Benemérita Universidad Autónoma de Puebla

Facultad Ciencias de la Computación

Proyecto

Lectura de QR y asistencia

Desarrollo de aplicaciones móviles

Alumna:

Fabiola Castillo Cuahuizo

NRC:50009

Primavera 2025

Contenido

Introducción

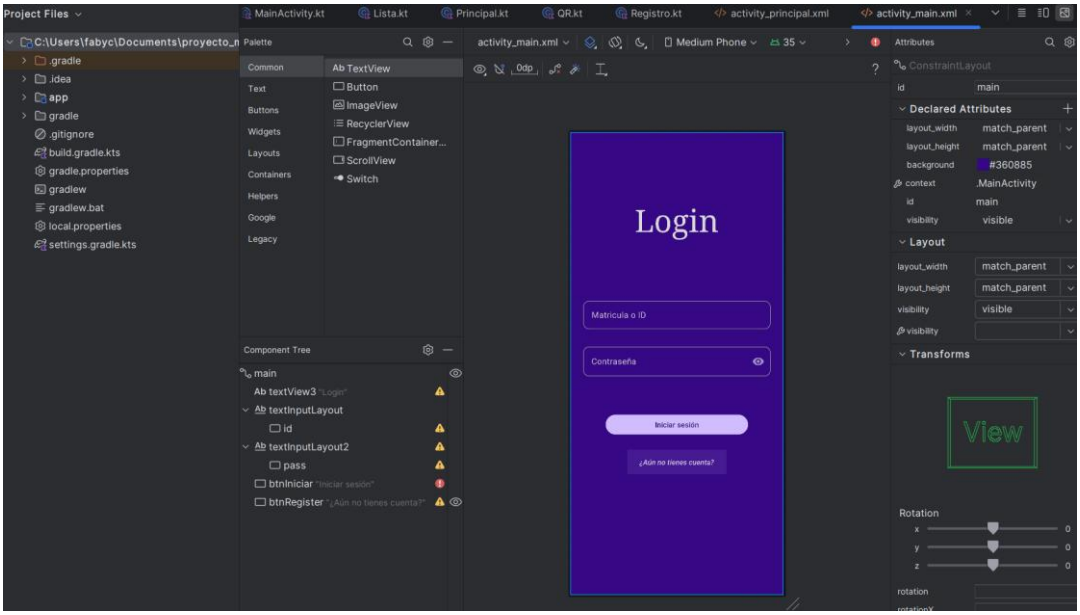
Para el proyecto se solicitó una aplicación móvil en la cual a través de la lectura de códigos qr se pueda tomar asistencia y esta se almacene en la base de datos, así como solo permitir una asistencia por día, por lo que si ya tomo asistencia ya no se puede almacenar otra asistencia.

Para realizar esta aplicación se usó Android studio para utilizar el lenguaje de programación kotlin, para la base de datos de uso MySQL a través de XAMPP, así como se utilizo Visual Studio para realizar los archivos php que iban a mantener la conexión con la base de datos.

Desarrollo

> Login

Se desarrollo la interfaz del login, la cual va a permitir el acceso a la aplicación



El funcionamiento se realizo en lenguaje Kotlin

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
    }
    fun clickBtnIniciar(view: View){

        if (id?.text.isNullOrEmpty() || pass?.text.isNullOrEmpty()) {
            Toast.makeText( context: this,  text: "Todos los campos son obligatorios", Toast.LENGTH_LONG).show()
            return
        }

        //val url= "http://192.168.100.38/Proyecto_moviles/login.php"
        val url= "http://172.21.232.56/Proyecto_moviles/login.php"
        val col: RequestQueue = Volley.newRequestQueue( context: this)

