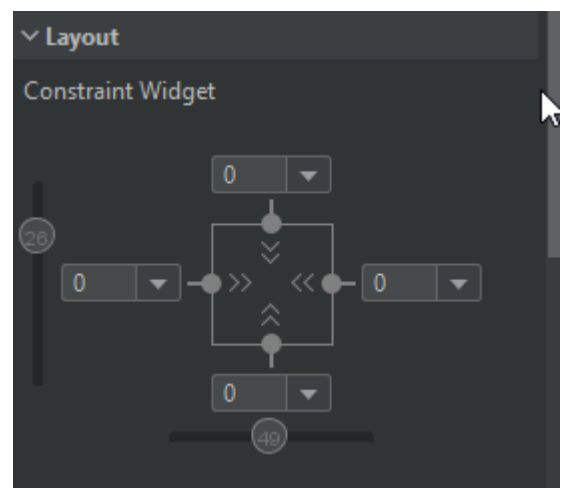


Imagen 1.6. Anclaje de objetos en el Gráfico.

dentro del *xml*. Esta propiedad se conoce como ‘anclar’, es decir, definir la zona de la pantalla donde queremos que aparezca el elemento, la posición del objeto ahora será relativa al punto de anclaje.

Veamos esto con un ejemplo:

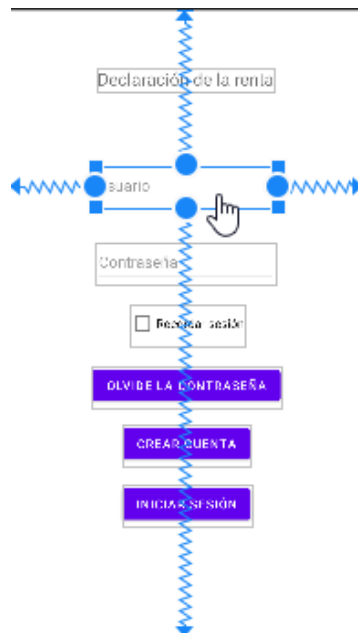


Imagen 1.7. Creación de un texto que aparece en el xml

Como se puede ver en la imagen, hemos configurado los *RectTransform* de este elemento para que el texto se ubique arriba en el centro, y un botón abajo. También es posible customizar los anclajes, lo vemos en una imagen.

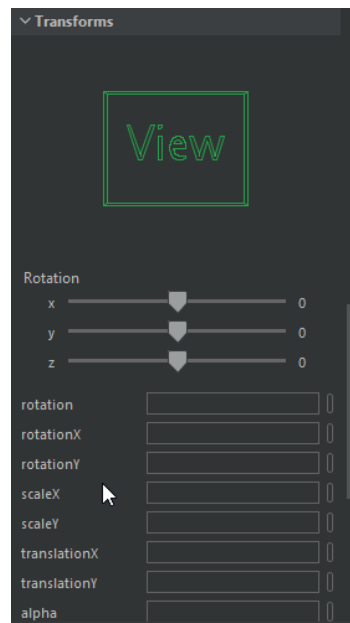


Imagen 1.8. Configuración de un texto a través de su Transform.

1.4. Introducción a la declaración de la renta

Ahora que conocemos un poco el entorno de desarrollo, pasamos a introducir la parte que nos ocupa en el proyecto, la declaración de la renta.

Primero, una pequeña introducción sobre que es la declaración de la renta. La declaración de la renta o del IRPF (Impuesto sobre la Renta de las Personas Físicas) es un trámite anual que tienen la obligación de realizar los residentes en España para regularizar su situación con la Agencia Tributaria. Sin embargo, no todas las personas tienen que hacerla, ya que depende de las rentas que se hayan obtenido a lo largo del año. Dado que las infracciones tributarias conllevan multas que pueden ser importantes, conocer cuándo y en qué casos es obligatorio presentar la declaración es fundamental para evitar gastos imprevistos que puedan afectar en el futuro a nuestra salud financiera.



Imagen 1.9. Introducción a la declaración de la renta.

1.4.1. IRPF

El IRPF es un impuesto que deben pagar las personas físicas residentes en España por todos los beneficios o rentas que hayan obtenido a lo largo del año fiscal. Es decir, por los ingresos netos obtenidos una vez restados a los ingresos brutos los gastos que sean deducibles. Incluye tanto los ingresos recibidos como asalariados (la nómina, por ejemplo) o los obtenidos como trabajadores por cuenta propia (es el caso de los

autónomos) como los procedentes del cobro de prestaciones públicas como una pensión.

También es obligatorio declarar los rendimientos del capital mobiliario (alquileres, por ejemplo) y ganancias patrimoniales, incluyendo estas últimas premios como la lotería, concursos, reembolso de fondos de inversión, operaciones con criptomonedas, etc.).



Imagen 1.10. Tramos del IRPF.

Estarán obligados a hacerla principalmente los que igualen o superen los 22.000 euros anuales pero cumpliendo unos requisitos;

- Siempre que procedan de un solo pagador.
- Cuando existan varios pagadores, siempre que la suma del segundo y posteriores por orden de cuantía no superen en su conjunto la cantidad de 1.500 euros.
- Si los únicos rendimientos de trabajo consistan en pensiones de la Seguridad Social y otras prestaciones pasivas.

1.5. Código con Android Studio.

Dentro del proyecto en Android Studio, uno de los más importantes dentro del desarrollo es el del código. Ya que sin el claramente no funcionaría la aplicación, como ya se dijo anteriormente, está desarrollado en java ya que es el que más se ha trabajado en el curso y es bastante completo.

1.5.1. Creación de código en Android Studio.

La creación de código es bastante parecida a la de java como ya se dijo, ya que sigue siendo java aunque con algunos toques diferentes.

A continuación se enseñara un poco del código realizado por partes de la aplicación realizada.

```
package com.example.proyecto_declaracion;

import ...

public class MainActivity extends AppCompatActivity {
    Button btnLogin;
    SharedPreferences preferences;
    SharedPreferences.Editor editor;
    public EditText usuario;
    public EditText contraseña;
    private CheckBox chekRecordardatos;
    private Connection conexion = null;
    private Statement St;
    private ResultSet rs;
    private Button btncrean;
    private Button btnolvide;
    public static final String ENVIAR_usuario = "nombre";
    public static final String ENVIAR_contraseña = "contraseña";
    private SharedPreferences prefe;
    private SharedPreferences prefe2;
```

Imagen 1.11. Variables de la clase.

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    usuario = (EditText) findViewById(R.id.usuario);
    contraseña = (EditText) findViewById(R.id.contraseña);
    btnLogin = (Button) findViewById(R.id.logear);
    btncrear = (Button) findViewById(R.id.create);
    btnolvide = (Button) findViewById(R.id.olvide);
    chekRecordardatos = (CheckBox) findViewById(R.id.recordar);
    SharedPreferences preferencias = getSharedPreferences( name: "usuario", Context.MODE_PRIVATE);
    usuario.setText(preferencias.getString( s: "usuario", s1: ""));
    SharedPreferences preferencias2 = getSharedPreferences( name: "contraseña", Context.MODE_PRIVATE);
    contraseña.setText(preferencias2.getString( s: "contraseña", s1: ""));
    if(prefe != null && prefe2 != null){
        chekRecordardatos.setChecked(true);
    }else{
        chekRecordardatos.setChecked(false);
    }
}

```

Imagen 1.12. Método del Main Activity principal.

