UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Título del Proyecto:

Canciones Inglés

Autor:

Arrufat Sánchez, Ana María

Director:

Tomás Gironés, Jesús

TESINA PARA LA OBTENCIÓN DEL TÍTULO DE:

Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles

Mes del 2016

Contenido

Introducción 4
Descripción del problema 4
Objetivos 5
Motivación 5
Arquitectura de la aplicación 6
Esquema del diseño 6
Diagrama de casos de uso 6
Diagrama de clases 7
Modelo de datos 18
Estructura de la base de datos de Firebase 18
Estructura de un usuario: 18
Estructura de una canción: 18
Vistas 19
TabbedActivity 19
Navigation Drawer e inicio de sesión 20
Vista canción 20
Rellenar palabras 21
Conclusiones 21
Bibliografía 22
Anexos 22
Código fuente en GitHub 22

Introducción

Descripción del problema

El vertiginoso avance tecnológico experimentado en los últimos años ha revolucionado la vida contemporánea en todos sus aspectos. Hoy en día, en un mundo cada vez más globalizado y conectado gracias a la aparición de Internet, el inglés se ha convertido en una de las principales herramientas para la comunicación en todo tipo de actividades. Según un artículo publicado por el periódico El País [1] "se calcula que el 80% del contenido publicado en internet está en inglés".

Una manera más informal de medir el uso de los idiomas en internet, sería mirar el número de artículos publicados para cada idioma en Wikipedia:

Idioma	Número de artículos publicados
Inglés	5.483.037
Alemán	2.104.051
Ruso	1.422.657
Español	1.355.571

Como podemos observar, la diferencia entre los dos idiomas con mayor número de artículos publicados es abismal.

Podemos ver la globalidad del idioma mediante la cantidad de personas que lo utilizan. Aunque bien es cierto que se sitúa en el tercer puesto en el ránking de los idiomas más hablados en el mundo de forma nativa, por detrás del chino y el español, gana toda batalla cuando se habla de personas que hablan o dominan el inglés como segunda lengua, estimándose que cerca de un billón de personas lo aprende como segunda lengua.

Por todo ello, se hace evidente que el aprendizaje de este idioma se hace prácticamente imprescindible si uno no quiere quedarse desconectado del mundo. Y hay algo en lo que todos los expertos coinciden: cuanto antes, mejor.

Está demostrado que al aprender dos idiomas desde pequeño se interioriza el segundo idioma de forma innata y genera beneficios en el niño que repercutirán de manera muy positiva, entre los cuales se pueden destacar los siguientes

- Mayor desarrollo cognitivo
- Mejor capacidad de comunicación.
- La mente se vuelve más flexible y creativa.
- Mejoran las habilidades de resolución de problemas.
- Rapidez mental.
- Mejor concentración y atención selectiva.

Esta sería la situación ideal: que cualquier padre del siglo XXI dedicara esfuerzos a procurar que sus hijos estuvieran diariamente en contacto con el inglés. Sin embargo, ¿qué hacemos para que niños más mayores aprendan y se interesen por este idioma?

No es un secreto que la metodología empleada en los colegios para la enseñanza del inglés no ha sido hasta hace poco la mejor vía para captar el interés del alumnado.

En la mayoría de colegios de nuestro país el idioma se enseña de manera muy rígida, "cuadrada" e incluso aburrida, contextualizada y poco realista. Enseñar una serie de reglas, la lista de verbos irregulares, los tiempos verbales, etc. con la finalidad de hacer una prueba, no es una buena metodología si el objetivo es conseguir que el alumnado sea capaz de desenvolverse con soltura en una conversación real.

De hecho, en muchos casos lo único que realmente se consigue es que el alumnado, abrumado por tener que aprender tantas reglas y sin ver una aplicación real a su aprendizaje (entender en mayor o menor medida una película en Ingles, ser capaz de mantener una conversación, etc.), comience a tener una mala relación con el idioma y pierda el interés.

Objetivos

Con el objetivo de suplir estas carencias en la enseñanza del inglés y despertar el interés de los más pequeños, Mario Gómez Sacedón [2] desarrolló una aplicación que, a través de la música, ofrece una manera más interactiva, entretenida y gratificante de aprender inglés a los más pequeños, en la que puedan desde el primero momento ver recompensados sus esfuerzos.

El objetivo de este proyecto consistirá en dotar de nueva funcionalidad a esta aplicación. Por una parte, permitir que la aplicación ***Canciones Inglés*** se conecte con Firebase. Con ello podremos incorporar un método de autentificación además de un sistema de almacenamiento en la nube. Por otra, se incorporará un nuevo modo reproducción "Rellenar palabras" en el que se ocultarán palabras al azar que el usuario tendrá que adivinar.

Motivación

Como se ha indicado en Descripción del problema, es indudable que saber inglés hoy en día es no sólo una herramienta que nos abre muchas puertas si no que resulta casi imprescindible. Para obtener un título universitario, para optar a un puesto de trabajo... cada vez son más los sitios que nos exigen tener un buen nivel de inglés.

Sin embargo, esta progresiva exigencia no ha ido acompaña en la mayoría de los casos (o, al menos, no se ha incrementado al mismo rito) que la mejora de los métodos de enseñanza del idioma.

Esto, unido al reto a nivel de programación que me planteaba el desarrollo de una aplicación de estas características, fueron los principales motivos que me llevaron a decantarme por este proyecto.