        val validarPost = object : StringRequest(Request.Method.POST, url,
            Response.Listener<String> { response ->
                try{
                    // Depurar el contenido de la respuesta
                    //Toast.makeText(this, "Re: $response", Toast.LENGTH_LONG).show()
                    val jsonResponse = JSONObject(response)
                    val success = jsonResponse.getBoolean( name: "success")
                    if(success){
                        val principal = Intent( packageContext: this, Principal::class.java)
                        startActivity(principal) // Iniciar la nueva actividad
                    }else{
                        Toast.makeText( context: this,  text: "Usuario o contraseña incorrectos", Toast.LENGTH_LONG).show()
                    }
                }catch (e : Exception){
                    Toast.makeText( context: this,  text: "Error en los datos", Toast.LENGTH_LONG).show()
                }
            }, Response.ErrorListener { error ->
                Toast.makeText( context: this,  text: "Error $error", Toast.LENGTH_LONG).show()
            }
        ) {
```

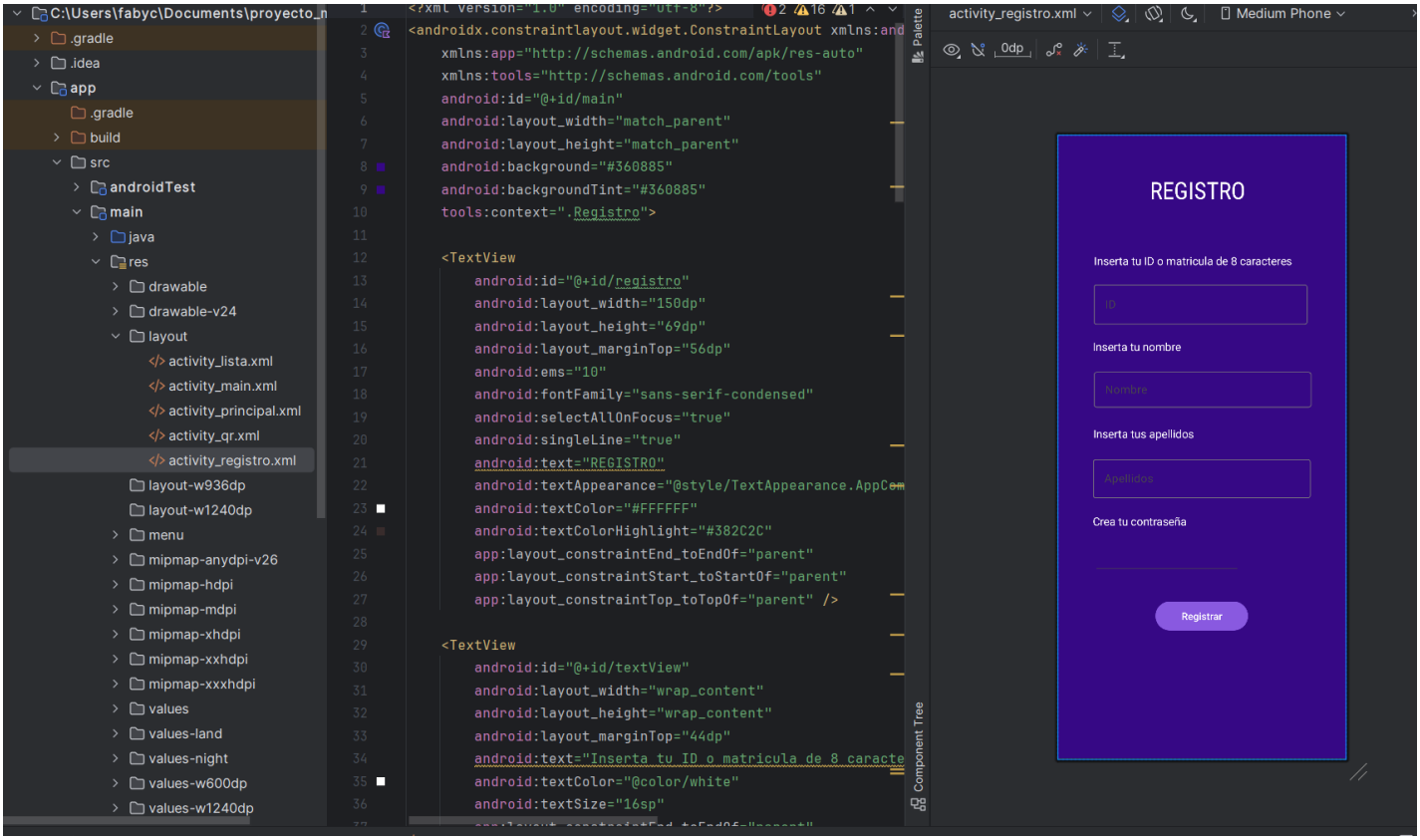
```
) {
    override fun getParams(): Map<String, String>? {
        return mapOf(
            "ID" to id?.text.toString(),
            "pass" to pass?.text.toString()
        )
    }
}
col.add(validarPost)
```

Se implemento una función llamada clickBtnIniciar la cual funciona si se da clic al botón de Iniciar Sesión, de inicio se ocupa una librería llamada volley la cual va a servir para hacer la comunicación con php y esta se pueda conectar a la base de datos, por lo que se coloca el link del archivo php que va a realizar todo el proceso de verificar si esta el usuario en la base de datos o si el usuario y contraseña están correctamente. La forma de ingreso de datos es a través de método post, y lo que va a hacer es mandar a través de método post el usuario y contraseña que el usuario ingreso, se procesa la información en el archivo php y si es correcto va a mandar una respuesta Json (success->true) de esta forma permitiendo que el usuario pueda acceder a la ventana principal, en caso de que la respuesta sea (success->false) este mandará un mensaje de que el usuario o contraseña es incorrecto.

login.php > ...