```

btnLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(usuario.getText().toString() == null || contraseña.getText().toString() == null){
            Toast toast = Toast.makeText(getApplicationContext(),
                toast: "Introduzca todos los datos correspondientes.",
                Toast.LENGTH_SHORT);
            toast.show();
        }else{
            connect();
            try {
                String consulta;
                consulta = "SELECT usuario , contraseña FROM cliente WHERE usuario LIKE " + usuario.getText().toString() + " ";
                St = conexion.createStatement();
                rs = St.executeQuery(consulta);
                if(rs.next()) {
                    Intent siguiente = new Intent( packageContext MainActivity.this, MainActivity_datos.class);
                    String user = usuario.getText().toString();
                    String pass = contraseña.getText().toString();
                    siguiente.putExtra("usuario", user);
                    siguiente.putExtra("password", contraseña.getText().toString());
                    startActivity(siguiente);
                } while(rs.next());
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
});

```

Imagen 1.13. Método Login.


```

btncrear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent siguiente = new Intent(view.getContext(), MainActivity_Crear.class);
        startActivity(siguiente);
    }
});
btnolvide.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent siguiente = new Intent(view.getContext(), MainActivity_correo.class);
        startActivity(siguiente);
    }
});

```

Imagen 1.14. Método Crear y olvide contraseña.

```

public Connection connect() {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        Class.forName("net.sourceforge.jtds.jdbc.Driver");
        //conexion = DriverManager.getConnection (url,user, password);
        conexion = DriverManager.getConnection( url: "jdbc:jtds:sqlserver://192.168.1.151;databaseName=declaracion;user=jesus;pa

        System.out.println("Conectado a la base de datos con éxito !!!");
    } catch (Exception e) {
        System.out.println("NO CONECTA!");
        e.printStackTrace();
    }
    return conexion;
}

```

Imagen 1.15. Método que hace conexión a la base de datos.

2. Planificación del Proyecto

Ahora vamos a pasar a describir todas las fases por las que ha pasado el desarrollo de esta aplicación.



Imagen 2.1. Planificación del proyecto.

Vamos a dividir las etapas del proyecto en varias fases que se desarrollarán punto a punto:

- *Brain Storming* y elección de idea, entorno de desarrollo... etc.
- Elección del escenario y temática. Modelado del escenario.
- Definición y desarrollo de las clases.
- Retoques finales. Creación de una Build.
- Pruebas.

2.1. Elección de idea y entorno de desarrollo

En esta fase de desarrollo al principio, fue una etapa difusa en la que hubo dificultades para tener una idea clara para reunir los conceptos que se han aprendido y a la vez crear algo original.

La primera idea firme que se planteó (y fue al final la que se eligió) fue hacer una aplicación para dispositivos móviles que reuniera el máximo de conceptos aprendidos. Por ejemplo, una aplicación móvil que funcione como CRUD y se conecte a una base de datos.

Otra opción que se barajaba fue hacer un juego con Unreal.

Finalmente se decidió hacer un proyecto relacionado con el primer tema y relacionado con un tema que se tiene que hacer cada año por estas épocas como es la declaración de la renta.

Se buscó información de cómo se realizaba la declaración.

Una vez que se había buscado suficiente información tocaba organizar las ideas junto con la información buscada.

Una vez que ya se ha planteado todo esto toca hacer el mapeo del seguimiento de la aplicación, el cómo se va a realizar y que componentes va a tener la aplicación.

A partir de aquí, se decide desarrollar esa idea y se plantea una aplicación de la declaración de la renta conectada a una base de datos y que a su vez tenga un control de usuarios con los datos y que se pueda cambiar o recuperar la contraseña del usuario por un código que se envía al correo proporcionado, aparte de poder recordar los datos introducidos con un checkbox.

2.2. Definición y desarrollo de clases.

A continuación se explicara el cómo se ha desarrollado la aplicación;

- El primer paso fue tener los cálculos y los datos necesarios para la declaración de la renta.
- Luego de tener esos datos toco crear la base de datos con las dos tablas;
 - La primera tabla con los datos de login del usuario.
 - Y la segunda con los datos necesarios para la declaración de la renta.
- Luego de tener la base de datos creada, había que crear la primera pantalla que se dividía en varios sectores;
 - El primero sería el login como tal que comprobaría si el usuario y la contraseña existen o coinciden, en caso contrario mostraría un mensaje y que pasaría a otra pantalla;
 - Una pantalla que comprueba de primeras sí hay datos previos introducidos en la base de datos con el usuario y contraseña que ha introducido anteriormente en el login, si ya hay datos introducidos en la base de datos de los tres botones que hay en la pantalla: Insertar, Editar y Declaración de la renta. Se pondría en invisible el de Insertar ya que ya ha insertado los datos, y solo podría editarlos o hacer la declaración de la renta, en caso contrario se pondría invisible el de Editar ya que no puede editar datos debido a que no ha introducido nunca datos anteriormente. Tanto insertar como editar se van a una misma pantalla que tiene lo siguiente;

-
- Si viene de insertar los campos están en blanco y si viene de editar los campos se rellenarían con los que están en la base de datos con un ResultSet. Tiene además dos botones, el primero es guardar que insertaría o actualizaría los datos y el otro es cancelar que volvería a la anterior pantalla.
 - El botón de la declaración de la renta lo que hace es direccionar hacia una nueva pantalla que lo que hace es un cálculo con los datos necesarios para realizar la declaración de la hacienda correspondiente y una vez hecha lo muestra en la pantalla para saber si toca que devuelvan o sale a pagar.
 - El segundo que es la creación de un usuario nuevo, que antes de introducir el usuario haría la comprobación de que ese usuario no existiese, si existía mostraba un mensaje y en caso contrario introduciría el usuario junto con sus datos correspondientes.
 - La tercera sería olvide la contraseña que se divide en 3 pantalla;
 - La primera pantalla donde se introduciría el usuario con el correo, haría la comprobación de si ese usuario coincide con ese correo, si no coincide muestra un mensaje y en caso contrario, enviaría el código al correo proporcionado.
 - La segunda pantalla es donde se introduce el código enviado, que si el código enviado no coincide con el escrito, muestra el mensaje y en caso contrario pasaría a la siguiente y ultima pantalla.
 - La tercera pantalla lo que tendría sería dos textfield de tipo contraseña, la primera para introducir la contraseña nueva y la otra para repetirla y que no haya ninguna equivocación, si coinciden se

enviaría a la pantalla principal, si no coinciden mostraría un mensaje.

- El cuarto elemento sería un checkbox para recordar el inicio de sesión que funciona con shared preferences.

2.2.1 Shared Preferences

El uso principal de Shared Preferences es guardar las preferencias de los usuarios, configuraciones, tal vez datos (si no es muy grande) para que la próxima vez que la aplicación sea lanzada, estas piezas de información podrían ser recibidas y usadas.

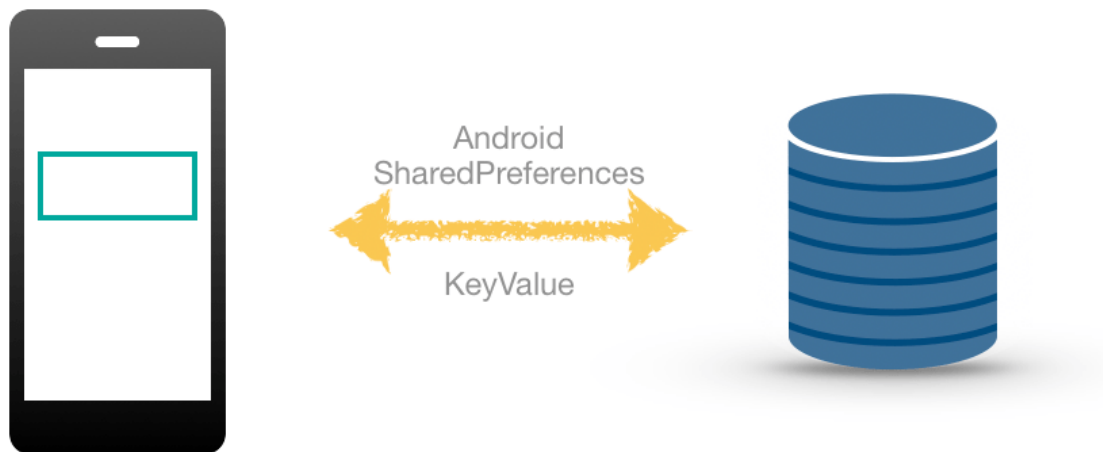


Imagen 2.2. Introducción a SharedPreferences.

Como controlar las preferencias;

Puedes crear un nuevo archivo de preferencias compartidas o acceder a uno existente llamando a uno de estos dos métodos:

- `getSharedPreferences()`: Este método se utiliza si se necesitan varios archivos de preferencias compartidas identificados por nombre, que se especifica con el primer parámetro. Puedes llamar a este método desde cualquier instancia de `Context` en tu app.
- `getPreferences()`: Este método se utiliza desde una instancia de `Activity` si se necesita utilizar un solo archivo de preferencias compartidas para la actividad. Como este método recupera un archivo de preferencias compartidas predeterminado que pertenece a la actividad, no necesitas indicar un nombre.

Ejemplo:

