REPORTE – PROYECTO SEGUNDO PARCIAL-PAR 5

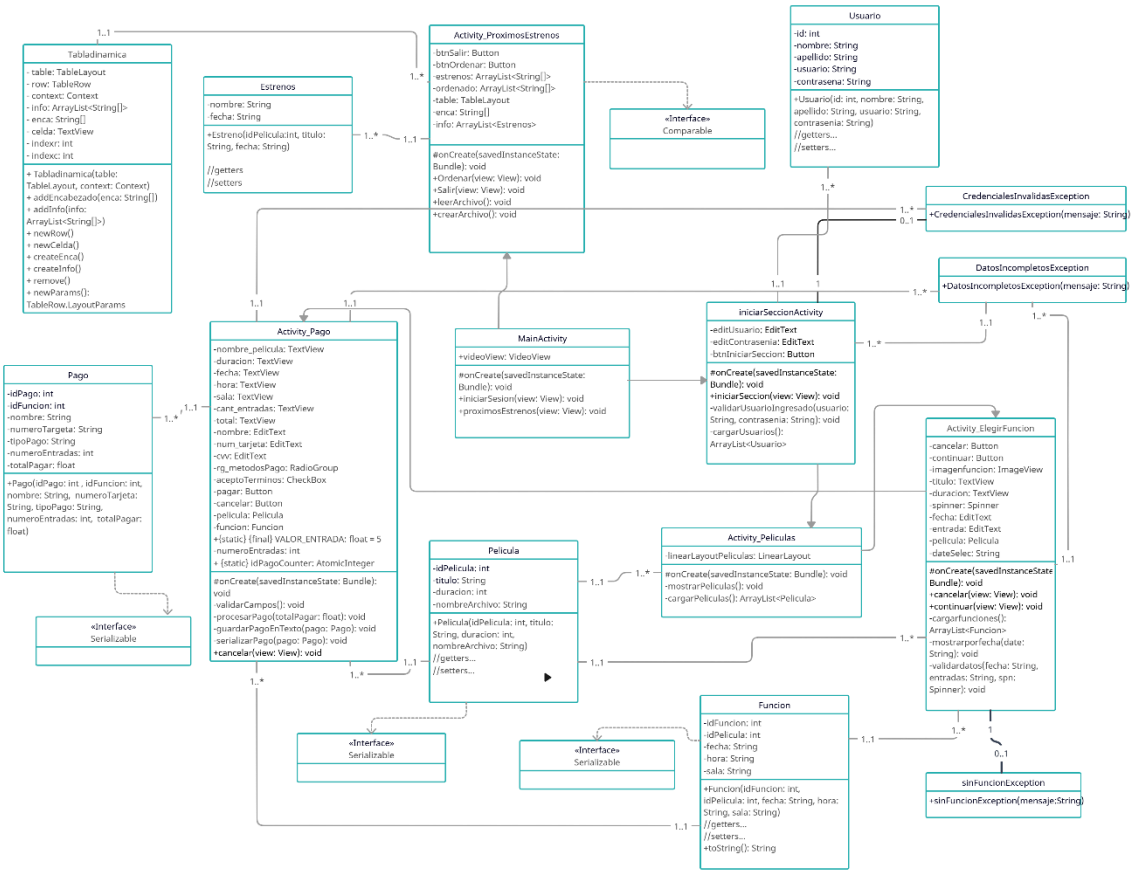
PROGRAMACIÓN ORIENTADA A OBJETOS
TÉRMINO II 2024-2025

Integrantes:
Camuendo Diaz Jesua Misael
Cuenca Sáez Jacqueline Anahí
Quimi De La O Valeria Doris

URL Repositorio:

Fecha: 26/01/2025

1. Diagrama de clases



2. Tareas

En esta sección incluirán el detalle de las tareas que se asignó a cada estudiante.