```php
<?php
    require_once 'conexion.php';
    $UserId = $_POST['ID'];
    $Password= $_POST['pass'];

    $stmt = $conexion->prepare("SELECT * FROM registroapp WHERE ID = ? AND pass = ?");
    $stmt->bind_param("ss", $UserId, $Password);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        echo json_encode(["success" => true]);
    } else {
        echo json_encode(["success" => false]);
    }

    $stmt->close();
    $conexion->close();
?>
```

> Registro



Se realizó el diseño del registro, este es una ventana que se original a dar clic en el botón de registrarse que se encuentra en el login.

En la función clickBtnInsertar, primero se delimita que no puede delimita que si hay alguna variable sin llenar no pueda continuar, es decir que para que se pueda registrar deben estar todos los campos completos.

En esta ocasión la conexión php que hay es a través del link del archivo registro.php, una vez obtenidos los parámetros que el usuario ingreso para realizar su registro estos se mandan a través de método POST, por lo que en el archivo php se realiza la conexión con la base de datos y estos se almacenan en una tabla llamada registroapp, si se guardaron correctamente manda un mensaje de que el usuario fue insertado exitosamente y lo manda de vuelta a la página de login.

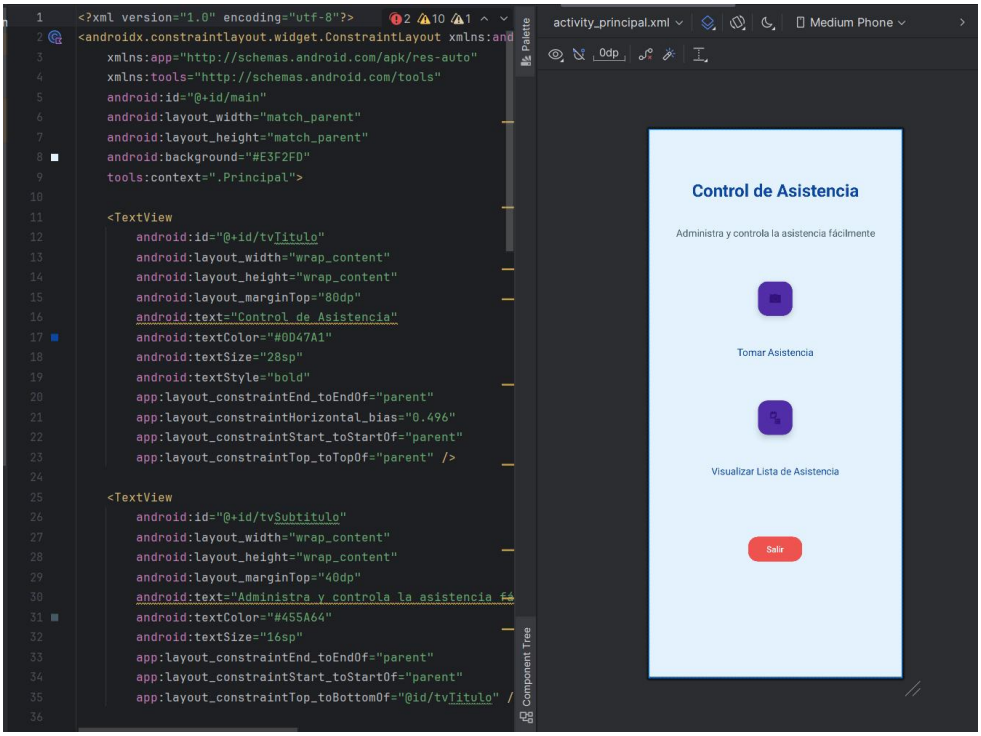
```kotlin
fun clickBtnInsertar(view:View){
    if (ID?.text.isNullOrEmpty() || nombre?.text.isNullOrEmpty() || apellidos?.text.isNullOrEmpty() || password?.text.isNullOr
        Toast.makeText( context: this,  text: "Todos los campos son obligatorios", Toast.LENGTH_LONG).show()
        return
    }