```
SharedPreferences sharedPref =  
getActivity().getPreferences(Context.MODE_PRIVATE);
```

Como escribir en las preferencias;

Para realizar operaciones de escritura en el archivo de preferencias compartidas, se crea un `SharedPreferences.Editor` llamando a `edit()` en `SharedPreferences`.

Pasa las claves y los valores que se desee escribir con métodos como `putInt()` y `putString()`. Luego, se llama a `apply()` o a `commit()` para guardar los cambios. Por ejemplo:

```
SharedPreferences sharedPref =  
getActivity().getPreferences(Context.MODE_PRIVATE);  
  
SharedPreferences.Editor editor = sharedPref.edit();  
  
editor.putInt(getString(R.string.saved_high_score_key), newHighScore);  
  
editor.apply();
```

Como leer los datos

Para recuperar valores de un archivo de preferencias compartidas, se llaman a métodos como `getInt()` y `getString()`, proporciona la clave del valor que desees y, opcionalmente, un valor predeterminado para mostrar si no se encuentra la clave. Por ejemplo:

```
SharedPreferences sharedPref =  
getActivity().getPreferences(Context.MODE_PRIVATE);  
  
int defaultValue = getResources().getInteger(R.integer.saved_high_score_default_key);  
  
int highScore = sharedPref.getInt(getString(R.string.saved_high_score_key),  
defaultValue);
```

2.2.2 Sql Server

Microsoft SQL Server es ideal para almacenar toda la información deseada en bases de datos relacionales, como también para administrar dichos datos sin complicaciones, gracias a su interfaz visual y a las opciones y herramientas que tiene.

Es algo vital, especialmente en webs que tienen la opción de registrar usuarios para que inicien sesión.



Imagen 2.3. Introducción a SqlServer.

Para las compañías, emplear esta herramienta es esencial por las facilidades que plantea y las utilidades con las que cuenta. Si se tiene un listado de clientes, un catálogo de productos o incluso una gran selección de contenido multimedia disponible, Microsoft SQL Server ayuda a gestionarlo absolutamente todo. Es básico para el buen funcionamiento de una web o de cualquier aplicación.

Su componente principal está compuesto por un motor relacional encargado del procesamiento de comandos, consultas, así como del almacenamiento de archivos, bb.dd., tablas y búferes de datos. Sus niveles secundarios están destinados a la gestión de la memoria, programación y administración de las interacciones de solicitud y respuesta con los servidores que alojan las bases de datos.

2.2.2.1 Funciones y Características de Microsoft SQL Server

Algunas de las funciones principales que distinguen a Microsoft SQL Server, son su variedad de herramientas destinadas a la gestión y análisis de datos, así como la inteligencia empresarial con la que obtener conocimientos sobre tu negocio y clientes apoyadas en machine learning.

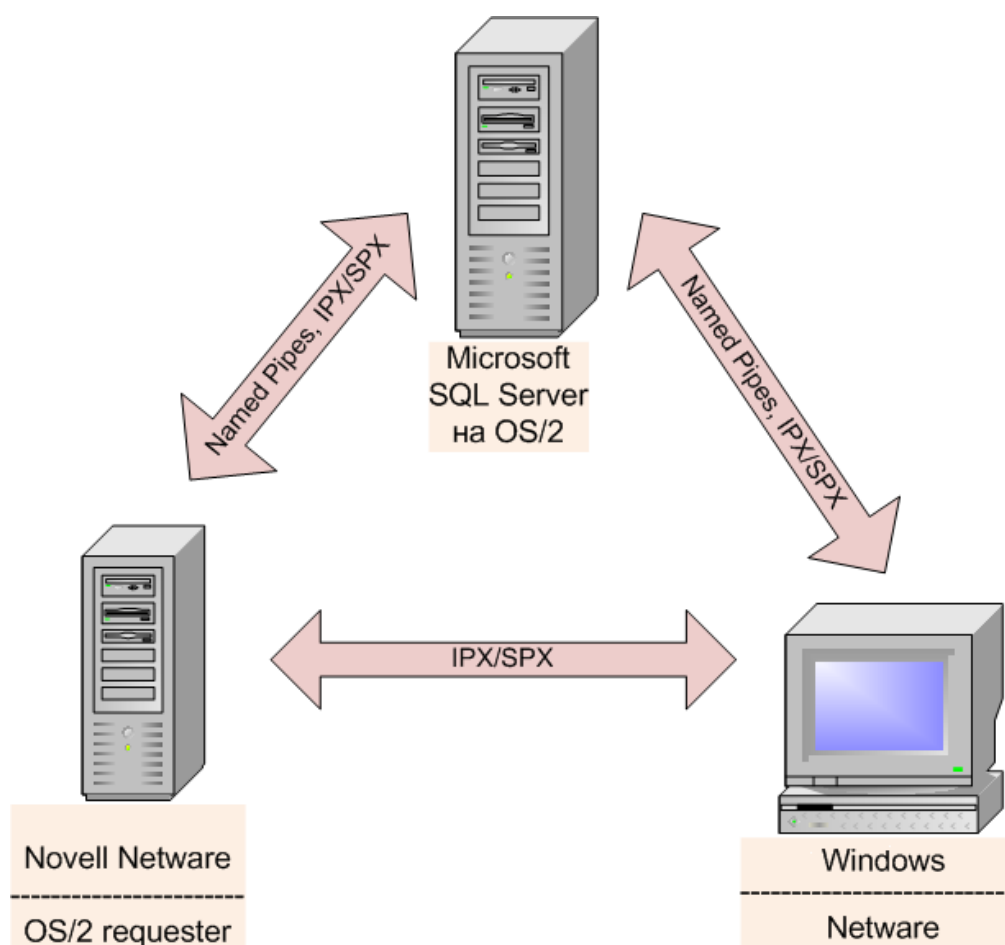


Imagen 2.4. Función y características de SqlServer.

Microsoft SQL Server permite integrar muy fácilmente tus datos en aplicaciones y aprovechar un amplio conjunto de servicios cognitivos con los que potenciar la inteligencia artificial en cualquier escala de datos, tanto en entornos on-premises y cloud gracias a su integración con Azure AI.

Generalmente, los servidores SQL Server ofrecen al usuario una alta disponibilidad con la que permitir procesos de conmutación más rápidos. Sus funcionalidades de memoria integrada permiten incrementar la flexibilidad y facilidad de uso otorgando una perfecta integración con la familia de servidores Microsoft Server.

Al estar basada en código abierto es muy fácil acceder y la gran mayoría de programadores que trabajan en desarrollo web han usado Microsoft SQL Server en alguno de sus proyectos, ya que además de estar muy extendido también tiene una gran comunidad que ofrece soporte a otros usuarios.

Las principales características son:

- Inteligencia en todos sus datos con clústeres de Big Data: pudiendo consultar todo tu patrimonio de datos desde SQL Serve hasta Oracle sin necesidad de replicarlos.

-
- Elección de Lenguaje y Plataforma: Desde Windows o Linux, hasta implementaciones con Kubernetes
