

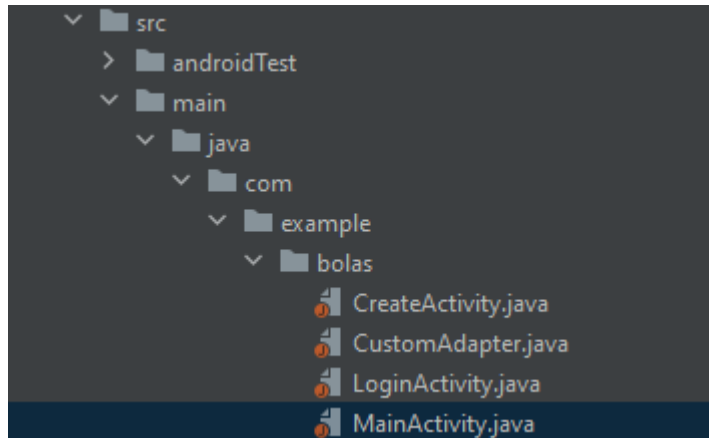
## Tasca 1.2: Creació d'una aplicació amb bases de dades



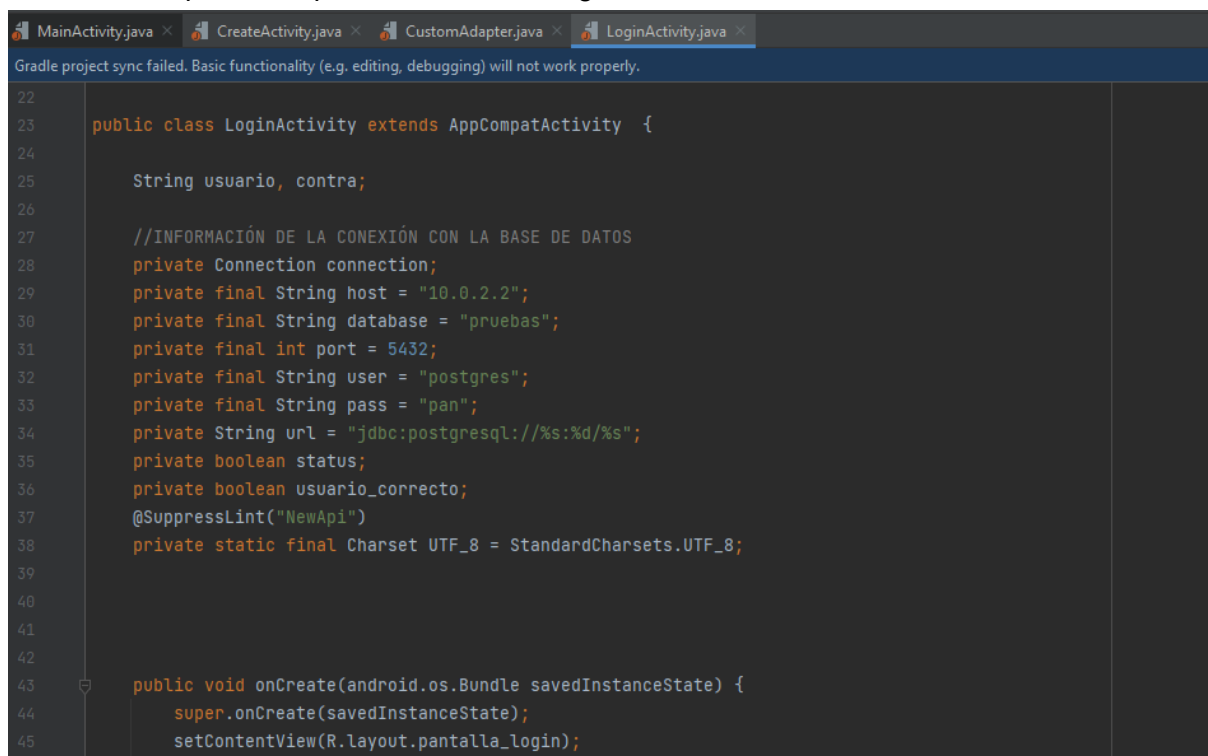
## **index**

<b>Part 1: fichers i classes</b>	<b>3</b>
<b>Part 2: base de dades</b>	<b>14</b>
<b>Part 3: joc de proves/demostracio d'us</b>	<b>16</b>

## Part 1: Fichers i Classes



Per a aquesta activitat, hem creat diferents classes de java per a tindre les funcions necessaries separades i per a millor control i gestio d'errors.



Començant per el login, al principi tenim les dades necessaries per a poder fer la conexio amb la bbdd.

```
public void onCreate(android.os.Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.pantalla_login);

    //DECLARAMOS TOASTS Y LOS CAMPOS DE TEXTO DEL INICIO DE SESIÓN
    EditText t_usuario = findViewById(R.id.editTextTextPersonName);
    EditText t_contra = findViewById(R.id.editTextTextPassword);
    Toast toast_bien = Toast.makeText(getApplicationContext(), "Sesión iniciada!", Toast.LENGTH_SHORT);
    Toast toast_mal = Toast.makeText(getApplicationContext(), "Usuario incorrecto.", Toast.LENGTH_SHORT);

    //DECLARAMOS EL BOTÓN CON UN ACTIONLISTENER EL CUAL, EN CASO DE QUE LA CONEXIÓN SEA CORRECTA Y HAYA UN USUARIO VÁLIDO, REDIRIGIRÁ AL MAIN
    Button b_iniciar_sesion = findViewById(R.id.button);
    b_iniciar_sesion.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v){
            usuario = String.valueOf(t_usuario.getText());
            contra = String.valueOf(t_contra.getText());
            prueba_conexion();
            if(usuario_correcto == true){
                toast_bien.show();

                startActivity(new Intent(LoginActivity.this, MainActivity.class));
            } else {
                toast_mal.show();
            }
        }
    });
}
```