    //val URL= "http://192.168.100.38/Proyecto_moviles/registro.php"
    val URL= "http://172.21.232.56/Proyecto_moviles/registro.php"
    val cola: RequestQueue = Volley.newRequestQueue( context: this)
    val resultadoPost = object : StringRequest(Request.Method.POST, URL,

        Response.Listener<String> { response ->
            Toast.makeText( context: this,  text: "Usuario insertado exitosamente", Toast.LENGTH_LONG).show()
            val back = Intent( packageContext: this, MainActivity::class.java)
            startActivity(back) // Iniciar la nueva actividad
            finish()

        }, Response.ErrorListener { error ->
            Toast.makeText( context: this,  text: "Error $error", Toast.LENGTH_LONG).show()
        }
    ){
        override fun getParams(): MutableMap<String, String>? {
           val parametros = HashMap<String, String>()
           parametros.put("ID", ID?.text.toString())
            parametros.put("nombre", nombre?.text.toString())
            parametros.put("apellidos", apellidos?.text.toString())
            parametros.put("pass", password?.text.toString())
            return parametros
        }
    }
    cola.add(resultadoPost)
}
```

registro.php | conexion.php | login.php | asistencia.php | lista.php

registro.php > ...

```php
<?php
if ($_SERVER["REQUEST_METHOD"]=="POST")
    require_once 'conexion.php';
    $ID =$_POST["ID"];
    $nombre=$_POST["nombre"];
    $apellidos=$_POST["apellidos"];
    $pass=$_POST["pass"];

    $query= "INSERT INTO registroapp
                (ID, nombre, apellidos, pass)
            VALUES
                ('".$ID."', '".$nombre."', '".$apellidos."', '".$pass."')";

    if ($conexion->query($query) === TRUE) {
        echo "Usuario insertado exitosamente";
    } else {
        echo "Error: " . $query . "<br>" . $conexion->error;
    }

?>
```

> Principal

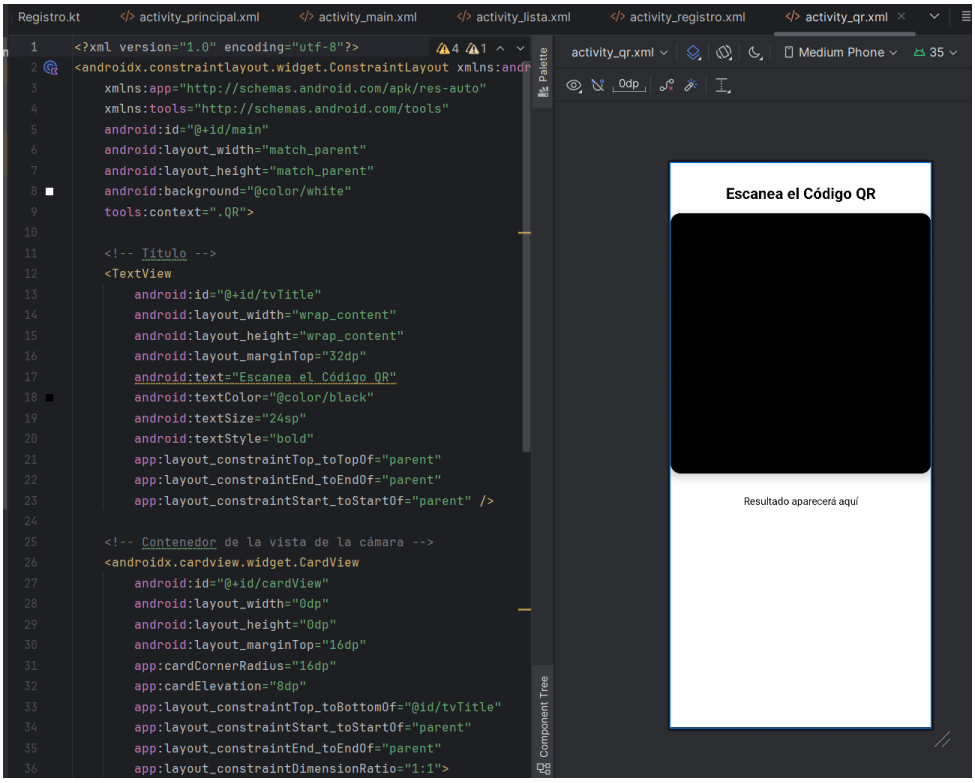


Se realiza la interfaz de la página principal, para acceder a esta se tiene que iniciar sesión correctamente, de otra forma no se va a poder acceder.

El funcionamiento de esta pagina principal es solo dar acceso a las demás ventanas.