Arquitectura de la aplicación

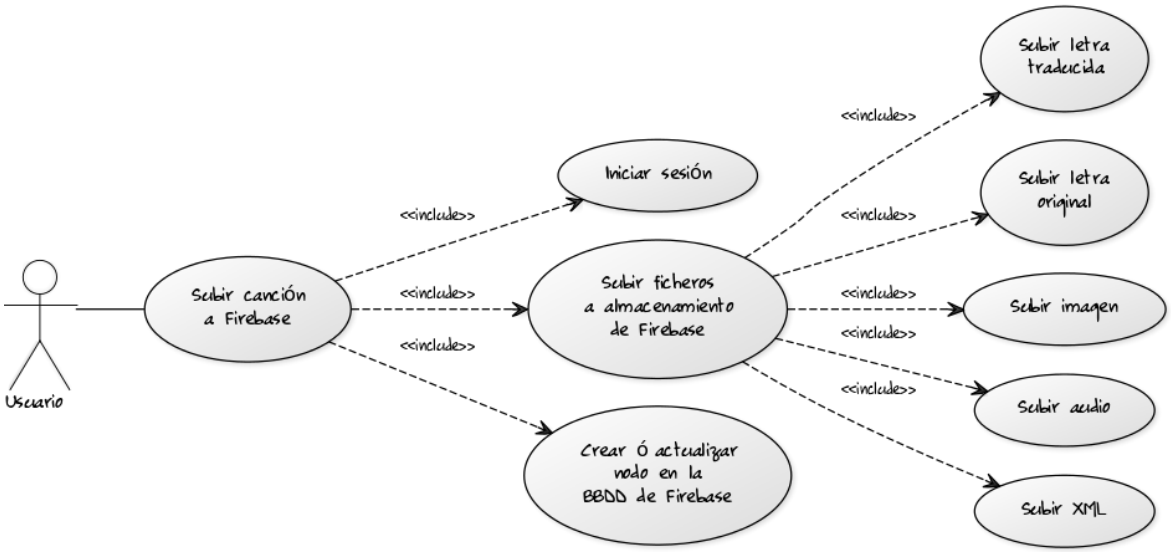
Esquema del diseño

Diagrama con los diferentes componentes del diseño y sus interrelaciones. Justificación de las principales decisiones tomadas en el diseño.

Diagrama de casos de uso

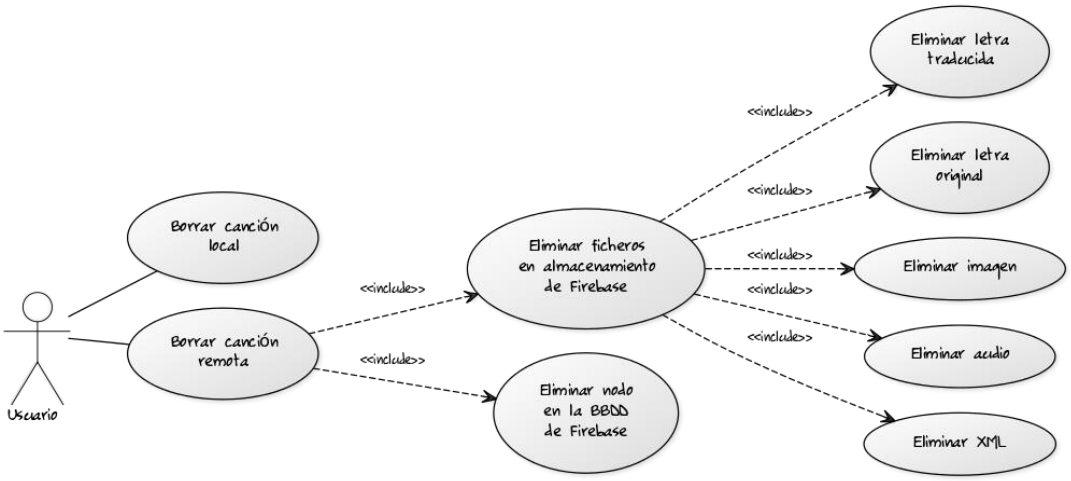
A los casos de uso ya existentes se han añadido:

Subir canción a Firebase

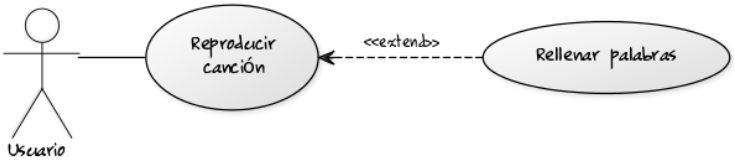


Borrar canción de Firebase

El caso de uso borrar se ha subdividido en dos nuevos casos de uso: *Borrar canción local* y *Borrar canción remota*. Un usuario solo podrá borrar una canción de Firebase si es el propietario de la misma, es decir, si fue él quien la subió.



Reproducir una canción en modo "Rellenar palabras"



Iniciar y cerrar sesión

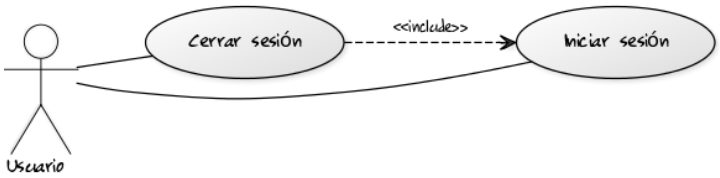


Diagrama de clases

Para ofrecer toda la nueva funcionalidad planteada ha sido necesario reestructurar la arquitectura del proyecto.

FirebaseSingleton

En primer lugar se ha creado una nueva clase, `FirebaseSingleton`, que permite obtener las referencias a las bases de datos, almacenamiento y usuarios de Firebase de manera rápida.

Canción

Para adaptar esta clase a la nueva funcionalidad se ha desarrollado el método `downloadXML()` cuya funcionalidad es análoga a la del método `leerXM()`: leer un XML que se encuentra alojado en una determinada URL (`getXml()`) y parsearlo para crear el objeto canción.