Tenim un void el cual revisa si les dades de inici de sessio son correctes, i ens hi avisa amb un toast dependent del cas.

```
private void connect()
{
    EditText t_usuario = findViewById(R.id.editTextTextPersonName);
    EditText t_contra = findViewById(R.id.editTextTextPassword);

    Thread thread = new Thread(new Runnable() {
        @Override
        public void run()
        {
            try {
                Class.forName("org.postgresql.Driver");
                connection = DriverManager.getConnection(url, user, pass);
                status = true;
                Statement query = connection.createStatement();

                //ENCRIPТАMOS LA CONTRASEÑA CON MD5
                byte[] bytes_contra = digest(t_contra.getText().toString().getBytes(UTF_8));
                String contra_encryptada = bytesToHex(bytes_contra);

                //REALIZAMOS LA CONEXIÓN, SI DEVUELVE UN VALOR, EL LOGIN ES CORRECTO Y SE CARGA OTRA ACTIVIDAD, EN CASO CONTRARIO, SE MOSTRará UN TOAST DE FALLO Y NO INICIARÁ SESIÓN
                String sql = "SELECT prueba1 FROM usuarios WHERE usuario = '" + t_usuario.getText().toString() + "' AND contra = '" + contra_encryptada + "'";
                System.out.println(sql);
                ResultSet rs = query.executeQuery(sql);

                if (rs.next()) {
                    usuario_correcto = true;
                } else {
                    usuario_correcto = false;
                }

                System.out.println("connected:" + status);
            }
            catch (Exception e)
            {
            }
        }
    });
}
```

Tenim una funcio la qual ens realitza la conexio a la base de dades, amb contrasenya encryptada amb md5.

Dependent del cas, ens fara un syso de connected status o del error.

```
//FUNCIONES PARA CONVERTIR A MD5
private static byte[] digest(byte[] input) {
    MessageDigest md;
    try {
        md = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException e) {
        throw new IllegalArgumentException(e);
    }
    byte[] result = md.digest(input);
    return result;
}

private static String bytesToHex(byte[] bytes) {
    StringBuilder sb = new StringBuilder();
    for (byte b : bytes) {
        sb.append(String.format("%02x", b));
    }
    return sb.toString();
}
```

I aquí tenim les funcions per a poder encriptar/desencriptar la contrasenya.

```
57 //CONSTRUCTOR QUE INSERTARÁ EN LAS VARIABLES LA INFORMACIÓN QUE LE PASAMOS DESDE LA SELECT DEL MAIN
58 public CustomAdapter(Context ctx, String[] titulos, String[] asuntos, String[] usuarios, String[] ids, String[] tipos_elemento, String[] elementos, String[] fechas, String[] ubicaciones) {
59     this.url = String.format(this.url, this.host, this.port, this.database);
60     this.context = ctx;
61     this.titulos = titulos;
62     this.cuerpos = asuntos;
63     this.usuarios = usuarios;
64     this.ids = ids;
65     this.tipos_elemento = tipos_elemento;
66     this.elementos = elementos;
67     this.fechas = fechas;
68 }
```

En la clase adapter, tenim tot el que fa falta per a agafar les dades de les incidencies i llistar-les.

Començem amb les variables que enmagatzemaran la informacio de la ListView, variables per a la conexi3n a la base de dades, i el constructor a on inicialitzem les variables globals.

```
public View getView(int position, View convertView, ViewGroup parent) {
    convertView = LayoutInflater.inflate(R.layout.listview_incidentes, null);

    //CREAMOS LAS VARIABLES DE LOS CAMPOS DE TEXTO, IMAGENES, ETC.
    ImageView imageView = (ImageView) convertView.findViewById(R.id.icono);
    Button boton = (Button) convertView.findViewById(R.id.boton_cerrar);
    TextView textView = (TextView) convertView.findViewById(R.id.titulo);
    TextView textView1 = (TextView) convertView.findViewById(R.id.cuerpo);
    TextView textView2 = (TextView) convertView.findViewById(R.id.usuario_creador);
    TextView textView3 = (TextView) convertView.findViewById(R.id.id_incidencia);
    TextView textView4 = (TextView) convertView.findViewById(R.id.tipo_elemento);
    TextView textView5 = (TextView) convertView.findViewById(R.id.elemento);
    TextView textView6 = (TextView) convertView.findViewById(R.id.fecha_creacion);
    TextView textView7 = (TextView) convertView.findViewById(R.id.ubicacion);

    imageView.setImageResource(imagenes[position]);

    //AÑADIMOS UN ACTIONLISTENER QUE SIRVA PARA CERRAR LA INCIDENCIA Y ADEMÁS REFRESCA LA LISTA
    boton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            connect(ids[position], titulos[position]);
            Toast.makeText(context, "Se ha cerrado la incidencia.", Toast.LENGTH_SHORT).show();
            Intent myIntent = new Intent(context, MainActivity.class);
            myIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(myIntent);
        }
    });

    //INSERTAMOS LA INFORMACIÓN RECIBIDA EN LOS CAMPOS
    textView.setText("Título: " + titulos[position]);
    textView1.setText("Descripción: " + cuerpos[position]);
    textView2.setText("Usuario: " + usuarios[position]);
    textView3.setText("ID: " + ids[position]);
    textView4.setText("Tipo: " + tipos_elemento[position]);
    textView5.setText("Elemento: " + elementos[position]);
    textView6.setText("Fecha: " + fechas[position]);
    textView7.setText("Lugar: " + ubicaciones[position]);
}
```

Continuem amb un void a on creem les variables dels camps de text, un action listener que serveix per a tancar incidencies i refrescar la llista, insertar la informacio que toca en cada camp,...

```
//MÉTODO QUE SIRVE PARA CREAR UN CANAL DE NOTIFICACIONES
private void CreateNotificationChannel(){
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
        CharSequence nombre = "notifications";
        String descripcion = "canal de notificaciones";
        int importancia = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel notifications = new NotificationChannel("notifications", nombre, importancia);
        notifications.setDescription(descripcion);

        NotificationManager notificationManager = context.getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(notifications);
    }
}
```

