Práctica 3. Trivial

Convocatoria extraordinaria

OBJETIVO DE LA PRÁCTICA.

Esta práctica tiene como objetivo el desarrollar una aplicación en Android, principalmente la interfaz gráfica, que consista en un juego de preguntas y respuestas.

ENUNCIADO.

El alumno creará una aplicación en Android que presente preguntas y tres respuestas donde el usuario deberá seleccionar la correcta, basándose en el tradicional juego *Trivial Pursuit*™.

Ilustración 1. Trivial Pursuit Edición Genus, España, 1995

El juego en el que nos vamos a basar es: Horn Abbot International Limited, *Trivial Pursuit Ed. Genus*, Spain, 1995.

El funcionamiento que deberá tener la aplicación será el siguiente:

- Se le plantearán al usuario una serie de preguntas que tendrán tres respuestas de la que solo una será correcta.
- El usuario deberá contestar bien una respuesta de cada categoría para ganar el juego.
- Las preguntas se clasifican en seis categorías: geografía, espectáculos, historia, arte y literatura, ciencias y naturaleza, y deportes y ocio.

Se provee al alumno de un paquete (`es.esne.eop.trivial.questions`) para ayudarle a la generación de preguntas y respuestas. Este paquete consta de dos clases explicadas en el anexo de este documento:

- `QuestionModel`: modelo de cada pregunta. Cada objeto de esta clase será una pregunta. En esta clase, también se define una enumeración con las categorías de cada pregunta.
- `QuestionDB`: actúa como base de datos de preguntas. Dispone de un array de objetos `QuestionModel` con visibilidad privada. Este objeto tiene tres preguntas de cada categoría, aunque el alumno puede añadir más si lo desea. Además, dispone de dos métodos de extracción aleatoria de preguntas: uno para preguntas de cualquier categoría y otro para preguntas de una categoría determinada como parámetro.

Obligaciones de implementación.

- El alumno deberá llevar un contador de respuestas correctas de cada categoría para comprobar cuándo el usuario ha finalizado el juego.
- Una vez el juego finalice, se le podrá sugerir comenzar una partida nueva.

Queda a decisión del alumno para mejorar la nota el añadir temporizadores a las preguntas, un dado que presente un color aleatorio que sea el que decida la categoría de la pregunta que se le va a hacer al usuario o el uso de un tablero, entre otros.

Ejemplo de interfaz.

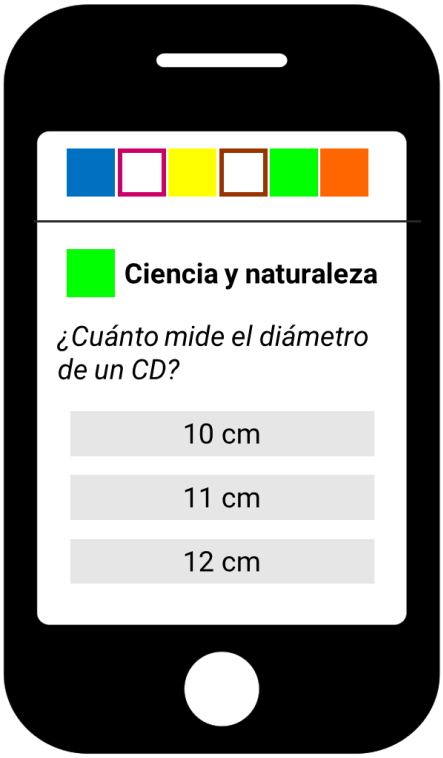

Ilustración 2. Ejemplo de interfaz

PERIODO DE REALIZACIÓN.

- Entrega máxima: **30/06/2019 a las 23:59 h**.

MATERIAL A ENTREGAR.

- Memoria de la práctica. Debe incluir el proceso de diseño de la aplicación, tanto a nivel gráfico como de lógica de funcionamiento.
- Código de la aplicación comentado y con mensajes de depuración. Deberán subirse al campus:
  i. Carpeta app.
  ii. Carpeta gradle.
  iii. Archivos build.gradle, gradlew, gradlew.bat, settings.gradle.