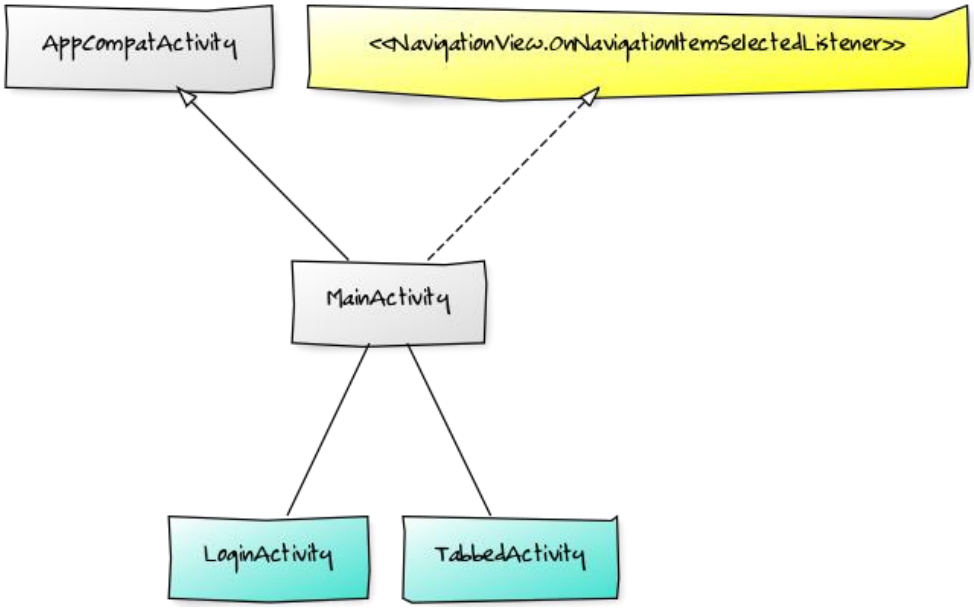
```
public void downloadXML() {
    try{
        URL url = new URL(getXml());
        URLConnection conn = url.openConnection();
        conn.connect();

        InputStream is = conn.getInputStream();
        InputSource inputSource = new InputSource(is);
        SAXParserFactory fabrica = SAXParserFactory.newInstance();
        SAXParser parser = fabrica.newSAXParser();
        XMLReader lector = parser.getXMLReader();
        ManejadorXML manejadorXML = new ManejadorXML();
        lector.setContentHandler(manejadorXML);
        lector.parse(inputSource);

        setEtiquetado(manejadorXML.getCancionXML().getEtiquetado());
        setLetra(manejadorXML.getCancionXML().getLetra());

    } catch (...) {}
}
```

Por otra parte ha sido necesario incluir los siguientes atributos, con sus respectivos métodos `getter` y `setter`, para ajustar su esquema al definido en la base de datos de Firebase.

```
private String audio;           // URL en la que se encuentra el audio
private String id;              // Identificador de la imagen
private String imagen;          // URL en la que se encuentra la imagen
private String txt_original;    // URL en la que se encuentra el fichero
                                // con la letra original
private String txt_traducido;   // URL en la que se encuentra el fichero
                                // con la letra traducida
private String xml;             // URL en la que se encuentra el XML
private String user;            // Usuario que creo/subió la canción
```

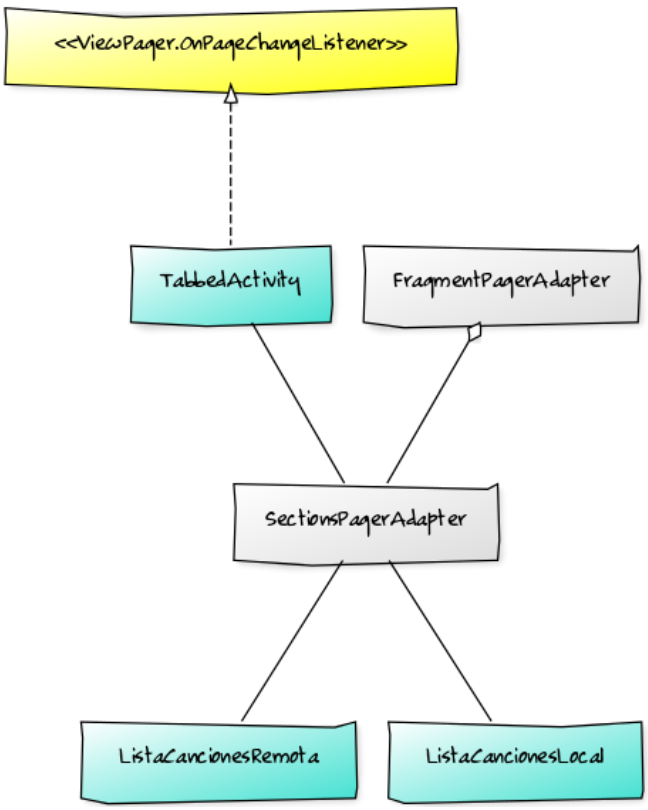
Actividad principal:

Ahora, la actividad principal está formada por un contenedor que puede albergar dos fragments distintos: `LoginActivity` o `TabbedActivity`. Además, se ha incluido un `NavigationDrawer` para permitir cambiar entre estos dos fragments, mostrar la pantalla de preferencias o visualizar el dialogo "**Acerca de…**".



