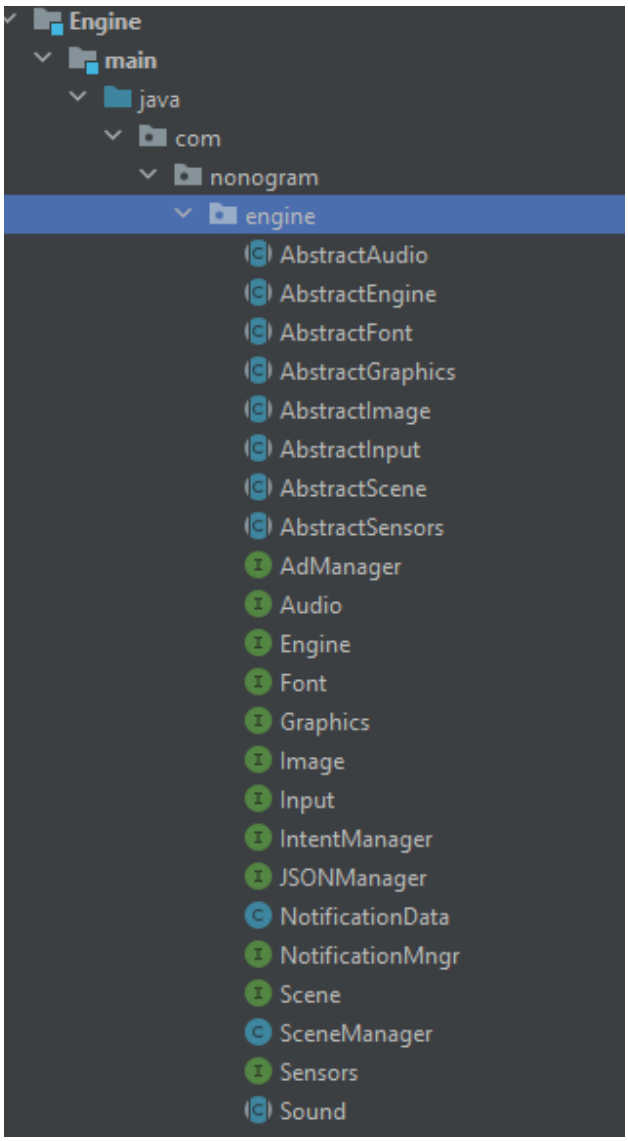
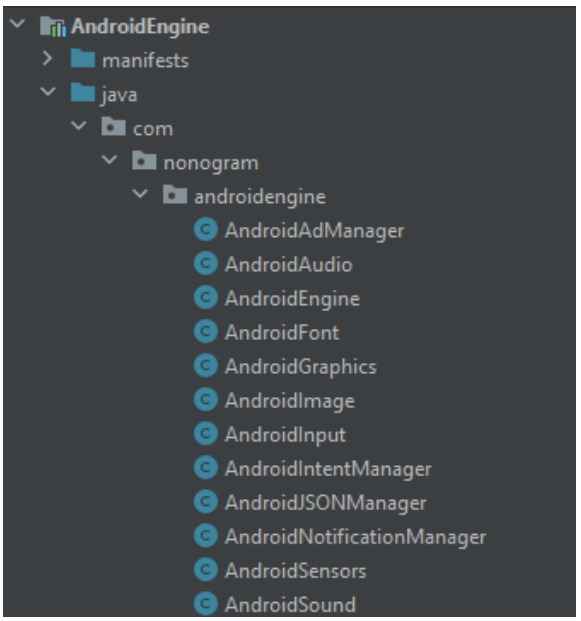
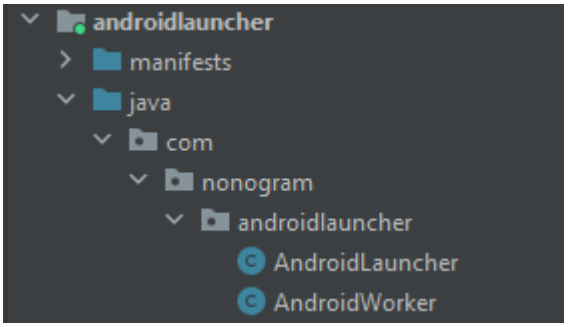
# VDM – Práctica 2 – Grupo 04

## 1. Arquitectura de clases



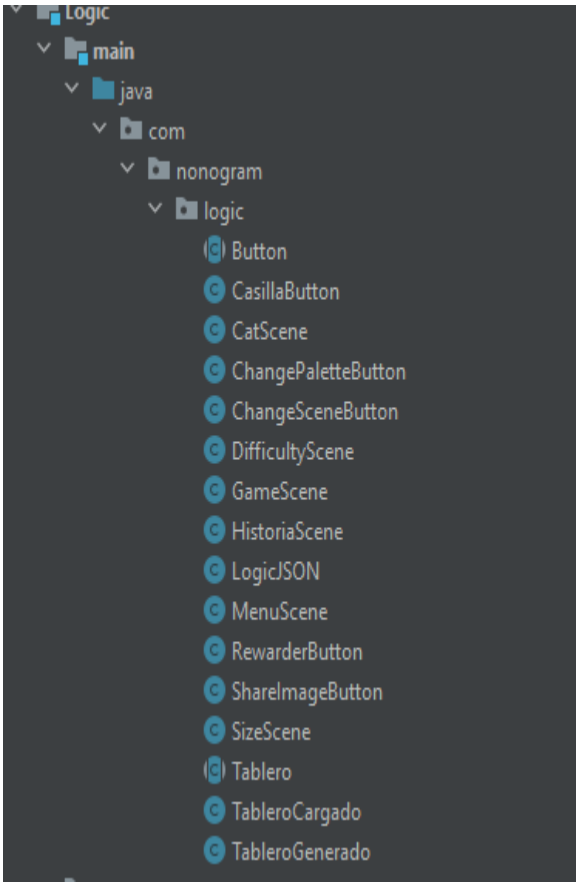

Para esta práctica hemos eliminado los módulos de PC. La arquitectura de clases es prácticamente la misma, sumándole las nuevas clases necesarias para implementar nuevas funcionalidades: NotificationMngr, IntentManager, AdManager. Estos, tienen sus implementaciones para la plataforma de Android en el módulo AndroidEngine.
Además, ahora tenemos más funcionalidades en AndroidLauncher, ya que tenemos la clase AndroidWorker que se utiliza junto a las notificaciones.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //Accedemos a las dos vistas
    setContentView(R.layout.androidlauncher);
    c = (ConstraintLayout) findViewById(R.id.parent_linear_layout);
    gameView = findViewById(R.id.surfaceView);
    _mAdView = findViewById(R.id.adView);
    //creamos el motor
    _myEngine = new AndroidEngine( context: this, gameView);
    //manejo de errores: si se crea mal algo, para antes de empezar.
    if(!_myEngine.init()){
        _myEngine.close();
    }
    //cargamos los anuncios
    ((AndroidAdManager)_myEngine.getAdManager()).setAdView(_mAdView, c);