Tambe tenim un metode que crea un canal de notifiacions, per a poder enviar una, quan es tanqui una de les incidencies.

```
private void connect(String id, String titulo)
{
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run()
        {
            try {
                Class.forName("org.postgresql.Driver");
                connection = DriverManager.getConnection(url, user, pass);
                status = true;
                Statement query = connection.createStatement();

                //UPDATE PARA CAMBIAR EL ESTADO DE LA INCIDENCIA A CERRADO, PILLAMOS LA ID DE UN PARÁMETRO QUE PASAMOS A TRAVÉS DE LA FUNCIÓN
                String sql = "UPDATE tickets SET activo = 'false' WHERE id = "+id;
                System.out.println(sql);
                query.executeUpdate(sql);

                //CREAMOS UNA NOTIFICACIÓN DE QUE SE HA CERRADO LA INCIDENCIA Y LA MANDAMOS
                CreateNotificationChannel();
                NotificationCompat.Builder builder = new NotificationCompat.Builder(context, "notifications")
                    .setSmallIcon(R.drawable.quebien)
                    .setContentTitle("Incidencia cerrada.")
                    .setContentText("La incidencia " + titulo + " con ID: " + id + " ha sido cerrada.")
                    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

                NotificationManagerCompat notificationManager = NotificationManagerCompat.from(context);
                notificationManager.notify(100, builder.build());

                System.out.println("connected:" + status);
            }
            catch (Exception e)
            {
                status = false;
                System.out.print(e.getMessage());
                e.printStackTrace();
            }
        }
    });
}
```

Tenim el void connect, que canvia l'estat d'una incidencia, i envia la notificacio a l'usuari per a avisar-lo de quina incidencia ha sigut tancada.



```
MainActivity.java  CreateActivity.java
Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work properly. Try Again Open Build

55 //DECLARACIÓN DE BOTONES Y SPINNER
56 Button campo_fecha = (Button) findViewById(R.id.seleccionar_fecha);
57 TextView textofecha = (TextView) findViewById(R.id.textofecha);
58
59 final Spinner dropdown = findViewById(R.id.i_tipo);
60 String[] items = new String[]{"Ordenador", "Servidor", "Portatil", "Otro"};
61 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_dropdown_item, items);
62 dropdown.setAdapter(adapter);
63
64
65 Button boton_cancelar = (Button) findViewById(R.id.b_cancelar);
66 boton_cancelar.setOnClickListener(new View.OnClickListener() {
67     public void onClick(View v){
68         startActivity(new Intent(CreateActivity.this, MainActivity.class));
69     }
70 });
71
72
73 Button boton_crear = (Button) findViewById(R.id.b_crearincidencia);
74 boton_crear.setOnClickListener(new View.OnClickListener() {
75     public void onClick(View v){
76         url = String.format(url, host, port, database);
77         crear_incidencia();
78     }
79 });
80
81
82 //ONCLICKLISTENER QUE NOS PERMITE GENERAR UNA INTERFAZ PARA SELECCIONAR UNA FECHA
83 campo_fecha.setOnClickListener(new View.OnClickListener() {
84     @Override
85     public void onClick(View v) {
86
87         final Calendar c = Calendar.getInstance();
88
89
90         int year = c.get(Calendar.YEAR);
91         int month = c.get(Calendar.MONTH);
92         int day = c.get(Calendar.DAY_OF_MONTH);
93
94         DatePickerDialog datePickerDialog = new DatePickerDialog(
```

Continuem amb la classe Create.

En aquesta classe declarem els elements visibles en pantalla com els botons, el spinner,...  
també un click listener per a generar una interfaz la qual permet seleccionar la data.

```
private void crear_incidencia()
{
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run()
        {
            try {
                Looper.prepare();
                Class.forName("org.postgresql.Driver");
                connection = DriverManager.getConnection(url, user, pass);
                status = true;
                Statement query = connection.createStatement();

                //SE SELECCIONAN TODOS LOS CAMPOS RELLENABLES Y SE RELLENAN VARIABLES CON LOS VALORES EN STRING DE ESTOS
                EditText i_usuario = findViewById(R.id.i_usuario);
                EditText i_elemento = findViewById(R.id.i_elemento);
                Spinner i_tipo = findViewById(R.id.i_tipo);
                EditText i_ubicacion = findViewById(R.id.i_ubicacion);
                EditText i_asunto = findViewById(R.id.i_asunto);
                EditText i_descripcion = findViewById(R.id.i_descripcion);
                TextView textofecha = findViewById(R.id.textofecha);

                String t_usuario = String.valueOf(i_usuario.getText());
                String t_elemento = String.valueOf(i_elemento.getText());
                String t_tipo = String.valueOf(i_tipo.getSelectedItem());
                String t_ubicacion = String.valueOf(i_ubicacion.getText());
                String t_asunto = String.valueOf(i_asunto.getText());
                String t_descripcion = String.valueOf(i_descripcion.getText());
                String t_textofecha = String.valueOf(textofecha.getText());

                System.out.println(t_usuario.length());

                //COMPROBACIONES DE QUE LOS CAMPOS TENGAN INFORMACIÓN
                if(t_usuario.length() == 0){
                    Toast.makeText(getApplicationContext(), "El campo de usuario no puede estar vacío.", Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
}
```

Continuant, tenim un metode per a crear incidencies, i que revisa si els camps tenen informacio o estan buits.

```
//COMPARAMOS LA FECHA DE HOY CON LA DE LA INCIDENCIA
String timeStamp = new SimpleDateFormat("yyyy-MM-dd").format(Calendar.getInstance().getTime());
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
Date fecha_hoy = sdf.parse(timeStamp);
Date fecha_incidencia = sdf.parse(t_textofecha.replace("Fecha: ", ""));

//SI LA FECHA ES SUPERIOR A LA DE HOY, SE DEVUELVE UN ERROR
if (fecha_incidencia.after(fecha_hoy)) {
    Toast.makeText(getApplicationContext(), "La fecha tiene que ser anterior o actual.", Toast.LENGTH_SHORT).show();
} else {
    //EN CASO DE QUE TODAS LAS COMPROBACIONES TENGAN ÉXITO, INSERTAMOS LA INCIDENCIA EN LA BASE DE DATOS
    String sql = "INSERT INTO tickets (usuario, elemento, tipo_elemento, ubicacion, asunto, cuerpo, fecha, activo) VALUES ('" + t_usuario + "', '" + t_elemento + "', '" + t_tipo + "', '" + t_ubicacion + "', '" + t_asunto + "', '" + t_descripcion + "', '" + t_textofecha + "', 1)";
    System.out.println(sql);
    query.executeUpdate(sql);

    //RECARGAMOS EL LISTADO DE INCIDENCIAS
    Toast.makeText(getApplicationContext(), "Se ha creado la incidencia.", Toast.LENGTH_SHORT).show();
    startActivity(new Intent(CreateActivity.this, MainActivity.class));
}
}
catch (Exception e)
{
    status = false;
    System.out.print(e.getMessage());
    e.printStackTrace();
}
}
```