```
class Principal : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_principal)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets

        }
        val btnAsistencia= findViewById<FloatingActionButton>(R.id.btnAsistencia)
        btnAsistencia.setOnClickListener{
            val open= Intent( packageContext: this, QR::class.java)
            startActivity(open)
        }

        val btnVisualizarLista =findViewById<FloatingActionButton>(R.id.btnVisualizarLista)
        btnVisualizarLista.setOnClickListener{
            val open1= Intent( packageContext: this, Lista::class.java)
            startActivity(open1)
        }

        val btnSalir= findViewById<Button>(R.id.btnSalir)
        btnSalir.setOnClickListener{
            val op = Intent( packageContext: this, MainActivity::class.java)
            startActivity(op)
        }
    }
}
```

> Lectura de QR



La interfaz de lectura del qr es sencilla pero permite visualizar de forma correcta el qr, así como el mensaje contenido en el qr.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding= ActivityQrBinding.inflate(layoutInflater)
    setContentView(binding.root)

    cameraExecutor= Executors.newSingleThreadExecutor()
    barcodeScanner =BarcodeScanning.getClient()

    val requestPermissionLauncher = registerForActivityResult(ActivityResultContracts.RequestPermission()){
        isGranted ->
        if(isGranted) {
            startCamera()
        }else{
            Toast.makeText( context: this,  text: "Permisos de la camára necesarios", Toast.LENGTH_LONG).show()
        }
    }
    requestPermissionLauncher.launch(Manifest.permission.CAMERA)
}
```

Esta función permite controlar los permisos de la cámara, porque para poder acceder a la cámara se ocupan varias librerías

```
implementation(libs.barcode.scanning)
implementation(libs.camera.core)
implementation(libs.androidx.camera.camera2)
implementation(libs.androidx.camera.lifecycle)
implementation(libs.androidx.camera.view)
```

Estas librerías permitan el acceso a la cámara, pero antes se tienen que pedir los permisos necesarios al usuario para poder tener este acceso.