    _myEngine.getAdManager().loadAds();
    //creamos e inicializamos la escena principal
    _myEngine.getSceneManager().setGameSize( w: 450, h: 800);
    //inicializamos la primera escena
    MenuScene sceneinicial = new MenuScene();
    if(!_myEngine.getSceneManager().push(sceneinicial)){
        _myEngine.close();
    }
    //detecta si ha entrado mediante una notificacion
    Intent intent = getIntent();
    if(intent.getExtras()!=null && intent.getExtras().containsKey("notification")){
        sceneinicial.handleOpeningNotifications();
    }
    //gestión de la barra de notificaciones
    getSupportActionBar().hide();
    View decorView = getWindow().getDecorView();
    // Hide the status bar.
    int uiOptions = View.SYSTEM_UI_FLAG_FULLSCREEN;
    decorView.setSystemUiVisibility(uiOptions);

}
```

En el método OnCreate de la actividad en AndroidLauncher, ahora accedemos a dos vistas en lugar de solo a una, la vista del anuncio y la vista de juego. Además, inicializamos también los anuncios y comprobamos si se ha entrado mediante una notificación.
En la propia clase tenemos los diferentes listeners para el que el flujo de android funcione correctamente y se comunique con nuestro Engine (gestor de la vista de juego), además de la gestión de notificaciones de cierre que utilizará un Worker.