Tambe compara la data de la incidencia amb l'actual, revisant si la data es igual o menor a la actual, i donant un error si la data es superior.  
I també recarga la llista d'incidencies.

```
//SIRVE PARA CREAR EL CANAL DE NOTIFICACIONES
private void CreateNotificationChannel(){
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
        CharSequence nombre = "notifications";
        String descripcion = "canal de notificaciones";
        int importancia = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel notifications = new NotificationChannel("notifications",nombre,importancia);
        notifications.setDescription(descripcion);

        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(notifications);
    }
}
```

I per ultim en aquesta classe, tenim una funcio que crea el canal de notifiacions.

```
//CREAMOS LA VARIABLE QUE ALMACENARÁ NUESTRA LISTVIEW
private ListView lv;

public void onCreate(android.os.Bundle savedInstanceState) {
    //CARGAMOS NUESTRO LAYOUT
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    //INICIALIZAMOS EL TEXTO SI NO HAY INCIDENCIAS Y EL BOTÓN DE REFRESCAR
    Button boton_refrescar = findViewById(R.id.b_refrescar);
    TextView texto_sin = findViewById(R.id.sin_incidencias);

    //ASIGNAMOS UN ONCLICK LISTENER AL BOTÓN DE ACTUALIZAR
    boton_refrescar.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v){
            startActivity(new Intent(MainActivity.this, MainActivity.class));
        }
    });

    //INICIALIZAMOS EL BOTÓN DE CREAR Y LE ASIGNAMOS UN ONCLICKLISTENER
    Button boton_crear = findViewById(R.id.b_crear);

    boton_crear.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v){
            startActivity(new Intent(MainActivity.this, CreateActivity.class));
        }
    });

    //CREAMOS LA CONEXIÓN QUE POBLARÁ NUESTRA LISTVIEW Y LE PASAMOS EL CONTEXTO NECESARIO
    prueba_conexion(texto_sin);
}

private void prueba_conexion(TextView texto_sin)
{
    //CREAMOS LA URL Y LA FORMATAMOS PARA QUE ESTA SEA USABLE
    this.url = String.format(this.url, this.host, this.port, this.database);
}
```

Per acabar, tenim la classe principal: MainActivity.

En aquesta posem en marxa els mètodes vistos anteriorment, com per exemple carregar el layout, click listener per a actualitzar, boto de crear, la conexio que ens deixara veure els elements per pantalla,...

```
private void connect(TextView texto_sin)
{
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run()
        {
            try {
                //CREAMOS E INICIAMOS LA CONEXIÓN
                Class.forName("org.postgresql.Driver");
                connection = DriverManager.getConnection(url, user, pass);
                status = true;
                Statement query = connection.createStatement();

                //CONTAMOS CUANTAS INCIDENCIAS HAY
                String sql = "SELECT COUNT(*) FROM tickets WHERE activo = 'true'";
                ResultSet rs = query.executeQuery(sql);

                int longitud = 0;

                while (rs.next()){
                    longitud = rs.getInt(1);
                }

                //EN CASO DE HABER 1 O MÁS, EL TEXTO DE QUE NO HAY INCIDENCIAS NO SE MUESTRA
                if(longitud != 0){
                    texto_sin.setText("");
                }

                //CREAMOS ARRAYS QUE ALMACENARÁN LA INFORMACIÓN PARA LA LISTVIEW
                String[] titulos = new String[longitud];
                String[] cuerpos = new String[longitud];
                String[] usuarios = new String[longitud];
                String[] ids = new String[longitud];
                String[] tipos_elemento = new String[longitud];
                String[] elementos = new String[longitud];
                String[] fechas = new String[longitud];
                String[] ubicaciones = new String[longitud];
                int[] imagenes = new int[longitud];
            }
        }
    });
}
```

També tenim un mètode que ens escriu per pantalla si no hi ha incidències, o en cas de que hi hagi una o més, en comptes del text es llistaran les incidències.

Afegim la creació d'arrays per a emmagatzemar la informació de les incidències, per a després poder llistar-les d'una forma més fàcil.