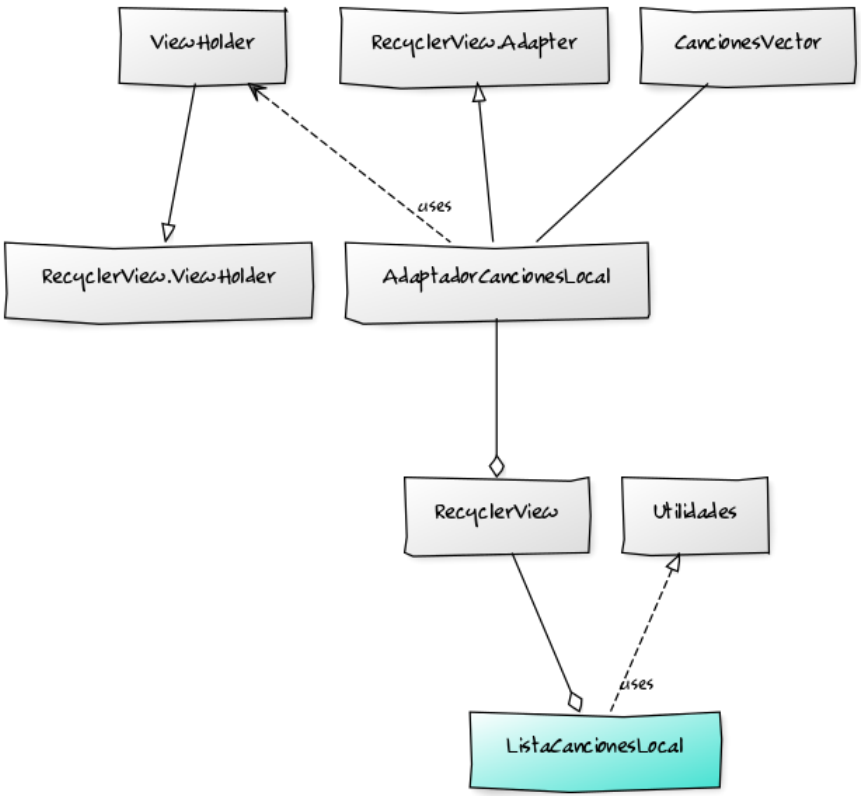
TabbedActivity

Este fragment está formado por un `ViewPager` y su correspondiente adaptador, `SectionsPagerAdapter`, el cuál tendrá asociado un `TabLayout` que se encargará de señalar qué fragment esta visible en cada momento: `ListaCancionesLocal` o `ListaCancionesReomoto`.



Listar de canciones guardadas en el sistema de archivos del disposivo

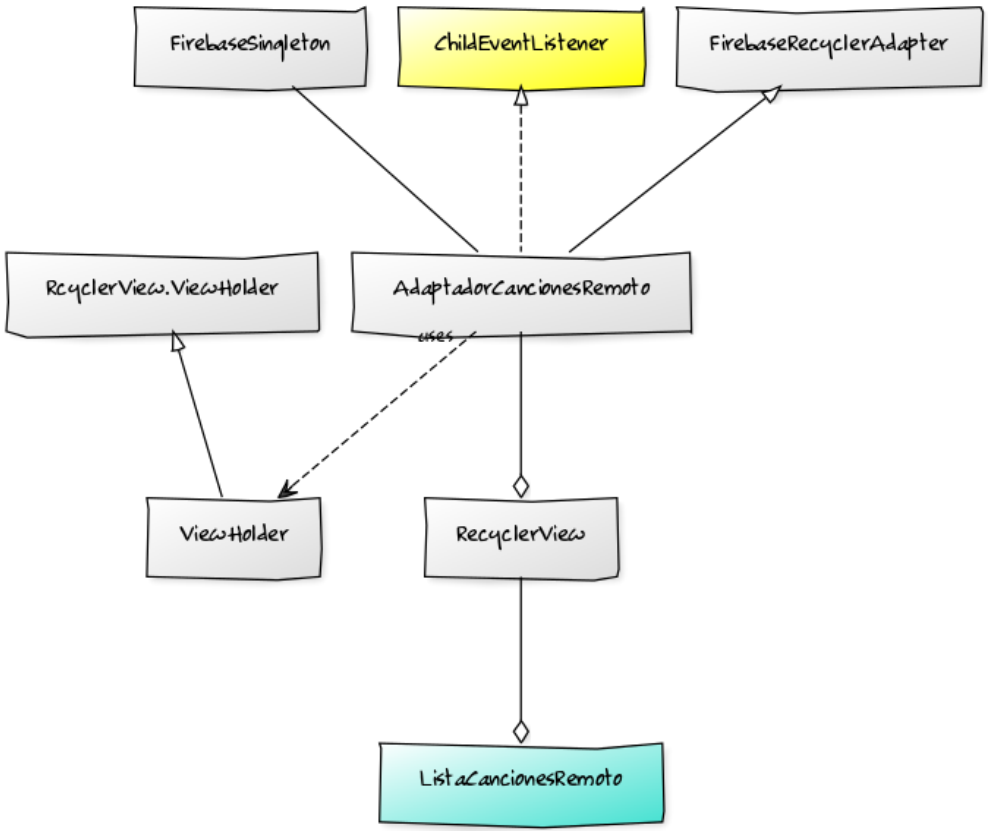
El fragment `ListaCancionesLocal` es el que contiene toda la funcionalidad que ya existía. En él se listan las canciones que hay en la memoria del dispositivo.



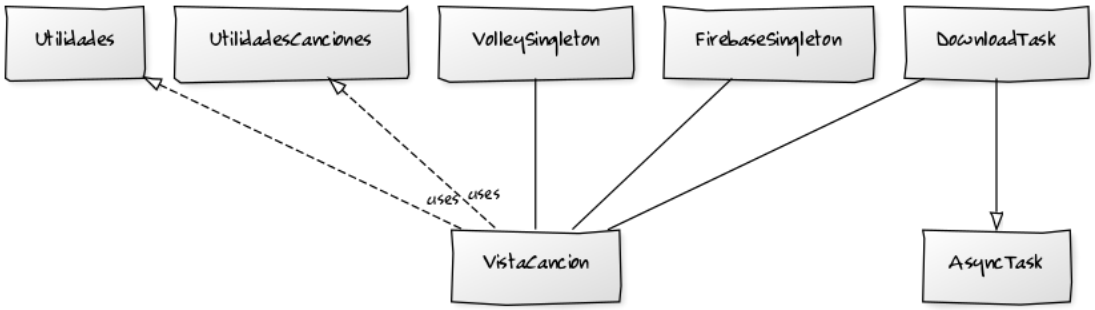
Su implementación se ha completado incluyendo las comprobaciones de permisos oportunas antes de acceder al contenido de la tarjeta SD. Si no se tiene permiso de escritura/lectura, se solicitarán al usuario y sólo se accederá realmente al contenido en el caso de ser concedidos. Esto se ha implementado con la ayuda de la clase Utilidades, explicada más detalladamente en el punto Utilidades.