 - Capacidades de bases de datos inteligentes: in-memory, soporte de memoria persistente, tempdb optimizado para memoria
 - Cifrado de datos y cumplimiento normativo: Su sistema de protección, supervisión y clasificación de datos la han convertido durante 9 años en una de las principales plataformas más seguras según la base de datos del National Institute of Standards and Technology.
 - BI móvil y escalabilidad: Permitiendo integrar fácilmente tus sistemas de gestión de bases de datos con cualquier dispositivo y servicios de Azure para obtener mejor rendimiento y capacidad de análisis sobre los datos.

Versiones de Microsoft SQL Server

La versión más reciente es Microsoft SQL Server 2019, disponible en 4 ediciones diferentes destinadas para cada diferentes perfiles y usos:

Enterprise: Siendo la edición más completa, está destinada para aquellas organizaciones que requieran trabajar con altos niveles de servicio para cargas de trabajo críticas.

- Estándar: Destinada para ofrecer a las pequeñas empresas una herramienta que les brinde una administración básica de datos para la ejecución de sus aplicaciones, admitiendo herramientas de desarrollo comunes a nivel on-premise o entornos cloud.

-
- Express: Esta versión está diseñada para aquellos fabricantes de software independientes o principiantes que deseen disponer de una base de datos gratuita como método de aprendizaje, para compilar pequeñas aplicaciones de servidor y escritorio destinadas al uso de estos

 - Developer: Integrando toda la funcionalidad del Enterprise, está diseñada para todos aquellos desarrolladores que deseen disponer de un sistema de prueba y desarrollo para la compilación de cualquier tipo de aplicación en SQL Server, no estando disponible para un entorno de producción, tan sólo de prueba.

2.2.3 Toast

Se ha trabajado con Toast para las muestras de mensajes.

Ha continuación se explicará un poco de él;

Una notificación emergente del tipo "toast" (tostada) en términos informáticos hace referencia a un elemento de control gráfico que comunica determinados sucesos al usuario sin forzarlo a reaccionar inmediatamente ante la notificación. En IBM Curam Universal Access, se utiliza el componente de alerta del sistema de diseño web como base para representar nuestras notificaciones emergentes "toast", y permitir que se muestren estas notificaciones independientemente del contenido de visualización principal en cualquier función de la aplicación.

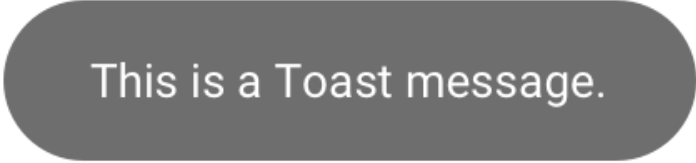


Imagen 2.5. Mensaje Toast.

El componente <Toast> expuesto es el componente preferido para mostrar notificaciones emergentes "toast". Acepta las propiedades, definidas por el componente de alerta del sistema de diseño web, sin tener que especificar el componente como una Alerta y las propiedades 'banner', 'center' y 'toast'. También requiere que se defina una propiedad 'text'.

2.2.4 JDBC

Se ha utilizado para la conexión a la base de datos Sql Server.

A continuación se explicara un poco sobre él;

Java Database Connectivity (en español: Conectividad a bases de datos de Java), más conocida por sus siglas JDBC,¹² es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

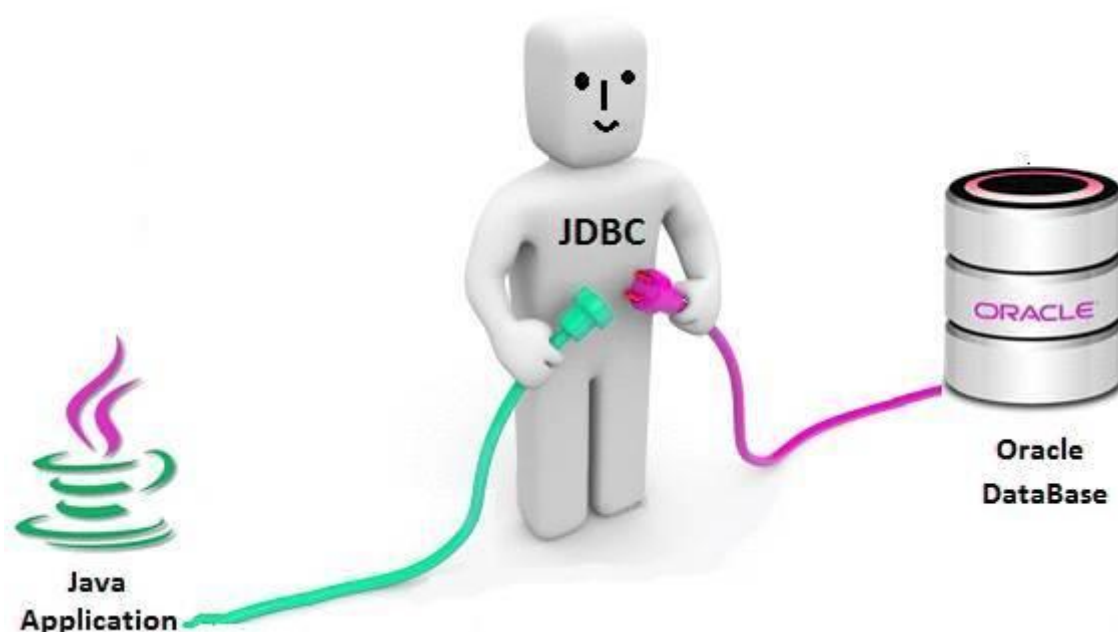


Imagen 2.6. Introducción a JDBC.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar.

Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella

estableciendo una conexión; para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tarea con la base de datos a la que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

2.2.4.1 Tipos de Drivers

Los drivers JDBC son adaptadores del lado del cliente (instalados en la máquina cliente, no en el servidor) que convierten la petición proveniente del programa JAVA a un protocolo que el SGBD pueda entender.

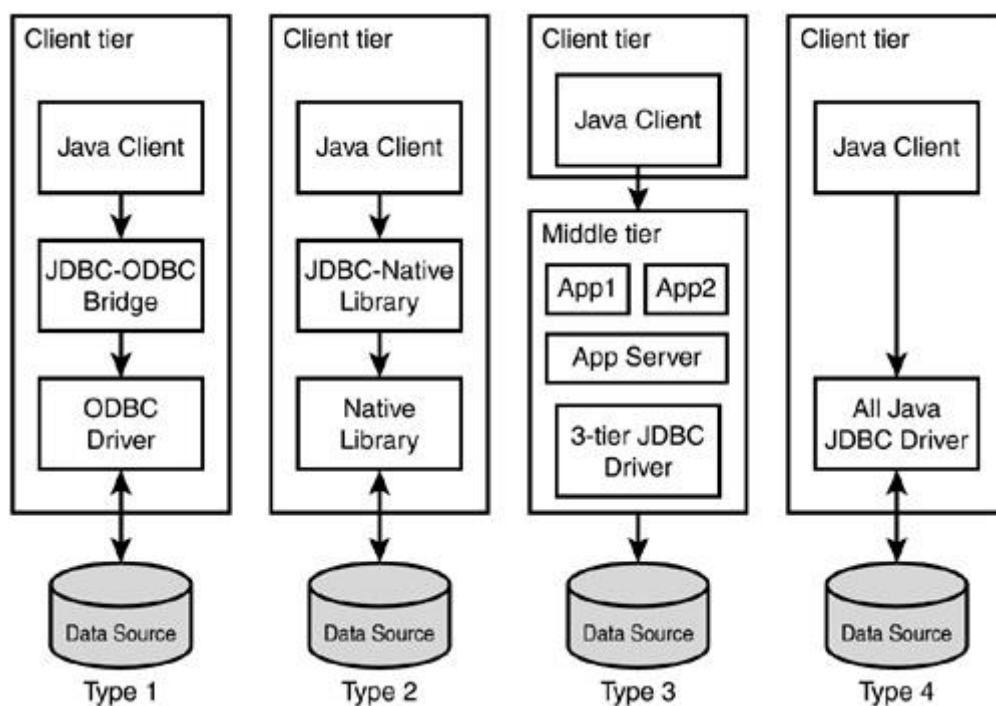


Imagen 2.7. Tipos de Drivers.

-
- Driver JDBC Tipo 1 (también llamado Puente JDBC-ODBC) convierte el método JDBC a una llamada a una función ODBC. Utiliza los drivers ODBC para conectar con la base de datos.