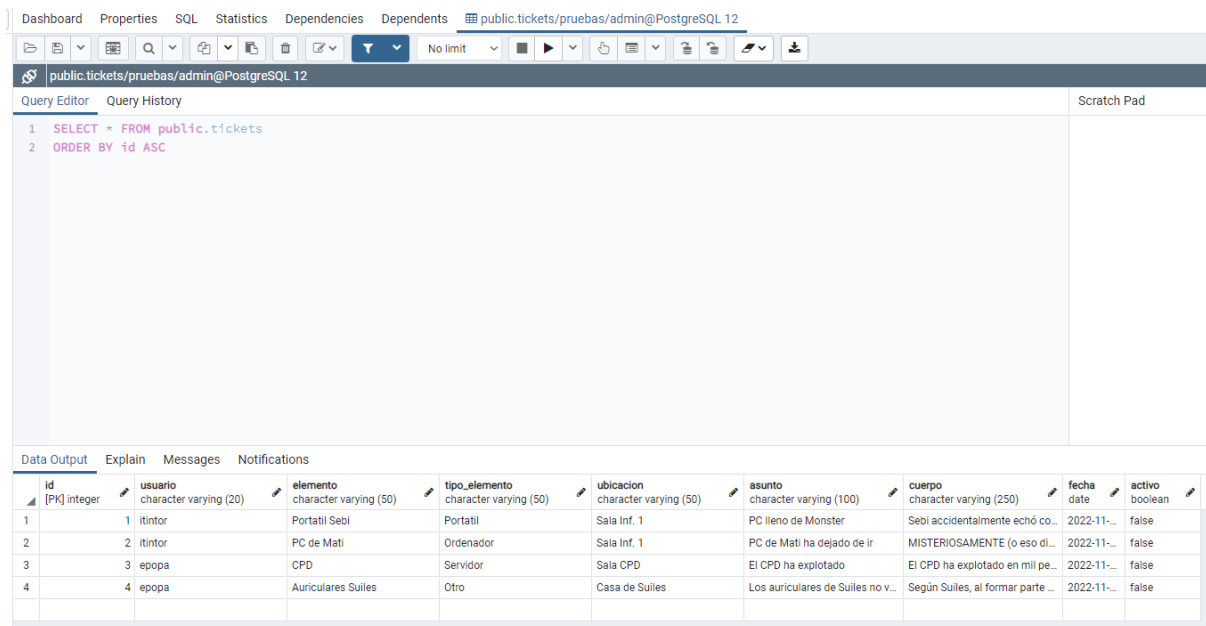
```
//EN CASO DE ENCONTRAR REGISTROS, INSERTAMOS EN LOS ARRAYS LA INFORMACIÓN
while (rs.next()) {
    titulos[contador] = rs.getString("asunto");
    cuerpos[contador] = rs.getString("cuerpo");
    usuarios[contador] = rs.getString("usuario");
    ids[contador] = String.valueOf(rs.getInt("id"));
    tipos_elemento[contador] = rs.getString("tipo_elemento");
    elementos[contador] = rs.getString("elemento");
    fechas[contador] = rs.getString("fecha");
    ubicaciones[contador] = rs.getString("ubicacion");
    if(rs.getString("tipo_elemento").equals("Servidor")){
        imagenes[contador] = R.drawable.servidor;
    } else if(rs.getString("tipo_elemento").equals("Ordenador")) {
        imagenes[contador] = R.drawable.dejadelpc;
    } else if(rs.getString("tipo_elemento").equals("Portatil")) {
        imagenes[contador] = R.drawable.portatil;
    } else {
        imagenes[contador] = R.drawable.quecojoneseseso;
    }
    contador++;
}

//LE INTRODUCIMOS AL LISTVIEW LOS REGISTROS PARA QUE LOS INTERPRETE
CustomAdapter customAdapter = new CustomAdapter(getApplicationContext(), titulos, cuerpos, usuarios, id$, tipos_elemento, elementos, fechas, ubicaciones, imagenes);
lv.setAdapter(customAdapter);

System.out.println("connected:" + status);
}
catch (Exception e)
{
    status = false;
    System.out.print(e.getMessage());
    e.printStackTrace();
}
```

Per a cada incidencia es llistara la seva informacio, i per ultim es mostrara en la interfaz d'usuari

## Part 2: Base de Dades



The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The main area displays a query: `SELECT * FROM public.tickets ORDER BY id ASC`. Below the query editor, there is a 'Data Output' section showing the results of the query. The results are displayed in a table with 10 columns: id, usuario, elemento, tipo\_elemento, ubicacion, asunto, cuerpo, fecha, and activo. The table contains 4 rows of data.

id	usuario	elemento	tipo_elemento	ubicacion	asunto	cuerpo	fecha	activo
1	itintor	Portatili Sebi	Portatili	Sala Inf. 1	PC lleno de Monster	Sebi accidentalmente echó co...	2022-11-...	false
2	itintor	PC de Mati	Ordenador	Sala Inf. 1	PC de Mati ha dejado de ir	MISTERIOSAMENTE (o eso di...	2022-11-...	false
3	epopa	CPD	Servidor	Sala CPD	Ei CPD ha explotado	Ei CPD ha explotado en mil pe...	2022-11-...	false
4	epopa	Auriculares Suiles	Otro	Casa de Suiles	Los auriculares de Suiles no v...	Según Suiles, al formar parte ...	2022-11-...	false

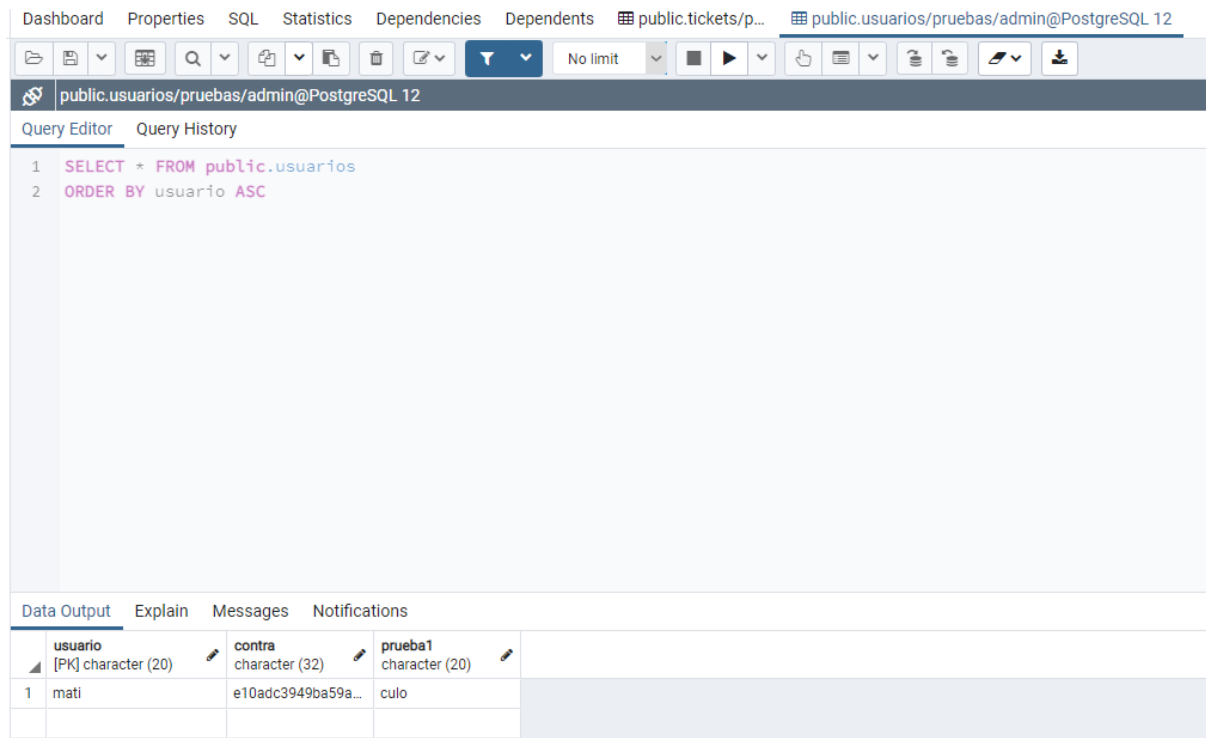
La estructura de la nostre base de dades implementada es bastant simple, es traca de dues taules:

Una taula per a les incidencies amb els següents camps:

- id: un camp autoincremental, el qual també es la clau primaria
- usuario: un camp en el que posem el nom de l'usuari afectat
- elemento: aquí s'introdueix el nom de l'element afectat
- tipo\_elemento: enmagatzema en la bbdd el tipus d'element donat en el desplegable
- ubicacion: ubicacio dintre "l'empresa", tipus sala o departament
- asunto: descripcio breu del problema
- cuerpo: descripcio completa del problema
- fecha: camp de tipus date que enmagatzema la data del problema
- activo: un boolea, el qual utilitzem de la següent forma:

Si activo esta en false significa que la incidencia ja ha sigut arreglada,

En canvi si esta en true, voldra dir que no ho esta, i sortira llistada en la app.



The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The current tab is SQL, and the query editor is active. The query being executed is:

```
1 SELECT * FROM public.usuarios
2 ORDER BY usuario ASC
```

Below the query editor, the 'Data Output' tab is selected, showing the results of the query. The results are displayed in a table with three columns: 'usuario' (PK character (20)), 'contra' (character (32)), and 'prueba1' (character (20)). The first row of data is:

	usuario [PK] character (20)	contra character (32)	prueba1 character (20)
1	mati	e10adc3949ba59a...	culo

D'altra banda, tenim una taula per als usuaris, per poder fer el login, a on podem veure el nom d'usuari i el password encriptat en md5.



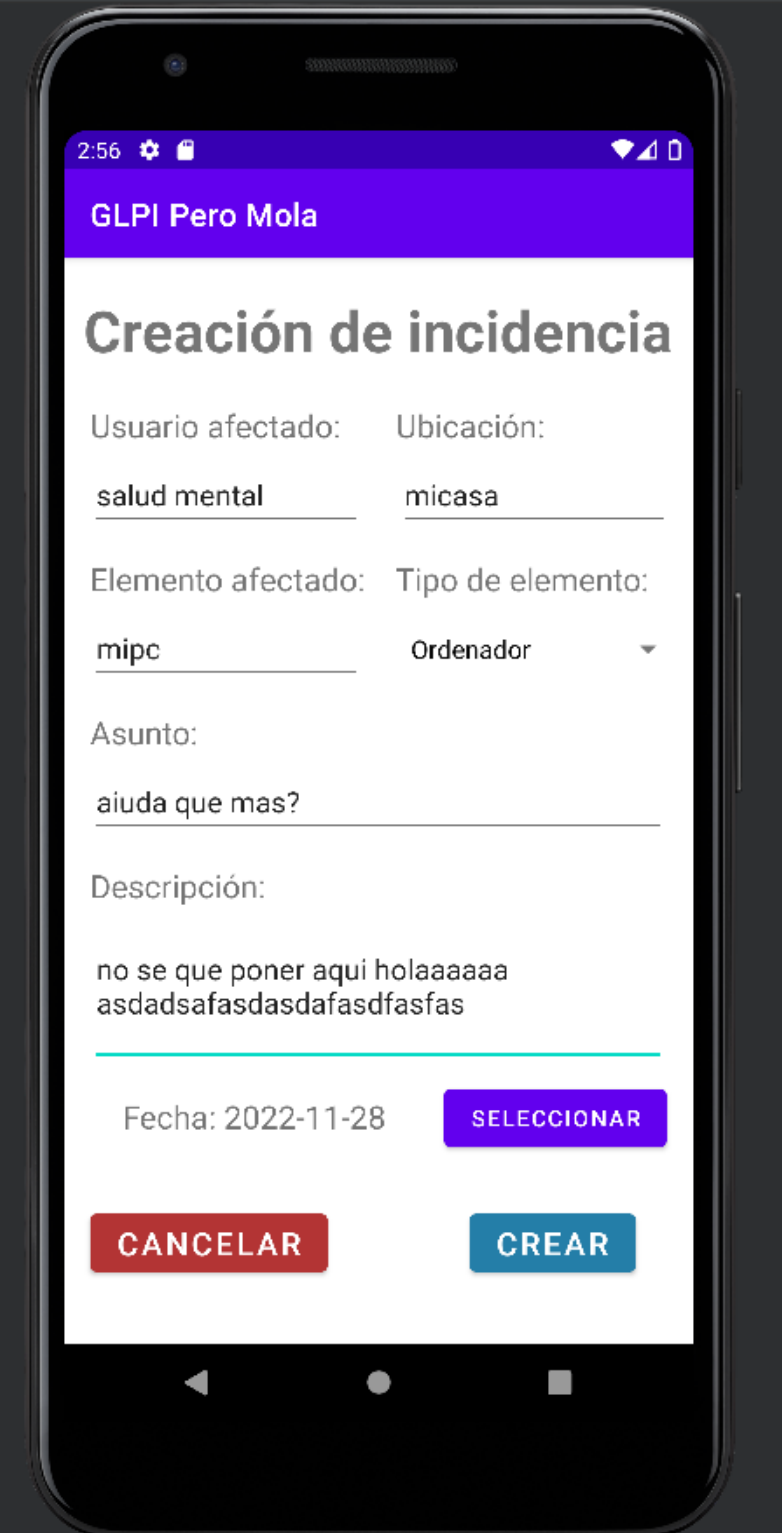
## Part 3: Joc de Proves/Demostracio d'Us



La primera pantalla que podem trobar al obrir la app es el login.