# VDM – Práctica 2 – Grupo 04




En cuanto a la lógica, debido a los requirimientos de esta práctica, ahora tenemos dos tipos de tableros: generados (aleatorios) y cargados (mediante archivos .json). Por lo tanto, tenemos una división de escenas que funcionan de la siguiente manera:


La rama de los niveles cargados (izquierda) se divide en categorías según los tamaños. Hemos recogido niveles de cuatro categorías: 5x5,10x10,15x15,20x20. Cada categoría tiene 20 niveles, y cuando terminas todos los niveles de una categoría, se desbloquea la siguiente.

En cuanto a la retención de jugadores, hemos realizado el sistema de paletas con 10 paletas (la primera de todas es cambiante según la luz que le entre al teléfono mediante el sensor de luminancia). Se desbloquea una nueva paleta cada 5 niveles (en caso de querer probar a desbloquear paletas, se puede modificar el valor de int UNLOCK_PALETTE_EVERY que se encuentra en MenuScene.java).

Además, existe un sistema de vidas, comenzando en la primera partida con 5 vidas y pudiendo ganar vidas cada vez que se visualiza un anuncio de tipo Rewarded. Estos anuncios aparecerán cuando se llegue a las 0 vidas y de manera constante en la escena de Menu. Para perder una vida, hay que pulsar una casilla errónea.

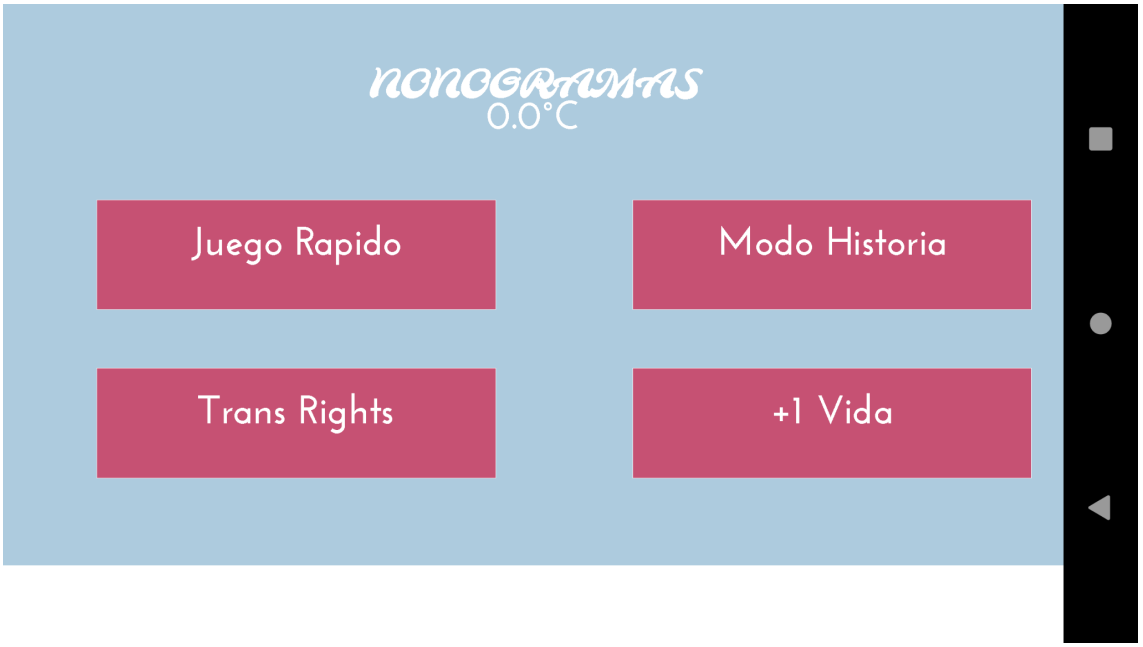
El sistema de notificaciones envía una notificación cuando se cierra el juego. Una vez se cierra, pasados 2 segundos (valor modificable en el método handleClosingNotifications() que se encuentra en MenuScene) aparecerá una notificación indicando que se puede volver a entrar en la aplicación. Si el usuario entra mediante este método, se le recompensará con una vida.