 - Driver JDBC Tipo 2 (también llamado driver API-Nativo) convierte el método JDBC a llamadas nativas de la API de la base de datos. Es más rápido que el puente JDBC-ODBC pero se necesita instalar la librería cliente de la base de datos en la máquina cliente y el driver es dependiente de la plataforma.
 - Driver JDBC Tipo 3. Hace uso de un Middleware entre el JDBC y el SGBD.
 - Driver JDBC Tipo 4 (también llamado Driver Java Puro directo a la base de datos). Es independiente a la plataforma.

Se ha utilizado el tipo 2 ya que es el que más se ha utilizado y con el que se siente más cómodo.

2.2.4.2 Métodos de la clase.

- DriverManagerPara cargar un driver
- ConnectionPara establecer conexiones con las bases de datos
- StatementPara ejecutar sentencias SQL y enviarlas a las BBDD
- PreparedStatementLa ruta de ejecución está predeterminada en el servidor de base de datos que le permite ser ejecutado varias veces
- CallableStatementPara ejecutar sentencias SQL de Procedimientos Almacenados.
- ResultSetPara almacenar el resultado de la consulta

2.2.5 Java.mail

JavaMail es una API Java que facilita el envío y recepción de e-mail desde código java a través de protocolos SMTP, POP3 y IMAP. JavaMail está integrado en la plataforma Java EE, pero también proporciona un paquete opcional para su uso en Java SE.¹



La última versión liberada bajo la identificación de Java EE es la 1.6.2, publicada en agosto de 2018. Existe otra implementación JavaMail de código abierto - GNU

Imagen 2.8. JavaMail.

JavaMail - aunque sólo soporta la versión 1.3 de la especificación JavaMail, además solo proporciona un único backend gratuito de NNTP, que permite utilizar esta tecnología para leer y enviar artículos de grupos de noticias.

2.2.5.1 Características

JavaMail implementa el protocolo SMTP (Simple Mail Transfer Protocol) así como los distintos tipos de conexión con servidores de correo -TLS, SSL, autenticación con usuario y password, etc.

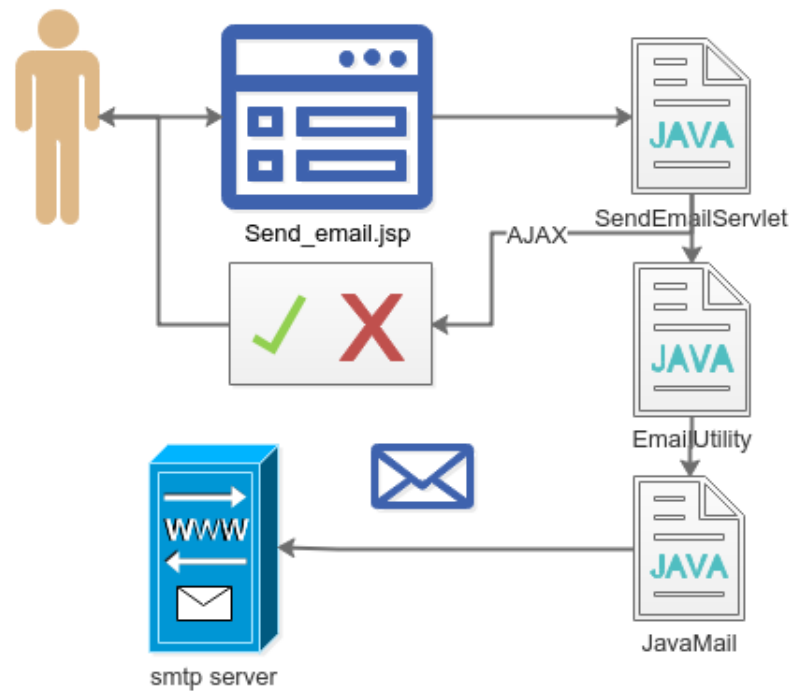


Imagen 2.9. Características de JavaMail.

JavaMail no se incluye en la JDK ni en la JRE, sino que debe conseguirse como un paquete externo. La última versión liberada es soportada con JDK 1.7 o superior. Debe, además, descargarse adicionalmente el JavaBeans Activation Framework en caso de usar una JDK inferior a la versión 6.

3. Fase de Pruebas.

En esta parte del desarrollo vamos a hacer una serie de pruebas para ver el correcto funcionamiento de la aplicación, también pediremos a personas externas al desarrollo del mismo que lo prueben y nos den su feed-back.

Dividiremos la fase de pruebas en dos partes; una en mitad del desarrollo y otra una vez acabado el desarrollo:

3.1. Pruebas dentro del desarrollo.

En esta parte de la fase de pruebas encontramos infinidad de pequeños errores que vamos solucionando poco a poco, a continuación, expondremos algunos ejemplos:

- Cuando estábamos creando el *Checkbox*, ocurría que al desmarcar el checkbox y al darle a iniciar sesión, la aplicación se cerraba, esto ocurría porque no se eliminaba correctamente los datos que se habían guardado anteriormente, ya que si estaba desmarcado porque nunca se había guardado datos, no ocurría este fallo.
- Otro de los fallos que se encontró es al darle a cancelar o a guardar una vez que se quiere introducir o actualizar los datos de la declaración de la renta ya al volver a la pantalla anterior no hacía de nuevo la comprobación para poner invisible según qué resultado saliese, esto ocurría porque el código que se utilizó que es `finish()`; terminaba el activity actual pero al volver al anterior volvía en el estado en el que se había dejado sin hacer la comprobación, se solucionó al poner un Intent para que vuelva a la pantalla anterior y haga todo lo que tenga esa pantalla en cuestión.

-