Listar las canciones disponibles en Firebase

Para mostrar la información almacenada de forma local en el fragment `ListaCancionesLocal`, se hizo uso de un `RecyclerView` y su respectivo adaptador `AdaptadorCancionesLocal`. El fragment `ListaCancionesRemoto` va a tener una estructura similar, aunque se ha adaptado para hacer uso de la biblioteca FirebaseUI. Para ello, se ha sustituido el `RecyclerView` por un `FirebaseRecyclerAdapter`. De esta manera será ahora el método `populateViewHolder` quién se encargue de rellenar la vista para cada elemento de la lista de canciones.



Vista canción



Al pulsar sobre un elemento de la lista de canciones se abre la acividad `VistaCancion`. A esta actividad se le ha añadido la funcionalidad necesaria para que además de visualizar las canciones almacenadas en el dispositivo, sea capaz de visualizar aquellas que se encuentren en la base de datos de Firebase.

Para ello, ha sido necesario sustituir la vista `ImageView` que contenía la portada por una vista `NetworkImageView` que permita visualizar imágenes almacenadas en Firebase, utilizando la librería `Volley`:

```
VolleySingleton volleySingleton = VolleySingleton.getInstance(mContext);
ImageLoader imageLoader = volleySingleton.getLectorImagenes();
portada.setImageUrl(cancion.getImagen(), imageLoader);
```

Por otro lado, se ha incluido control sobre el botón "**Volver atrás**" del dispositivo para evitar que la actividad se cierre directamente cuando se está ejecutando alguno de los modos de reproducción. Para conseguirlo se ha sobreescrito el método `onBackPressed` de manera que si estamos en medio de una reproducción, esta finaliza sin salir de la actividad.

```
@Override
public void onBackPressed() {
    if (!corriendo) {
        finish();
        return;
    }

    // ...
    mostrarMensaje(mContext, "Pulse otra vez para volver a la lista");
    corriendo = false;
    ponInfoCancion((int)id, false);
}
```

Además, se ha incluido la siguiente nueva funcionalidad: **Descargar canción**, **Subir canción**, **Eliminar canción del servidor** y **Rellenar palabras**.

Descargar una canción

Si el usuario selecciona está opción en el menú de la actividad, lo primero que se hará será comprobar si se tiene permiso de escritura en la tarjeta SD y, en caso de que no sea así, solicitarlos. Esto se hace mediante la clase Utilidades, explicada más detalladamente en el punto Utilidades.

Para implementar esta funcionalidad se ha optado por implementar la clase `DownloadFile`, descendiente de `AsyncTask`, para evitar el bloqueo del hilo principal. Esta clase se encargará de, dada una `Canción`, descargar todos los archivos asociados a la misma:

- El fichero de audio
- La imagen de la portada
- El fichero de texto con la letra original
- El fichero de texto con la letra traducida
- El fichero XML que contiene todos los datos de la canción (etiquetado, etc.)

Durante el proceso de descarga el hilo principal quedará a la espera, mostrando al usuario una barra de progreso, `ProgressBar`, que señalizará el avance del proceso.

```
@Override
protected String doInBackground(Cancion... params) {
    // ...
    download(cancion.getAudio(), nombreFichero + EXTENSION_AUDIO);
    publishProgress(1);
    download(cancion.getImagen(), nombreFichero + EXTENSION_IMAGEN);
    publishProgress(2);
    // ...
}

public void download(String strUrl, String nombreFichero)
                                                    throws IOException {
    URL url = new URL(strUrl);
    URLConnection conection = url.openConnection();
    conection.connect();

    InputStream input = new BufferedInputStream(url.openStream(),8192);
    String filePath = BASE_PATH + "/cancionesingles/"+ nombreFichero;
    OutputStream output = new FileOutputStream(filePath);

    int count;
    byte data[] = new byte[1024];
    while ((count=input.read(data))!=-1) output.write(data, 0, count);

    output.flush();
    output.close();
    input.close();
}
```

Una vez finalizada la descarga de todos los archivos se insertará en el array `CancionesVector` el nuevo elemento:

```
@Override
protected void onPostExecute(String file_url) {
    // ...
    CancionesVector cancionesVector = CancionesVector.getInstance();
    cancionesVector.anyade(cancion);
}
```

Cuando la tarea `DownloadTask` haya terminado su ejecución se notificará al adaptador `AdaptadorCancionesLocal` que se ha insertado un nuevo elemento y se mostrará la lista de canciones descargadas.

```
final DownloadFile task = new DownloadFile(mContext);
task.execute(cancion);
final Thread waitingThread = new Thread(Runnable()→ {
  try {
      while(task.getStatus() != AsyncTask.Status.FINISHED) {
          TimeUnit.SECONDS.sleep(1);
      }
      runOnUiThread(() → {
        int position = CancionesVector.getInstance().tamanyo() - 1;
        ListaCanciones.adaptador.notifyItemInserted(position);
        setResult(CANCION_DESCARGADA);
        finish();
      });
  } catch (InterruptedException e) { ... }
});
```

Subir una canción

Para poder subir una canción es necesario estar autenticado en Firebase. Si al seleccionar está opción en el menú el usuario no lo está se le preguntará si desea identificarse. En caso afirmativo, se abrirá la actividad de Iniciar sesión.