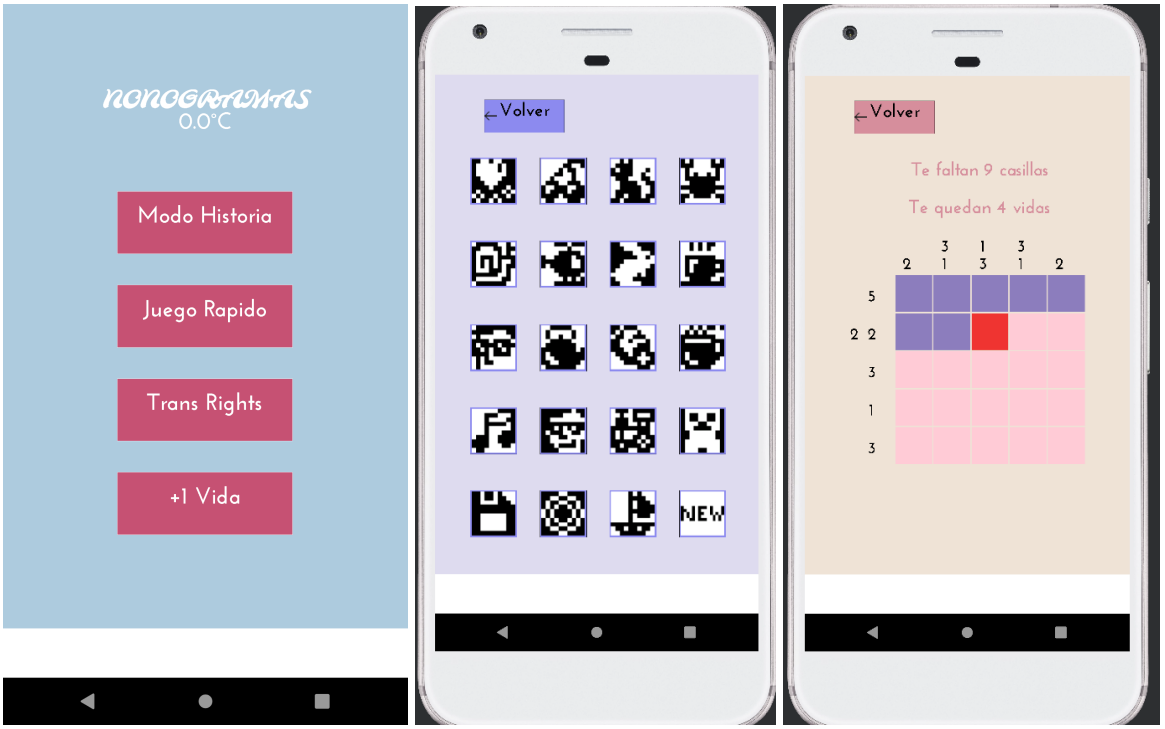
# VDM – Práctica 2 – Grupo 04

La lectura de archivos se realiza leyendo de manera abstracta cualquier archivo deseado como un String, e interpretando dicho String en el módulo de la Lógica, en el script LogicJSON. Nuestros json principales son:

- preferences.json: guarda el progreso y estado actual del ultimo tabledo cargado jugado, además de la paleta actual. Este archivo se almacena en la memoria interna del teléfono.
- checksum.json: se utiliza para comprobar que el archivo preferences.json guardado anteriormente no se modifique. Cada vez que se guarda un preferences.json, se guarda un valor asociado en checksum.json. Si se modifica el valor de preferences.json desde fuera de la aplicación, cuando lea dicho archivo comprobará si es igual al anterior mediante checksum.json. En caso negativo, borrará el archivo anterior de guardado y creará uno desde cero.
- Archivos .json para los niveles: cada nivel tendrá un archivo .json, divididos en carpetas según sus categorías (ejemplo: assets/JSON/Boards/5x5/0.json). En él, se guarda en un bool[][] el estado de victoria del tablero.

## 2. Capturas de pantalla:




## 3. Notas

Sobre los sensores: existe un medidor de temperatura, que la mayoría de móviles no tienen implementado, pero se puede observar como si cambiamos en el menú del emulador de Android Studio el valor de dicho sensor, cambiará el valor que está en MenuScene. Pasa

parecido con el sensor de luminancia, que sí que se puede utilizar bien en cualquier dispositivo, pero desconocemos de cuáles son los valores medidos en lux que identifican un ambiente oscuro y uno claro. Para su fácil comprobación, desde el menú de sensores se puede cambiar a >150 o <150 el valor para comprobar que la paleta 1 (Light/Dark) cambia según la luminancia.

Sobre los anuncios, como hablamos en la tutoría que tuvimos, en el emulador tarda bastante en cargar el banner y los rewarded no llegan a cargar. En cambio, hemos probado en diferentes dispositivos con API 30,31 y 32 y ambos cargaban perfectamente.