- Más adelante se encontró otro fallo al introducir el correo y enviar el código de verificación al darle a olvide contraseña, ya que aunque se introdujera el código que se había enviado no lo reconocía como el mismo, lo que se hizo para comprobar por qué pasaba esto, fue iniciar el proyecto en debug para saber si recogía bien el código o era otra cosa lo que fallaba, al iniciarse en ese modo se comprobó que lo enviaba correctamente pero aun así no lo reconocía como el mismo, entonces se pensó en un caso similar que ocurrió con una de las prácticas realizadas con anterioridad en el curso, por lo cual se optó a poner equals en lugar de los dos iguales, y como se suponía, de esta forma si funcionó.
 - El último fallo que se comprobó fue al introducir la nueva contraseña ya que al introducirla se actualizaba no solo la de ese usuario, sino que la de todos también, lo primero que se hizo fue comprobar en el código si se había escrito de forma correcta la consulta, ya que los errores tipográficos suelen ser bastante usuales, y puesto que estos fallos no los identifica de forma correcta Android Studio, se comprobó y efectivamente, el error era tipográfico ya que en la consulta no se había especificado de forma correcta a que usuario se le debía de actualizar la contraseña introducida, por lo que se cambió la consulta, esta vez añadiéndole el where para especificar el usuario, y de esa forma ya funcionaba correctamente.

3.2. Pruebas tras el desarrollo.

Estas pruebas las basamos en hacer que diferentes sujetos ajenos al desarrollo del proyecto prueben la aplicación y nos transmitan sus sensaciones. Puntualizar que los sujetos de pruebas son personas adultas que ya tienen familiaridad con el tema de declaraciones de la renta puesto que trabajan, algunos desde hace más de veinte años y otros desde hace poco más de cinco años.

- El primer sujeto de pruebas, encontró la adaptación al entorno y a los botones bastante sencilla e intuitiva, los elementos de la escena los visualiza con claridad y el funcionamiento de las mecánicas de la aplicación le pareció correcto en general. Al comenzar la experiencia consiguió hacer la declaración de la renta sin problemas. Como aspectos a mejorar, al sujeto de pruebas le hubiera gustado que hubiera un aspecto visual más vistoso y más opciones para la declaración de la renta.
- El segundo sujeto de pruebas, encontró el funcionamiento de la aplicación bastante intuitivo en general. Consiguió hacer la declaración de la renta correctamente sin ningún fallo. Como aspectos a mejorar, le hubiera gustado poder tener algunas explicaciones de los datos a introducir en la aplicación por lo demás le pareció todo correcto.
- El tercer sujeto de pruebas tuvo algunas dificultades para adaptarse al funcionamiento de la aplicación. En general le pareció correcto el funcionamiento de la aplicación, trató de hacer la declaración de la renta sin tener en cuenta la diferencia entre seguridad social y retenciones. Como aspectos a mejorar, el sujeto cree necesarias más indicaciones sobre como realizar correctamente la declaración de la renta y diferenciación de los datos del mismo.
-

4. Conclusiones y posibles mejoras.

Para finalizar, pasaremos a exponer las conclusiones; tanto personales como con respecto al proyecto en sí. Finalmente apuntaremos algunas mejoras en cuanto al desarrollo de la aplicación con vistas al futuro.

4.1. Conclusiones Personales.

Este proyecto ha sido fruto del trabajo y los conocimientos adquiridos gracias a estos dos últimos años y a la experiencia de manejo de varios aspectos de la aplicación, que me permitió estar aprendiendo sobre el mundo del desarrollo de aplicaciones para móviles, a la vez que adquiría experiencia en una empresa real.

Además de aprender distintos lenguajes y entornos de programación he podido experimentar de primera mano cómo es trabajar como desarrollador, lo cual ha sido una experiencia frenética a la vez que gratificante, desde el primer momento fui consciente del reto que esto suponía y que no había tiempo para relajarse. Este periodo ha sido de constante aprendizaje; y es con lo que me quedo de todo lo sucedido desde que comencé mi andadura en este sector que es nuevo para mí.