Al posar les dades de login ens sortira una pantalla a on podrem veure les incidencies.  
En aquest cas no n'hi ha, així que ens surt un missatge dient que no hi ha incidencies.



2:56

GLPI Pero Mola

## Creación de incidencia

Usuario afectado: salud mental Ubicación: micasa

Elemento afectado: mipc Tipo de elemento: Ordenador

Asunto: aiuda que mas?

Descripción:

no se que poner aqui holaaaaaa  
asdadsafasdasdafasdfasfas

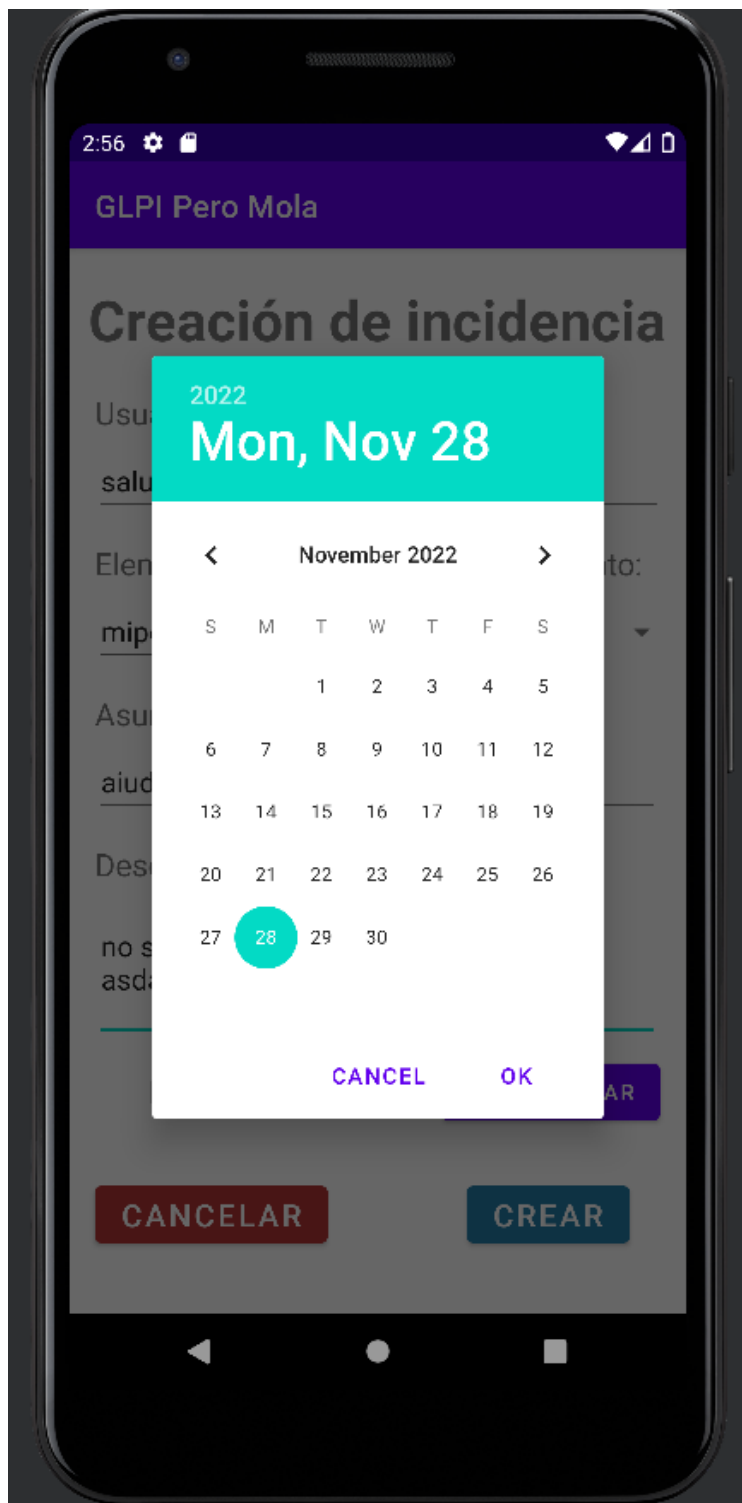
Fecha: 2022-11-28

SELECCIONAR

CANCELAR

CREAR

Per a crear una incidencia s'han d'omplir els camps, en cas contrari ens sortira un toast dient que l'hem d'omplir.



Per a la data, ens sortira un element semblant a un pop-up, a on haurem de seleccionar la data.