Estudiante (Camuendo Diaz Jesua Misael):

1. Tarea 1: Activity_ProximosEstrenos
2. Tarea 2: Activity_ElegirFuncion
3. Tarea 3: activity_elegir_funcion.xml

Estudiante (Cuenca Saez Jacqueline Anahi):

1. Tarea 1: MainActivity
2. Tarea 2: Activity_Pago
3. Tarea 3: activity_pago.xml

Estudiante (Quimi De La O Valeria Doris):

1. Tarea 1: iniciarSeccionActivity
2. Tarea 2: Activity_Peliculas
3. Tarea 3: activity_iniciar_seccion.xml

3. Identificación de teoría aplicada en programación orientada a objetos.

Manipulación de objetos

```
            String[] partes= linea.split( regex: ",");
            int id= Integer.parseInt(partes[0]);
            String nombre= partes[1];
            String apellido= partes[2];
            String usuario= partes[3];
            String contrasenia= partes[4];

            //Crear objetos usuario y agregarlos a la lista
            usuarios.add(new Usuario(id, nombre,apellido,usuario,contrasenia));
        }
```

Explicación: En este caso, se utiliza la clase Usuario para crear objetos que representan a cada usuario. Estos objetos almacenan atributos como id, nombre, apellido, usuario y contraseña. El código muestra cómo se crean y manipulan objetos dinámicamente, almacenándolos en una lista de usuarios.

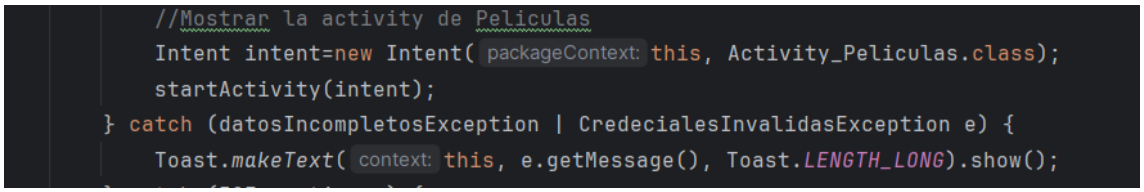
Manejo de Excepciones

```
1 usage
public void leerArchivo(){
    try{
        BufferedReader bf = new BufferedReader(new InputStreamReader(getResources().openRawResource(R.raw.estrenos)));
        String linea =null;
        while((linea= bf.readLine())!= null){
            String[] partes= linea.trim().split( regex: ",");
            String[] real = {partes[1],partes[2]};
            Estrenos e = new Estrenos(partes[1],partes[2]);
            estrenos.add(real);
            info.add(e);
        }
    } catch (FileNotFoundException e) {
        System.out.println("Archivo no Encontrado");
    } catch (IOException e) {
        System.out.println("Ha ocurrido una excepción");
    }

}
```

Explicación: En esta parte del código se encuentra el bloque try-catch, usado en el manejo de excepciones, try se ejecuta, en el momento que llega a ocurrar una excepción el flujo se transfiere al bloque catch. FileNotFoundException se utiliza para capturar errores que ocurren cuando un archivo no es encontrado, esta hereda de IOException, la cual puede ser lanzada por una variedad de problemas relacionados con la entrada y salida de datos, como ejemplo lectura de un archivo.

Manejo de Excepciones propias



```
            //Mostrar la activity de Peliculas
            Intent intent=new Intent( packageContext: this, Activity_Peliculas.class);
            startActivity(intent);
        } catch (datosIncompletosException | CredecialesInvalidasException e) {
            Toast.makeText( context: this, e.getMessage(), Toast.LENGTH_LONG).show();
```

Explicación: en esta parte del código se maneja excepciones propias, se hace uso de las clases datosIncompletosException y CredencialesInvalidasException, excepciones personalizadas para el manejo de validación de datos.

Lectura de Archivos

```
1 usage
private ArrayList<Funcion> cargarfunciones() {
    ArrayList<Funcion> funciones= new ArrayList<>();
    try(BufferedReader bf= new BufferedReader(new InputStreamReader(getResources().openRawResource(R.raw.funciones)))){
        String linea;
        while((linea=bf.readLine())!= null){
            String[] partes= linea.split( regex: ",");
            int idFuncion= parseInt(partes[0]);
            int idPelicula= parseInt(partes[1]);
            String fechas= partes[2];
            String hora= partes[3];
            String sala = partes[4];
            //crear objetos funciones
            funciones.add(new Funcion(idFuncion, idPelicula, fechas, hora,sala));
        }
```

Explicación: en esta parte del código, se usa un método con el fin de leer un archivo que contiene información sobre funciones, procesar sus datos y crear objetos de tipo Función, esto se empleó para crear objetos de Película, Pago, Usuario. El uso de BufferedReader nos permite una lectura más eficiente del archivo.