El primer contacto fue sin problemas puesto que a lo largo del curso se ha ensañado sobre todo a cómo desarrollarse y poder desenvolverse por uno mismo que es uno de los aspectos fundamentales para prácticamente cualquier aspecto de la vida por lo que no hubo dificultad a la hora de tener que pensar como requiere la programación, aun así se ha conseguido desarrollar aún más esta faceta y cada vez se domina más estos aspectos y otros, me siento preparado para entrar en el mercado laboral dentro del sector de las IT, que en definitiva era mi objetivo desde el principio.

4.2. Conclusiones sobre el proyecto.

Trabajar en un proyecto de desarrollo de una aplicación para móviles de manera totalmente individual ha supuesto todo un reto desde el primer momento. Desde las primeras etapas de desarrollo siempre he encontrado algún obstáculo que no me ponía las cosas nada fáciles, aunque poco a poco y con constancia y trabajo las cosas fueron saliendo adelante; en definitiva, estoy contento con lo que se ha conseguido en esta aplicación.

Este proyecto lo he desarrollado con vistas a que, en un futuro, se pueda seguir desarrollando la idea, además, el código ha sido diseñado siempre pensando en que sea lo más reutilizable posible. La documentación recogida en el transcurso del mismo, también se espera que sea de utilidad para futuros proyectos.

4.3. Posibles mejoras a futuro.

El planteamiento del proyecto se ha pensado para que en un futuro se puedan añadir diferentes elementos que puedan hacer de la aplicación una experiencia más completa. El factor tiempo ha sido siempre determinante para elegir una mayor o menor complejidad de la aplicación.

- ❖ Como primera mejora, se pensó en hacer varios tipos de datos a introducir para poder realizar las retenciones la misma aplicación para no tener que hacerlas con anterioridad e introducirla en la misma, de esta forma la aplicación sería más funcional en este aspecto y se acercaría más a la actualmente hecha por la agencia tributaria.
- ❖ Otra mejora podría ser introducir una pantalla nueva para poder realizar nómina y así tener más variedad en la aplicación para poder hacer más opciones y no solo la declaración de la renta, que aunque sea bastante útil, podría ser aún mejor.
- ❖ Una mejora interesante es mejorar el sistema de el envío de correo, para que en vez de que tenga que irse al correo y mirarlo, se pueda introducir automáticamente al

darle premisos a la aplicación, parecido a como lo hace la aplicación bancaria de BBVA.

- ❖ También añadir una pantalla nueva para poder saber que IRP se le tendría que aplicar a la Nómina para que el cliente tenga más controlado el tema de la declaración de la renta ya que esto influye en si le tiene que devolver o tuviese que pagar en cambio, por lo que sería una muy buena mejora a realizar en la aplicación.
- ❖ Por último, se podría añadir una última pantalla que lo que haría sería el cálculo necesario para tener el dato de la seguridad social y de esta forma ya no se tendría que traer el cliente prácticamente nada de fuentes externas para poder realizar la declaración de la renta y lo podría hacer prácticamente todo y algunas cosas más desde la propia aplicación y de esta forma quedaría una aplicación mucho más completa, aunque para añadir todos estos aspectos y anteriores dichos se le tendría que dedicar más tiempo no solo a la investigación para poder realizar todos estos aspectos, sino que también se le tendría que dedicar bastante tiempo al código en si y a las pruebas que se tendría que realizar para comprobar que todo en conjunto funcione correctamente.

5. Apéndices

5.1. Guía de instalación de la aplicación en un terminal Android.

En este apartado se explicará el proceso de instalación de la aplicación en un terminal Android. Lo que haremos será ir a la configuración del terminal y buscar la opción de permitir instalar aplicaciones de fuentes externas. Lo vemos en unas imágenes:

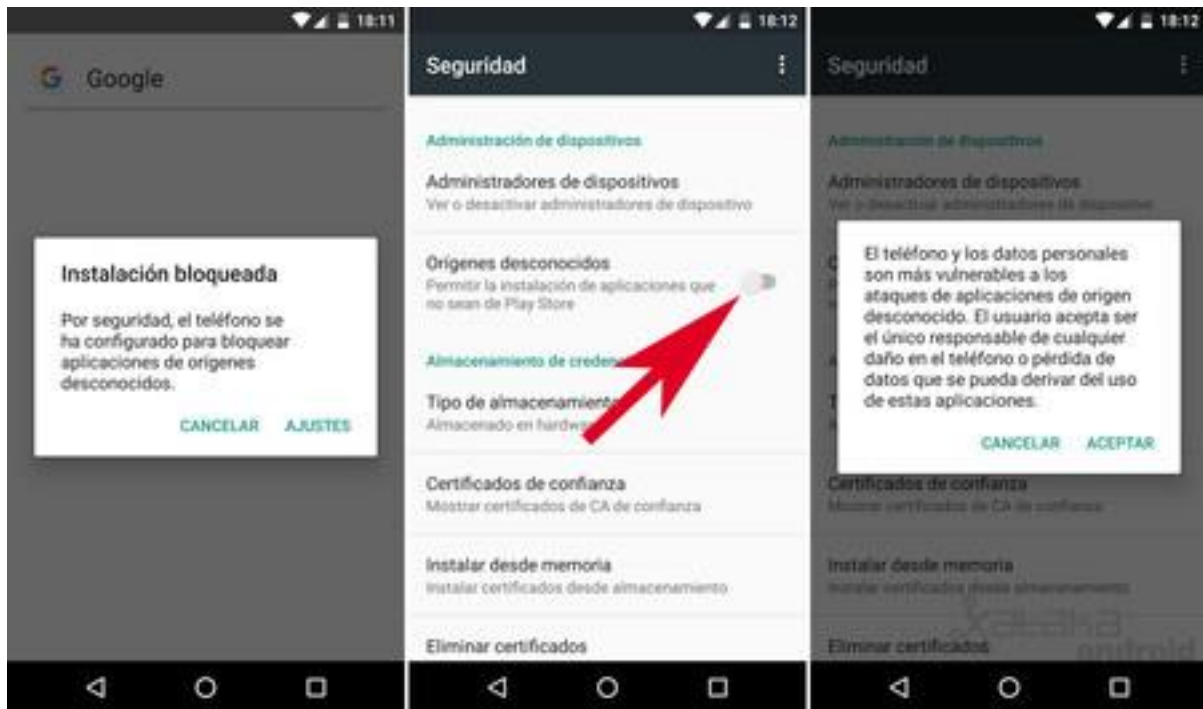


Imagen 5.1. Permitir instalar aplicaciones de fuentes externas.

Una vez hemos hecho esto, tendremos que poner la aplicación en google drive y descargar desde ahí de esta forma también se podrá descargar desde otras fuentes ya que se puede compartir archivos y elementos específicos en google drive.