Una vez identificado el usuario, se procederá a subir la canción. Para ello se han incluido en la clase `UtilidadesCanciones` los métodos necesarios. En primer lugar, hay que comprobar si la canción ya existe. Para ello nos serviremos del método `getDownloadUrl()` que proporciona la API de Firebase y que devuelve una `UploadTask`, una tarea asincrónica que devuelve una URL que se puede usar para descargar el objeto. Si la tarea termina en fallo será porque no existe ningún fichero alojado en el nodo.

```
public static void subirCancionAFireBase (final Cancion cancion,
        final ProgressDialog progressDialog, final Context mContext) {
    // ...
    String localPath = cancion.getAudio();
    Uri file = Uri.fromFile(new File(localPath));
    String nodo = titulo + file.getLastPathSegment();
    StorageReference fileRef = storageReference.child(nodo);
    Task<Uri> task = fileRef.getDownloadUrl();
    task.addOnSuccessListener((OnSuccessListener)(uri) → {
        progressDialog.dismiss();
        mostrarMensaje(mContext, "¡El archivo ya existe!");
    }).addOnFailureListener((exception)→ {
        subirFicheros(cancion, progressDialog, mContext);
    });
}
```

Si ha habido éxito se pasará a descargar todos los archivos asociado a la canción:

```
//...
subiendoDatos[index] = true;
StorageReference fileStorageRef = cancionStorageRef.child(fileName);
UploadTask uploadTask = fileStorageRef.putFile(file);
uploadTask.addOnFailureListener((exception)→ {
        subiendoDatos[index] = false;
        errorSubiendoDatos[index] = true;
        updateProgressDialog(progressDialog, mContext);
}).addOnSuccessListener((OnSuccessListener)(taskSnapshot)→{
        subiendoDatos[index] = false;
        errorSubiendoDatos[index] = false;
        Uri downloadUrl = taskSnapshot.getDownloadUrl();
        String nodo = getNodoByFilename(fileName);
        DatabaseReference nodoRef = fileDataBaseRef.child(nodo);
        nodoRef.setValue(downloadUrl.toString());
        updateProgressDialog(progressDialog, mContext);
});
```

Mientras se hace la subida el hilo principal permanecerá a la espera. Una vez haya finalizado la subida de todos los archivos se creará el nodo correspondiente en la basa de datos de firebase.

```
final Thread waitingThread = new Thread((Runnable)→{
  try {
    int count = 5;

    while(count > 0) {
      count = 0;
      for (int i = 0; i < subiendoDatos.length; i++)
          count = subiendoDatos[i] ? count + 1 : count;
      TimeUnit.SECONDS.sleep(1);
    }

    databaseRef.child("titulo").setValue(cancion.getTitulo());
    databaseRef.child("autor").setValue(cancion.getAutor());
    databaseRef.child("dificultad")
                        .setValue(cancion.getDificultad().ordinal());
    databaseRef.child("genero")
                        .setValue(cancion.getGenero().ordinal());
    databaseRef.child("id").setValue(cancion.getId());
    String uid = FirebaseSingleton.getInstance()
                    .getCurrentUser().getUid();
    databaseRef.child("user").setValue(uid);

  } catch (InterruptedException e) { ... }
});

waitingThread.start();
```

Eliminar una canción remota

Si el usuario abre una canción que ha sido subida por él al servidor el menú le dará la posibilidad de eliminarla. Para eliminar una canción de Firebase, primero se eliminarán todos los ficheros del almacenamiento de Firebase. Para ello se han incluido en la clase `UtilidadesCanciones` los métodos necesarios:

```
public static void borrarArchivoRemoto(
 StorageReference cancionRef, final String path, final int i) {
    Uri uri = Uri.parse(path);
    final String nodo = uri.getLastPathSegment().toLowerCase();

    StorageReference archivoRef = cancionRef.child(nodo);
    archivoRef.delete()
      .addOnSuccessListener((OnSuccessListener)(aVoid)→{
            borrandoDatos[i] = false;
            errorBorrandoDatos[i] = true;
    }).addOnFailureListener((exception)→{
            borrandoDatos[i] = false;
            errorBorrandoDatos[i] = true;
    });
}
```

Al igual que al subir una canción, durante el borrado el hilo principal permanecerá a la espera. Una vez eliminados todos los ficheros, se eliminará el nodo asociado a la canción de la base de datos de Firebase:

```
final Thread waitingThread = new Thread(Runnable()→ {
    try {
        int count = 5;
        while(count > 0) {
            count = 0;
            for (int i = 0; i < borrandoDatos.length; i++)
                count = borrandoDatos[i] ? count + 1 : count;

            runOnUiThread(()→ { progressDialog.setMessage(…); });
            TimeUnit.SECONDS.sleep(1);
        }
    } catch (InterruptedException e) { ... }

    // ...
    DatabaseReference cancionesRef = fbSinglt.getCancionesReference();
    cancionesRef.child(nombre).removeValue();
    finish();
});

waitingThread.start();
```

Rellenar palabras

En la vista "Canción" se ha incluido un nuevo modo de reproducción: **Rellenar palabras**. Su implementación y gestión se ha realizado siguiendo el esquema de los modos de reproducción ya implementados anteriormente: mediante "un hilo secundario (MiThread) para evitar que se bloquee el hilo principal de ejecución (hilo de interfaz)" [2].

Antes de empezar, el hilo principal se encarga de seleccionar de manera aleatoria qué palabras van a ser ocultadas durante la reproducción.

```
private void onPreExecuteModoRellenar() {
  frasesARellenar = new String[cancion.getLetra().size()];
  palabrasOcultas = new String[cancion.getLetra().size()];

  for (int i = 0; i < frasesARellenar.length; i++) {
    List<Frase> letra = cancion.getLetra();
    String fOriginal = letra.get(i).getFraseOriginal();
    String[] palabras = fOriginal.split(" ");

    int pos = (int)(Math.random() * palabras.length);
    palabrasOcultas[i] = palabras[pos].trim().toLowerCase();
    frasesARellenar[i] = fOriginal.replaceAll(palabras[pos], "_");

  // ...
  mediaPlayer.start();
}
```