Escritura de Archivos

```
1 usage
private void guardarPagoEnTexto(Pago pago) throws IOException {
    String datosPago = pago.getIdPago() + "," +
            pago.getIdFuncion() + "," +
            pago.getNombre() + "," +
            pago.getNumeroTarjeta() + "," +
            pago.getTipoPago() + "," +
            pago.getNumeroEntradas() + "," +
            pago.getTotalPagar() + "\n";

    try (FileWriter w = new FileWriter(new File(getFilesDir(), child: "pagos.txt"), append: true)) {
        w.write(datosPago);
    }
}
```

Explicación: en esta parte del código se hace uso de la escritura de archivos, con el objetivo de guardar información en un archivo sobre un pago.

Serialización

```
1 usage
private void serializarPago(Pago pago) throws IOException {
    try (FileOutputStream fos = new FileOutputStream(new File(getFilesDir(), child: "funcion" + pago.getIdFuncion() + ".bin"));
         ObjectOutputStream oos = new ObjectOutputStream(fos)) {
        oos.writeObject(pago);
    }
}
```

Explicación: La serialización permite convertir un objeto en un flujo de bytes para almacenarlo o transmitirlo. En el código, se implementa la serialización del objeto Pago a un archivo binario. Esto es útil para la persistencia de datos, asegurando modularidad y portabilidad.

Interfaz Comparable

```
public class Activity_ProximosEstrenos extends AppCompatActivity implements Comparable<String>{
```

```
1 usage
public void Ordenar(View view){
    info.sort((Estrenos e1, Estrenos e2) -> e1.getNombre().compareTo(e2.getNombre()));
    for(Estrenos i: info){
        String[] or = {i.getNombre(),i.getFecha()};
        ordenado.add(or);
    }
    Tabladinamica tabladinamica = new Tabladinamica(table,getApplicationContext());
    tabladinamica.remove();
    tabladinamica.addEncabezado(enca);
    tabladinamica.addInfo(ordenado);
}
```

Explicación: Aquí hacemos uso del método compareTo(), el cual está sobrescrito, ya que pertenece a la Interfaz Comparable. Gracias a este método podemos ordenar de manera alfabética debido a que compara caracter por caracter las cadenas de texto (nombres). Esto nos permite reducir código y lograr de una forma más fácil lograr actualizar la vista en orden alfabético.

Controladores de eventos

```
//configurar al hacer click en la imagen
imagenBoton.setOnClickListener(v ->{
    Intent intent = new Intent( packageContext: Activity_Peliculas.this, Activity_ElegirFuncion.class);
    intent.putExtra( name: "pelicula",movie);
    intent.putExtra( name: "imagen",parametro);//enviar el objeto pelicula seleccionada
    startActivity(intent); //iniciar actividadd
});
```

```
fecha.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Calendar calendar = Calendar.getInstance();
        int ano = calendar.get(Calendar.YEAR);
        int mes = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        DatePickerDialog dpiker = new DatePickerDialog( context: Activity_ElegirFuncion.this, new DatePickerDialog.OnDateSetListener() {
            1 usage
            @Override
            public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
                dateSelec = dayOfMonth+"/"+(month+1)+"/"+year;
                fecha.setText(dateSelec);
                mostrarporfecha( date: "0"+dateSelec);

            }
        },ano,mes,day);
```

Explicación: en este parte del código, se implementa controladores de eventos que respondan a las interacciones del usuario. El método setOnClickListener registra un listener para el evento de click en el imagenBoton. Asi asegurando que cuando el usuario haga clic en el botón, se ejecute el código definido dentro del bloque del listener.

Programación dinámica de GUI

```
//recorrer la lista de peliculas
for (Pelicula movie: cargarPeliculas()){

    //crear un layout para cada pelicula
    LinearLayout itemLayout= new LinearLayout( context: this);
    itemLayout.setOrientation(LinearLayout.VERTICAL);
    itemLayout.setLayoutParams(new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.WRAP_CONTENT));
    itemLayout.setPadding( left: 8,  top: 8,  right: 8,  bottom: 8);

    //crear un ImageButton para pelicula
    ImageButton imagenBoton= new ImageButton( context: this);
    LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,  // Ancho ajustado al contenido
            ViewGroup.LayoutParams.WRAP_CONTENT   // Alto ajustado al contenido
    );
    layoutParams.gravity = Gravity.CENTER; // Centrar el botón dentro del layout
```

```
    //Agregar el ImageButton al layout
    itemLayout.addView(imagenBoton);
    //Agregar el layout al contenedor
    linearLayoutPeliculas.addView(itemLayout);
```