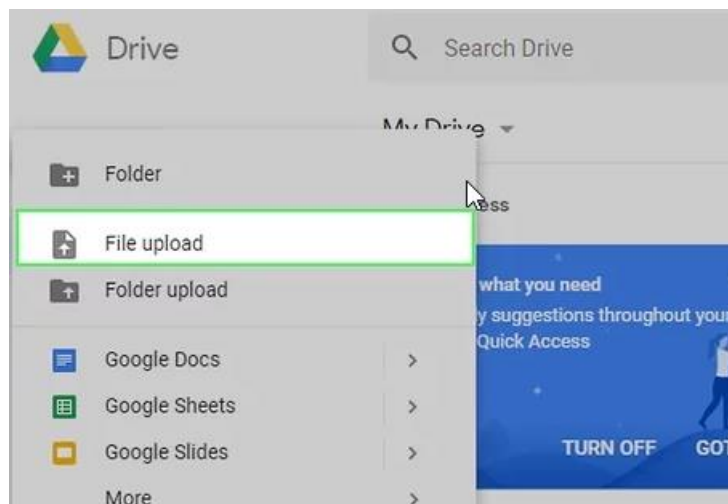


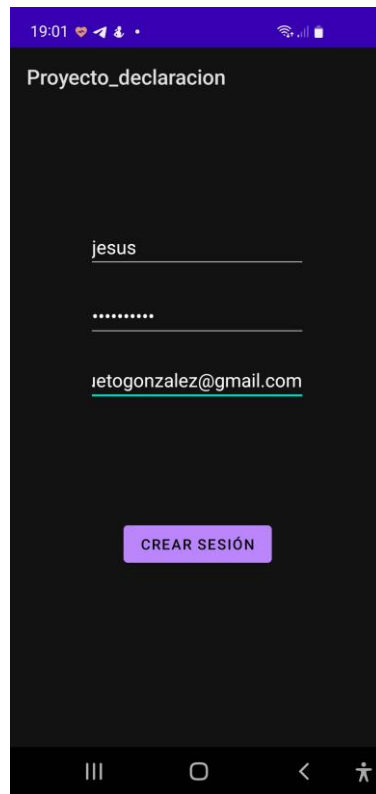
Imagen 5.2. Subida de archivos.

Por último, descargaremos la aplicación que hemos creado desde el enlace de google drive y la buscamos en el explorador de archivos de nuestro terminal para instalar la aplicación, una vez hecho todo esto, ya podremos abrir la aplicación sin ningún problema.

6. Manual de usuario

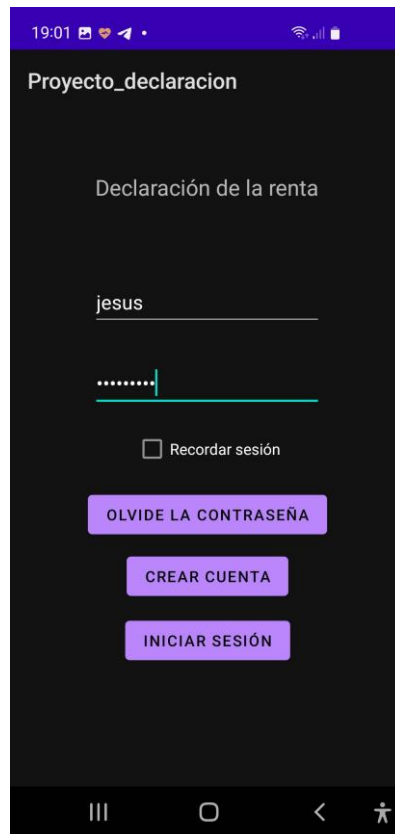
En este apartado se explicará de forma sencilla como utilizar la aplicación con todos sus aspectos y con fotos para su explicación.

1. Si no se tiene creado ninguna cuenta aún se puede crear en crear cuenta, una vez ahí se tendrá que introducir un usuario, una contraseña y un correo, para un uso que se explicará más adelante, que no existan ya en la base de datos.



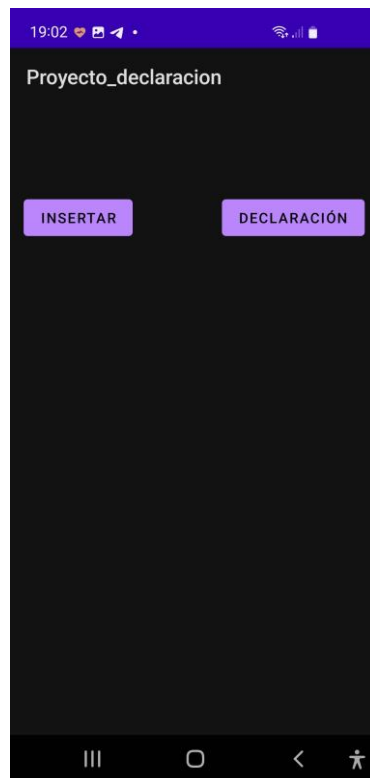
Imágen 6.1. Creación de cuenta.

2. Una vez creado el usuario ya puede iniciar sesión en la pantalla principal introduciendo el usuario y la contraseña creados anteriormente, y se le da a iniciar sesión, también se le puede dar a recordar datos para que la próxima vez que se entre en la aplicación no se tenga que introducir nuevamente los datos, si se quiere quitar los datos guardados solamente hay que desmarcarlo y ya está.



Imágen 6.2. Inicio de sesión.

3. Una vez logueado aparecerá una ventana que al ser la primera vez aparecerá dos botones; Insertar datos y declaración de la renta, ambos botones se explicarán en los siguientes apartados.



Imágen 6.3. Botones Insertar y declarar.

- Una vez se le dé a insertar datos aparecerán 3 campos a rellenar; salario bruto, seguridad social y retenciones, esos tres datos se han tenido que obtener anteriormente de forma externa para poder guardarlos aquí y realizar posteriormente el cálculo.



19:03

Proyecto_declaracion

21000

1463

1974

GUARDAR

CANCELAR

Imágen 6.4. Introducción de datos.

- Al guardar se enviará a la anterior ventana donde ya no estará el botón insertar datos sino que ahora aparecerá el de editar datos ya que se han introducido y existen datos relacionados con el usuario.



Imágen 6.6. Nueva ventana con datos

6. El botón de declaración de la renta lo que hace es un cálculo para comprobar si se tiene que pagar o tiene que devolver, según lo que salga pondrá una cosa u otra.

Sale a pagar 760.3800000000001

Imágen 6.7. Declaración de la renta.

7. Por ultimo está el botón de la ventana principal olvide la contraseña que en los siguientes apartados se explicara qué es lo que se tiene que hacer.

7.1 Primero se tiene que poner el usuario y el correo que se puso al crear la cuenta.

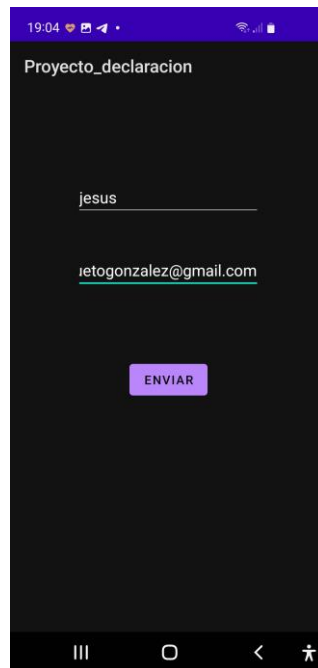
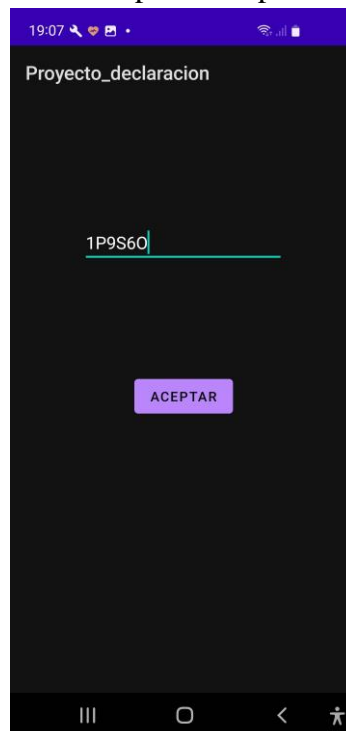
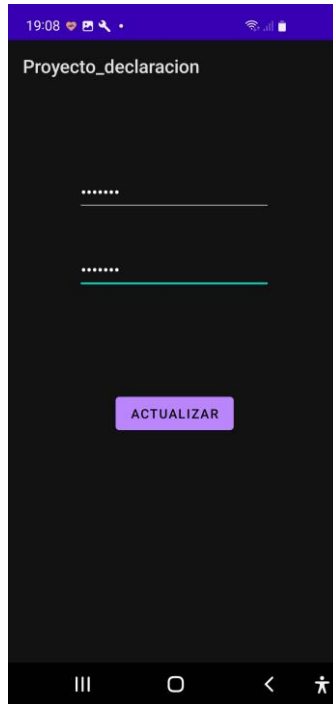


Imagen 6.8. Olvide la contraseña.

7.2 Lo segundo que se tiene que hacer es revisar el buzón de la cuenta de correos introducida, si no aparece en la bandeja principal estará en la bandeja de spam, una vez que se tiene el código que se ha enviado, se tendrá que introducir en la aplicación, si no es correcto porque por ejemplo se ha equivocado al introducirlo, se recomienda que se copie el código y se pegue en el campo de la aplicación.



7.3 Lo último que se tendrá que hacer es introducir la nueva contraseña y teniendo cuidado de que coincidan en ambos campos para que luego no haya confusiones en la contraseña.



Imágen 6.10. Nueva contraseña.

Aquí finaliza el manual de usuario creado lo más fácil y sencillo posible para que no haya ningún problema a la hora de ejecutar y probar la aplicación ya que es esencial que no pase esto, y sería muy poco profesional el no incluirlo, por lo que si se tiene algún problema no dude en consultarnos e intentaremos resolver todos sus dudas al respecto a la aplicación creada para realizar la declaración de la renta.

7. Bibliografía.

[Archivo librería Android](#)

[Declaración de la renta e IRPF](#)

[SharedPreferences](#)

[SqlServer](#)

[Toast](#)

[JDBC](#)

[JavaMail](#)