Cuando el hilo secundario empieza su ejecución, la canción empieza a reproducirse hasta que finaliza la primera frase. En ese momento, se pausa la reproducción y se espera a que el usuario introduzca la palabra oculta. La ejecución se reanudará cuando el usuario adivine la palabra o bien "se rinda" y pida que sea descubierta. Además, la frase actual podrá ser repetida tantas veces como quiera el usuario.

```
while (mediaPlayer.getCurrentPosition() <= frase.getTiempoFin() &&
corriendo) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            if (contador<cancion.getLetra().size() && contador >= 0) {
                fraseActual = contador;
                fOriginal.setText(frasesARellenar[contador]);
            }
        }
    });
}

// ...
mediaPlayer.pause();

try {
    while (!goOnClicked) {
        sleep(1000);
        if (repetirFrase) {
            mediaPlayer.seekTo(frase.getTiempoIni());
            goOnClicked = true;
            contador--;
        }
    }
} catch (InterruptedException e) { ... }

mediaPlayer.start();
contador++;
```

Utilidades

Para facilitar la solicitud de permisos desde las distintas actividades se ha creado la clase Utilidades en la que se ha incluido los métodos necesarios para comprar si se han concedido permisos necesarios:

```
public static boolean isPermissionGranted (String permission,
                                           Activity activity) {
    return ContextCompat.checkSelfPermission(activity,
            permission) == PackageManager.PERMISSION_GRANTED;
}
```

o solicitarlos si no es así:

```
public static void requestPermission (final String permission,
                                      final Activity activity) {
   boolean requestRationale = ActivityCompat
               .shouldShowRequestPermissionRationale(activity, permiss

   if (!requestRationale) {
       ActivityCompat.requestPermissions(
           activity, new String[] { permission},getCode(permission));
       return;
   }

   new AlertDialog.Builder(activity)
       .setTitle(null)
       .setMessage(getDialogTitle(permission))
       .setPositiveButton(R.string.confirmar, (dialog, wichButton) →{
              ActivityCompat.requestPermissions(
                   activity, new String[] { permission},
                   getCode(permission));
        })
        .setNegativeButton(R.string.cancelar, null).show();
}
```

Modelo de datos

Al modelo de datos ya existente basado en ficheros XML hay que añadir todo lo referente al almacenamiento de datos en Firebase.

Estructura de la base de datos de Firebase:

- `canciones`: Lista de canciones
- `usuarios`: Lista de usuarios

Estructura de un usuario:

Los datos de un usuario estarán accesibles en el nodo de la base de datos que tenga como clave el identificador del usuario.

- `email`: dirección de correo electrónico con la que se registró
- `name`: nombre del usuario

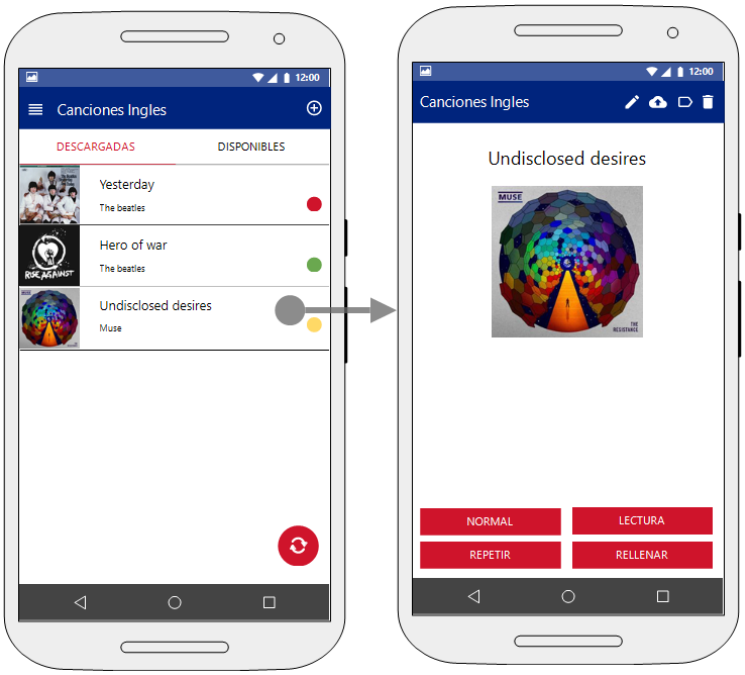
Estructura de una canción:

Los datos de una canción estarán accesibles en el nodo de la base de datos que tenga como clave el nombre de dicha canción, sin espacios y en minúsculas.