Fecha: 2022-11-30

SELECCIONAR

CANCELAR

CREAR

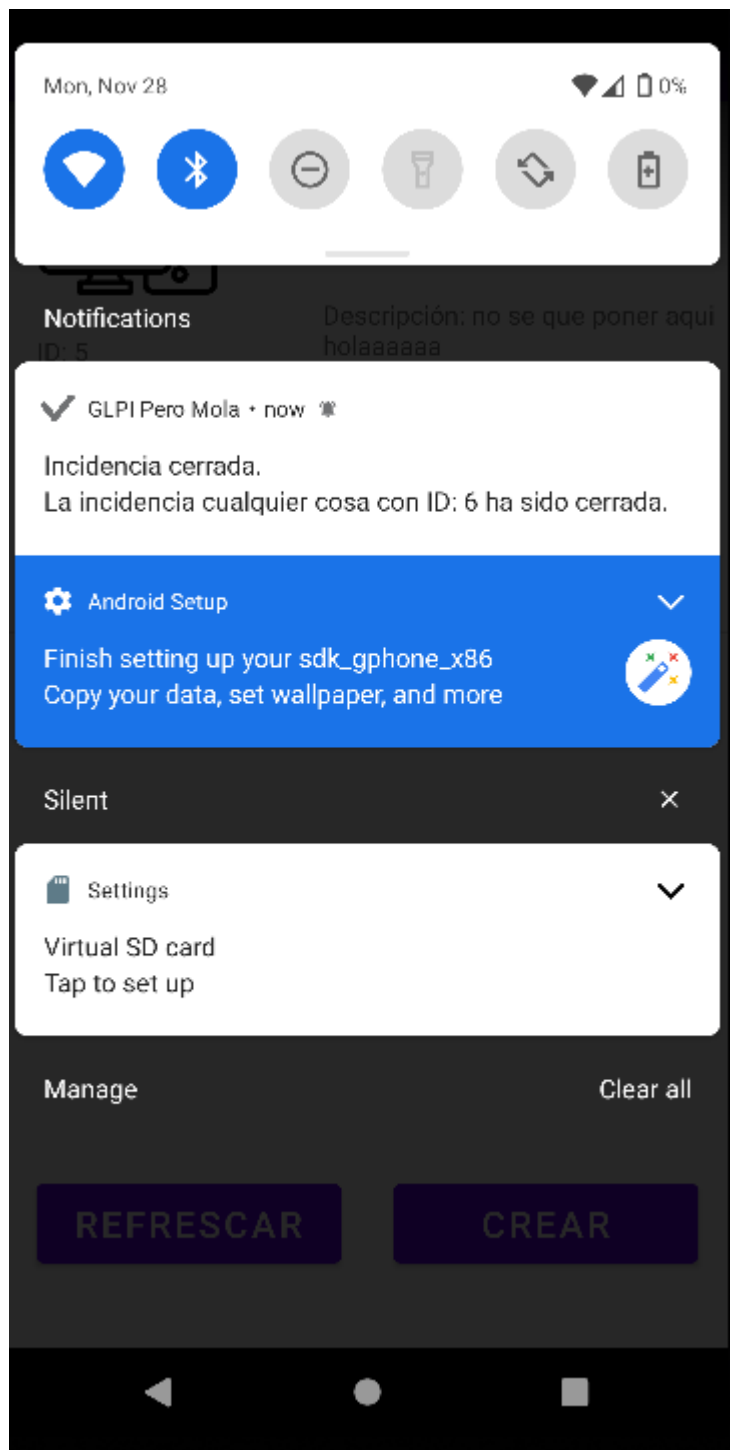
La fecha tiene que ser anterior o actual.

---

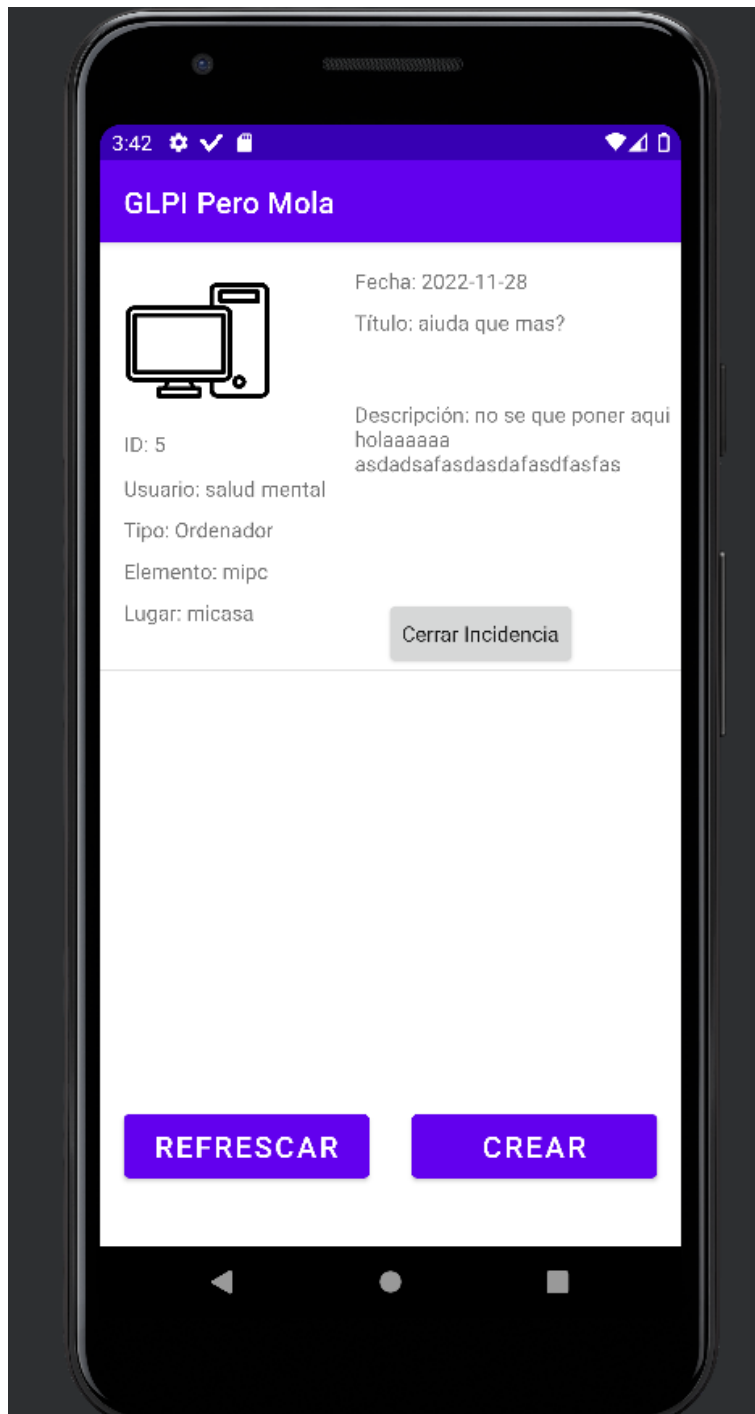
Si la data es superior a la data actual, no s'ens deixara crear el problema i ens sortira un toast avisant del nostre error.



Al tindre incidencias, les podrem veure d'aquesta forma.  
depenent de quin tipus d'element es tracta l'incidencia ens sortira una icona o una altra.



Al tancar una incidencia, podrem veure una notificacio la cual ens confirma que el problema ha sigut tancat.



I aqui podem veure com al tancar una incidencia ja no s'ens llistara mes en la app.