EVALUACIÓN.

Se evaluará tanto la calidad del código como de la memoria, así como la presentación de ambos y el funcionamiento de la aplicación.

Será obligatoria la entrega a través del campus virtual en el control habilitado para ello en el plazo determinado. En caso contrario, la práctica no se evaluará.

Las entregas que no cumplan con los requisitos obligatorios no serán corregidas.

ANEXO. CLASES YA DESARROLLADAS.

Las clases ya desarrolladas se presentan junto a una aplicación Android para mostrar su utilización. Esta aplicación únicamente genera preguntas y las imprime en un *ScrollView* con formato HTML.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</ScrollView>
```

MainActivity.java

Solo se muestra el código añadido en *onCreate*.

```
textView.append(Html.fromHtml(QuestionDB.getRandomQuestion().toHtmlRea
dableString()));

textView.append(Html.fromHtml(QuestionDB.getRandomQuestion(QuestionMod
el.Type.BLUE).toHtmlReadableString()));

textView.append(Html.fromHtml(QuestionDB.getRandomQuestion(QuestionMod
el.Type.PINK).toHtmlReadableString()));

textView.append(Html.fromHtml(QuestionDB.getRandomQuestion(QuestionMod
el.Type.YELLOW).toHtmlReadableString()));

textView.append(Html.fromHtml(QuestionDB.getRandomQuestion(QuestionMod
el.Type.BROWN).toHtmlReadableString()));

textView.append(Html.fromHtml(QuestionDB.getRandomQuestion(QuestionMod
el.Type.GREEN).toHtmlReadableString()));

textView.append(Html.fromHtml(QuestionDB.getRandomQuestion(QuestionMod
el.Type.ORANGE).toHtmlReadableString()));
```

QuestionModel.java

Solo se muestra la interfaz que utilizará el alumno.

```
package es.esne.eop.trivial.questions;

/**
 * Question model
 * @author Carolina on Dec. 2018
 */
public class QuestionModel {

    /**
     * Question type definition
     */
    public enum Type{
        BLUE,   //Geography
        PINK,   //Entertainment
        YELLOW, //History
        BROWN,  //Arts & literature
        GREEN,  //Sciences & nature
        ORANGE  //Sports & leisure
    }

    /**
     * Gets question type
     * @return Question type
     */
    public Type getType() { return type; }

    /**
     * Gets question statement
     * @return Question statement
     */
    public String getStatement() { return statement; }

    /**
     * Gets question first answer
     * @return First answer
     */
    public String getAnswer1() { return answers[0]; }

    /**
     * Gets question second answer
     * @return Second answer
     */
    public String getAnswer2() { return answers[1]; }

    /**
     * Gets question third answer
     * @return Third answer
     */
    public String getAnswer3() { return answers[2]; }

    /**
     * Gets number of correct answer
     * @return Number of correct answer
     */
    public int getCorrect() { return correct; }
```

```
    /**
     * Converts current object into a readable String
     * @return Readable object
     */
    public String toReadableString(){
        …
    }

    /**
     * Converts current object into a readable HTML String
     * @return Readable HTML object
     */
    public String toHtmlReadableString(){
        …
    }
}
```

QuestionDB.java

Dispone del *array* con las preguntas y respuestas. Solo se presenta la interfaz que usará el alumno.

```
package es.esne.eop.trivial.questions;

import java.util.Random;

/**
 * Questions data
 * Bibliography: question types, question statements and correct
 * answer are extracted from:
 *      Horn Abbot International Limited, Trivial Pursuit Ed. Genus,
 *      Spain, 1995.
 * @author Carolina on Dec. 2018
 */
public class QuestionDB {

    /**
     * Collection of questions
     */
    private static final QuestionModel[] QUESTIONS={…};

    /**
     * Gets a random question
     * @return Random question model
     */
    public static QuestionModel getRandomQuestion(){ … }

    /**
     * Gets a random questions from a question type
     * @param type Question type
     * @return Random question model object from question type
     */
    public static QuestionModel getRandomQuestion(QuestionModel.Type
    type){…}
}
```