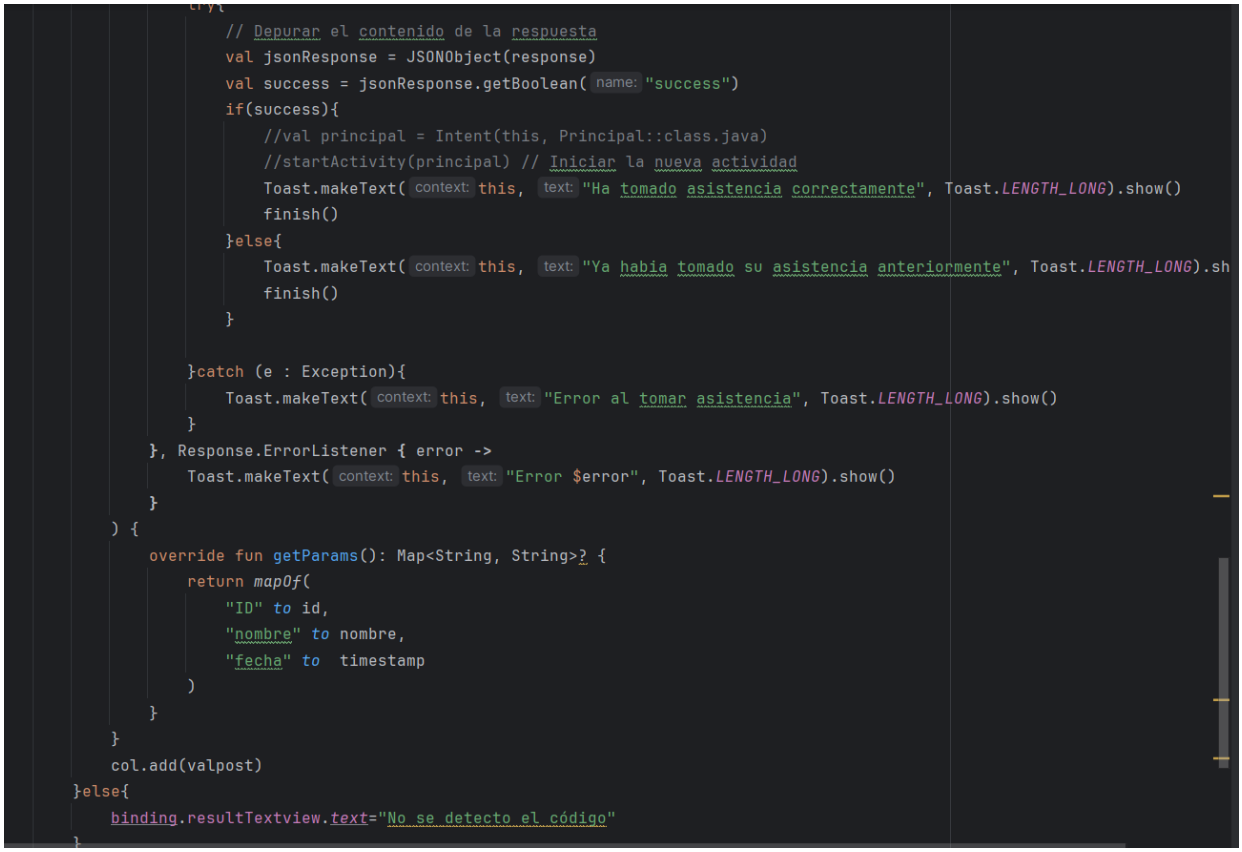
```
@OptIn(ExperimentalGetImage::class)
private fun processImageProxy(imageProxy: ImageProxy){
    val mediaImage = imageProxy.image

    if (mediaImage != null) {
        val image = InputImage.fromMediaImage(mediaImage, imageProxy.imageInfo.rotationDegrees)

        barcodeScanner.process(image)
            .addOnSuccessListener { barcodes ->
                for (barcode in barcodes) {
                    handleBarcode(barcode)
                }
            }
            .addOnFailureListener {
                //binding.resultTextview.text="Fallo al escanear el código"
                Toast.makeText( context: this,  text: "Fallo al escanear el código", Toast.LENGTH_LONG).show()
            }
            .addOnCompleteListener {
                imageProxy.close()
            }
    }
}
```

Esta función indica si se pudo escanear algo o no, si se pudo te lleva a otra función llamada handleBarcode la cual va a manejar la cuestión de los datos leídos.

```
private fun handleBarcode(barcode:Barcode){
    val txt = barcode.url ?.url ?: barcode.displayValue
    val timestamp = SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss", Locale.getDefault()).format(Date())

    val dateOnly = SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.getDefault()).format(Date())
    if(txt != null){

        val parts = txt.split( ...delimiters: " ", limit = 2)
        val id = parts[0]  // Extraer ID
        val nombre = parts.getOrElse( index: 1) { "Nombre no disponible" }

        binding.resultTextview.text = "ID: $id\nNombre: $nombre"

        if (dateOnly == lastDate) {
            return
        }
        lastDate = dateOnly


        //val url= "http://192.168.100.38/Proyecto_moviles/asistencia.php"
        val url= "http://172.21.232.56/Proyecto_moviles/asistencia.php"
        val col: RequestQueue = Volley.newRequestQueue( context: this)

        val valpost  = object : StringRequest(com.android.volley.Request.Method.POST, url,
            Response.Listener<String> { response ->
                try{
                    // Depurar el contenido de la respuesta
                    val jsonResponse = JSONObject(response)
                    val success = jsonResponse.getBoolean( name: "success")
                    if(success){
                        //val principal = Intent(this, Principal::class.java)
                        //startActivity(principal) // Iniciar la nueva actividad
                        Toast.makeText( context: this,  text: "Ha tomado asistencia correctamente", Toast.LENGTH_LONG).sho
                        finish()
```

En esta función lo primero que se hace es sacar la fecha y hora del momento en que se escanea el QR, el código a escanear solo va a contener el ID y el nombre del usuario, pero para poder determinar la asistencia se genera al momento. Para evitar que se estén guardando las asistencias por diferencia de segundos, se realiza una comparación de fechas con dateOnly y LastDate, generando solo una fecha y hora, las cuales se van a enviar a la base de datos. Para obtener los datos restantes se tiene establecido que el ID es solo de 9 caracteres o números, por lo que se separan los datos leídos del QR y dejando que los datos restantes sean el nombre del usuario.