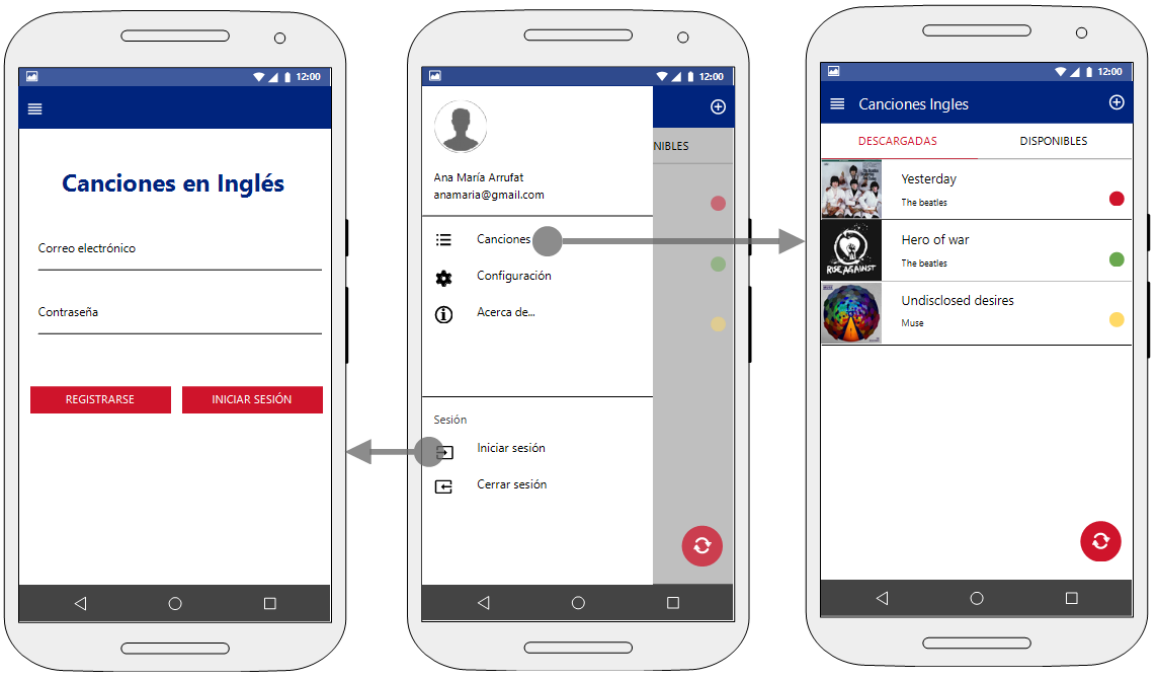
- `audio`: url en la que se encuentra el audio
- `autor`: autor de la canción
- `dificultad`: puede tomar los valores 1, 2 ó 3
- `genero`: género musical al que pertenece la canción
- `id`: identificador
- `imagen`: url en la que se encuentra el archivo con la imagen de la portada
- `titulo`: título de la canción
- `txt_original`: url en la que se encuentra el archivo con la letra original
- `txt_traducido`: url en la que se encuentra el archivo con la letra traducida
- `user`: identificador del usuario que creó la canción
- `xml`: url en la que se encuentra el archivo xml con los datos de la canción

Vistas

TabbedActivity



Navigation Drawer e inicio de sesión



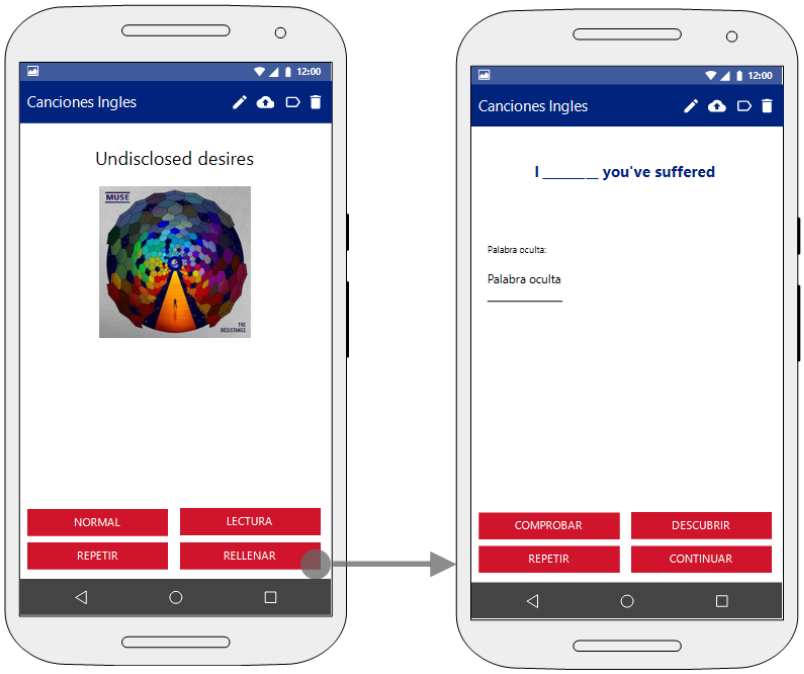
Vista canción

Al seleccionar la opción “Subir” se comprobará si el usuario está autentificado. Si no lo está se le preguntará si desea identificarse. En caso afirmativo, se le rederigirá a la vista de Inicio de sesión. En caso contrario, no se le permitirá subir la canción a Firebase.

Rellenar palabras

El nuevo modo de reproducción proporciona cuatro botones que nos permitirán realizar cada una de las acciones descritas en la sección Vista canción.

Cuando el hilo secundario empieza su ejecución, la canción empieza a reproducirse hasta que finaliza la primera frase. En ese momento, se pausa la reproducción y se espera a que el usuario introduzca en la caja de texto la palabra oculta. El usuario podrá repetir la frase actual tantas veces como quiera pulsando el botón *REPETIR*. La ejecución no podrá reanudarse hasta que el usuario adivine o "descubra", mediante el botón *DESCUBRIR*, la palabra oculta.



Conclusiones

Como conclusión, podemos afirmar que se ha conseguido el objetivo inicial. Se ha desarrollado una aplicación con la que los más pequeños pueden aprender inglés de una manera más interactiva que la que ofrece la metodología tradicional.

Como posibles trabajos futuros podríamos:

- Realizar mejoras en el modo Rellenar de manera que:
  - Incluya un sistema de puntuación
  - Permita elegir el grado de dificultad (ocultando más o menos palabras, …)
  - Evitar que se oculten palabras que aportan poco valor como, por ejemplo, *the, you, I, a*, etc.
- Incluir nuevos modos de reproducción
- Permitir editar canciones directamente en el servidor

Bibliografía

[1] Fernando Bejarano. *Saber inglés, clave para sacarle mayor provecho a Internet*. [En línea] http://www.elpais.com.co/mundo/saber-ingles-clave-para-sacarle-mayor-provecho-a-internet.html, 2017

[2] Gómez Sacedón, Mario. *Canciones Ingles*. [En línea] http://www.androidcurso.com/index.php/teleformacion/proyectos-finales/72-proyectos-finales-2016/master-moviles/701-canciones-ingles, 2016

Anexos

Código fuente en GitHub

Todo el código fuente está disponible en GitHub a través del siguiente enlace: `https://github.com/jesus-tomas-girones/CancionesIngles.git`