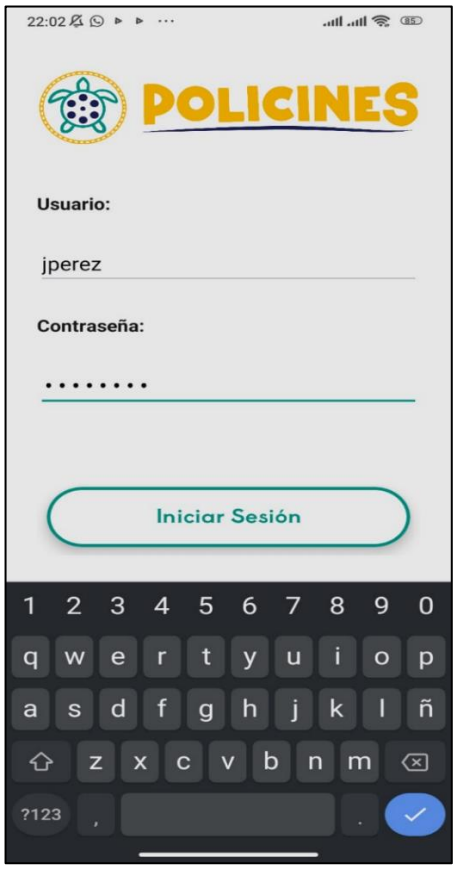
Explicación: en esta parte del código, se creó y configuro componentes de la interfaz de manera programática en función de los datos cargados, como una lista de película, en lugar de definir los componentes directamente en un archivo XML. Al recorrer la lista de películas, por cada película se generan componentes visuales como un linearLayout que actué como contenedor y imagenBottons.

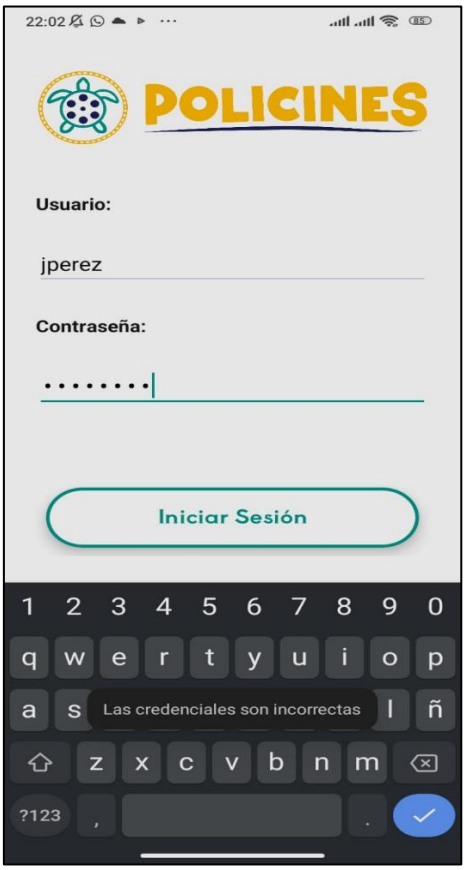
4. Programa en ejecución

En esta sección deberán incluir screenshots de su proyecto en ejecución, con el objetivo de que se visualice la ejecución de este.


POLICINES
INICIAR SESION
PROXIMOS ESTRENOS


POLICINES
Usuario:
Contraseña:
Iniciar Sesión


POLICINES
Usuario:
jperez
Contraseña:
Iniciar Sesión


POLICINES
Usuario:
jperez
Contraseña:
Iniciar Sesión
Las credenciales son incorrectas

POLICINES

Peliculas

Eliga una pelicula

POLICINES

Elija la función

Inside Out 2

Duración: 110 min

Fecha:

Horarios:

Entradas:

Continuar

Cancelar

POLICINES

Detalles de Función

Película: Inside Out 2
Duración: 110 min
Fecha: 04/12/2024
Función: 15:00 **- Sala** 2
Entradas: 2
Total a pagar: $10.0

Detalles de pago

Nombre

Número de tarjeta

CVV

Mastercard Visa

Acepto los términos y condiciones.

Pagar

Cancelar

POLICINES

Elija la función

Inside Out 2

Duración: 110 min

Fecha:

4/12/2024

Horarios:

15:00 - Sala 2

Entradas:

2

Continuar

Cancelar

5. JAVADOC

Agregar la documentación JAVADOC como una carpeta adicional a este reporte. Los métodos deben estar siempre comentados con el formato explicado en clase.