La pagina que va a tener la conexión con la base de datos es asistencia.php



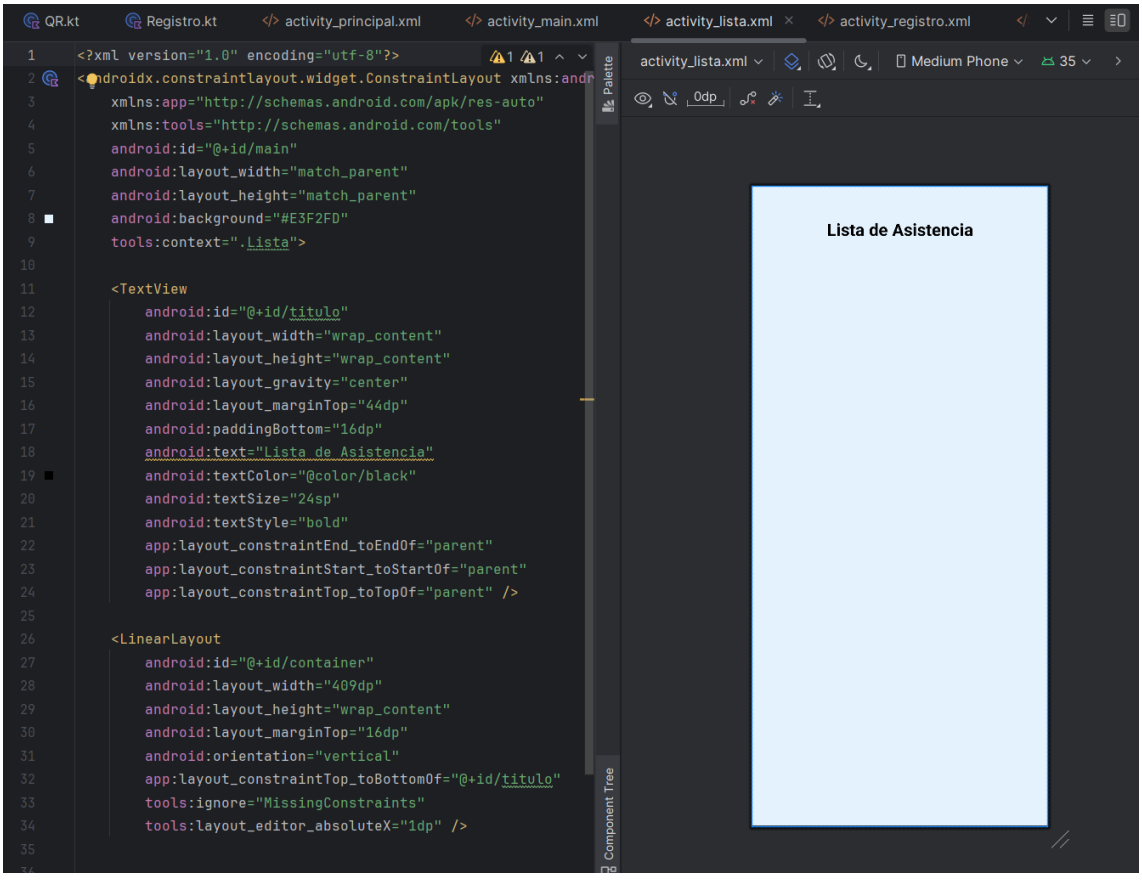
```
            try{
                // Depurar el contenido de la respuesta
                val jsonResponse = JSONObject(response)
                val success = jsonResponse.getBoolean( name: "success")
                if(success){
                    //val principal = Intent(this, Principal::class.java)
                    //startActivity(principal) // Iniciar la nueva actividad
                    Toast.makeText( context: this,  text: "Ha tomado asistencia correctamente", Toast.LENGTH_LONG).show()
                    finish()
                }else{
                    Toast.makeText( context: this,  text: "Ya habia tomado su asistencia anteriormente", Toast.LENGTH_LONG).sh
                    finish()
                }

            }catch (e : Exception){
                Toast.makeText( context: this,  text: "Error al tomar asistencia", Toast.LENGTH_LONG).show()
            }
        }, Response.ErrorListener { error ->
            Toast.makeText( context: this,  text: "Error $error", Toast.LENGTH_LONG).show()
        }
    ) {
        override fun getParams(): Map<String, String>? {
            return mapOf(
                "ID" to id,
                "nombre" to nombre,
                "fecha" to  timestamp
            )
        }
    }
    col.add(valpost)
}else{
    binding.resultTextview.text="No se detecto el código"
}
```

Se van a manejar nuevamente las respuestas con Json, indicando si es la primera vez que tomó asistencia muestra un mensaje de que se guardó correctamente, pero en caso de que ya estuviera su asistencia mandara un mensaje de que ya había tomado asistencia.

Para lo de la base de datos, también tiene una limitación de fechas, es decir que verifica el día, hace una búsqueda en la tabla de asistencias y verifica si ha tomado asistencia o no, en caso de que no, lo almacena en la base de datos, y en caso de que si manda la respuesta Json.

```php
<?php
if ($_SERVER["REQUEST_METHOD"]=="POST")
    require_once 'conexion.php';

    $ID =$_POST["ID"];
    $nombre=$_POST["nombre"];
    $date=$_POST["fecha"];

    $dateOnly = date("Y-m-d", strtotime($date));

    $ass = $conexion->prepare("SELECT * FROM asistenciaapp WHERE ID = ? and nombreCompleto =? and DATE(Fecha)=?");
    $ass->bind_param("sss", $ID, $nombre, $dateOnly);
    $ass->execute();
    $result = $ass->get_result();

    if ($result->num_rows > 0) {
        echo json_encode(["success" => false]);
    } else {
        echo json_encode(["success" => true]);

        $query= "INSERT INTO asistenciaapp
                (ID, nombreCompleto, Fecha)
            VALUES
                ('".$ID."', '".$nombre."', '".$date."')";

        if ($conexion->query($query) === TRUE) {
            //echo "Usuario insertado exitosamente";
        } else {
            echo "Error: " . $query . "<br>" . $conexion->error;
        }

    }

    $ass->close();
    $conexion->close();

?>
```

> Mostrar lista



Finalmente se hizo la interfaz de mostrar la lista de asistencia, la cual se va a ir llenando progresivamente.

```
fun visualizarAsistencia(){
    //val url= "http://192.168.100.38/Proyecto_moviles/lista.php"
    val url= "http://172.21.232.56/Proyecto_moviles/lista.php"
    val cola: RequestQueue = Volley.newRequestQueue( context: this)
    val request = JsonObjectRequest(
        Request.Method.GET, url,  jsonRequest: null,
        { response ->
            val success = response.getBoolean( name: "success")
            if (success) {
                val dataArray = response.getJSONArray( name: "lista")
                val contenedor= findViewById<LinearLayout>(R.id.container)

                for (i in 0 ≤ until  < dataArray.length()) {
                    val item = dataArray.getJSONObject(i)
                    val id = item.getString( name: "ID")
                    val nombre = item.getString( name: "nombreCompleto")
                    val fecha = item.getString( name: "Fecha")

                    // Crear un TextView dinámicamente para cada registro
                    val texto = TextView( context: this).apply {
                        text = "ID: $id\nNombre: $nombre\nFecha: $fecha"
                        textSize = 18f
                        setPadding( left: 5,  top: 16,  right: 0,  bottom: 16)

                    }
                    contenedor.addView(texto)
                }
            } else {
                val message = response.getString( name: "message")
                Toast.makeText( context: this, message, Toast.LENGTH_LONG).show()
            }
        },
        { error ->
            Log.e( tag: "Error", error.toString())
        }
```

Finalmente, para visualizar la asistencia se accede a la pagina lista.php la cual va a realizar la búsqueda de los registros que hay en la lista de asistencias, estos se van a almacenar en una lista y esta lista se va ir mandando a través de Json, este se va a ir imprimiendo a través de un for y en un textview para que se pueda visualizar.

lista.php > ...

```php
<?php
    require_once 'conexion.php';
    $lista=[];
    $stmt = $conexion->prepare("SELECT ID, nombreCompleto, Fecha FROM asistenciaapp");
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        while ($A= $result->fetch_assoc()){
            $lista [] =[
                'ID' => $A["ID"],
                'nombreCompleto' => $A["nombreCompleto"],
                'Fecha' => $A["Fecha"]
            ];
        }
        echo json_encode(["success" => true, "lista" => $lista]);
    } else {
        echo json_encode(["success" => false]);
    }

    $stmt->close();
    $conexion->close();
?>
```

Conclusión

Se puede concluir que fue algo complicado realizar la cuestión de envío-respuesta que hubo entre php y kotlin, se busco la forma de comunicación entre ambos y al final pudo funcionar.

Algo que también se puede decir que fue complicado, fue adecuar bien las interfaces y que estas fueran entendibles.