Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Sección de Ingeniería Informática

SchoolApp

Comunicación desde las aulas

Gonzalo J. García Martín

Trabajo de Fin de Grado
La Laguna, 23 de Septiembre, 2015

Índice

Índice

Introducción

- **Tutor académico:** Francisco de Sande González
- **Línea de trabajo del proyecto:** Programación de aplicaciones interactivas en *Android*.

Introducción

Objetivos

- Conocimientos del Grado en Ingeniería Informática.
- Conocimientos sobre *Android*.
- Diseño y desarrollo de un proyecto.
- Conocimientos sobre el uso de servicios en la nube y su utilización en aplicaciones *Android*.
- Repositorio online.
- Creación de una memoria técnica.
- LaTeX.

Índice

Especificación de Requisitos

Funcionalidades

- Registro (Roles).
- Contactos.
- Notificaciones.
- Circulares.
- Modificación de datos.
- Baja de Usuario.

Comunicaciones Permitidas

	Padre	Alumno	Profesor
Padre	Mensajes	-	Mensajes
Alumno	-	Mensajes	Mensajes
Profesor	Circulares y Mensajes	Circulares y Mensajes	Mensajes

Índice

Herramientas de Software

Android Studio: `http://developer.android.com/sdk`

- Entorno de desarrollo integrado.
- Uso intuitivo.
- Ligero.

Tutoriales

- *Suporting different devices*.
- *Building a dynamic UI with fragments*.
- *ActionBar Tab Swipe*.
- *TabHost Swipe*.
- *Adding animations*.
- *Expandable ListView*.

Herramienta Software

Firebase

- Proveedor de contenidos.
- Ofrece servicios en la nube.
- Seguro y sencillo (Funcionalidades).
- Datos tipo JSON.
- Consturucción de datos en tiempo real.
- Interfaz de programación de aplicaciones de seguridad basada en la expresión altamente flexible.

Tutorial

- *Firebase Storage*:
  `https://www.firebase.com/docs/android/guide/`

Herramientas de Software

Parse

- Proveedor de servicios en la nube.
- Alternativa a *Firebase*.
- Utiliza datos SQL.
- Múltiples SDK.
- Notificaciones tipo *Push*.

Tutorial

- *Parse Storage*: `http://www.sitepoint.com/creating-cloud-backend-android-app-using-parse/`

Índice

Aplicaciones Similares

Remind

- Forma sencilla de enviar mensajes a estudiantes y padres.
- Los profesores, monitores o administradores pueden enviar recordatorios, deberes o mensajes motivadores.
- Seguro, teléfonos confidenciales.
- Los profesores pueden enviar anuncios o mensajes tipo chat.

Aplicaciones Similares

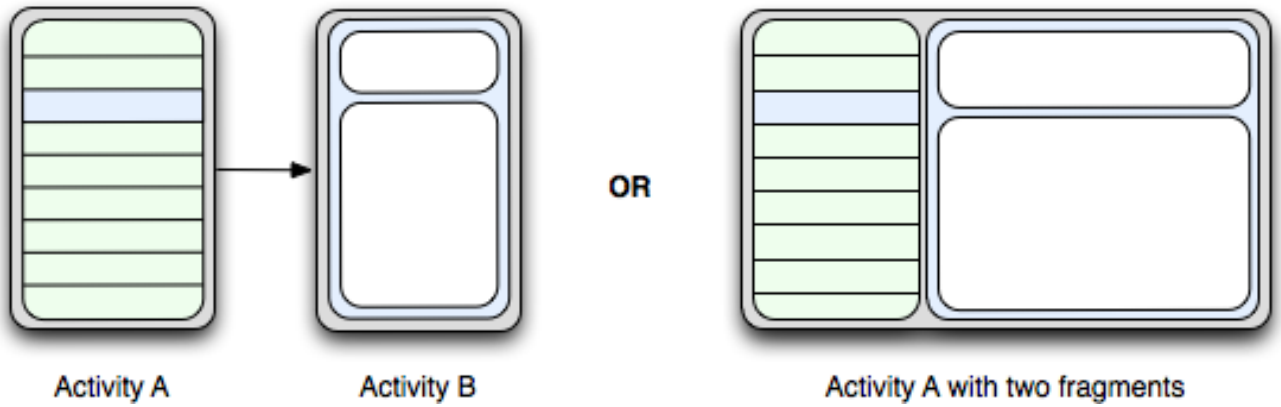
miColegioApp

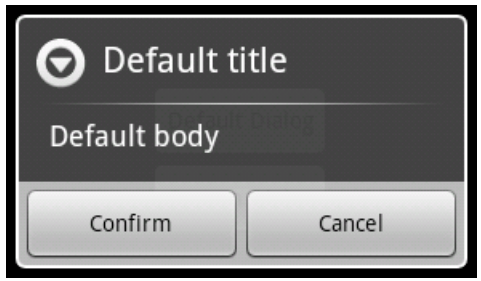
- El colegio contrata los servicios.
- Registro de usuarios.
- Introducir el código del centro en la aplicación.
- Se asocian a los usuarios con un centro.
- Los usuarios pueden recibir notificaciones, circulares y boletines.

Descripción

Definición de términos

- *Activity*.
- *Fragment*.
- *Dialog*.





Descripción

Descripción

Vídeo ilustrativo del funcionamiento de la aplicación.

Índice

Desarrollo de la Aplicación

Clases Principales

- Clase `Student`.
- Clase `Father`.
- Clase `Teacher`.
- Clase `Message`.

Ejemplo Student

```
1  //...
2  public class Student implements Parcelable {
3      private String name;
4      private String lastname;
5      private String school;
6      private String classroom;
7      private String mail;
8      private String telephone;
9      private String dni;
10     private String rol = "Alumno";
11     public Student(Map<String, Object> values) {
12         setName((String) values.get("nombre"));
13         setLastname((String) values.get("apellido"));
14         /*...*/ } }
```

Ejemplo Student

```
1  //...
2  public class Student implements Parcelable { //...
3      //*****Parte de la interfaz Parcelable*****//
4      public Student(Parcel in) { readFromParcel(in); }
5      @Override
6      public int describeContents() { return 0; }
7      @Override
8      public void writeToParcel(Parcel dest, int flags) {
9          dest.writeString(name);
10         dest.writeString(lastname);
11         /*...*/ }
12     public void readFromParcel(Parcel in) {
13         name = in.readString();
14         lastname = in.readString();
15         school = in.readString(); /*...En el mismo orden que writeToParcel*/}
16     public static final Parcelable.Creator CREATOR=new Parcelable.Creator(){
17         @Override
18         public Student createFromParcel(Parcel in){return new Student(in);}
19         @Override
20         public Student[] newArray(int size){return new Student[size];}};/*
               Parcelable.creator*/}
```

Base de datos en el dispositivo `MessageSQLHelper`

Tabla Conversations

- Id (Clave primaria).
- D.N.I del destinatario.
- D.N.I del remitente.

Tabla Messages

- Id (Clave Primaria).
- Id de la conversación.
- D.N.I del remitente.
- Nombre del remitente.
- Datos del del envío del mensaje.

Ejemplo MessageSQLHelper

```
1  //...
2  public class MessageSQLHelper extends SQLiteOpenHelper {/*...*/
3      @Override
4      public void onCreate(SQLiteDatabase db) {
5          String createMessagesTable = "CREATE TABLE messages ( " +
6                  "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
7                  "idConversation TEXT, " + "dniRemitter TEXT, " +
8                  "remitter TEXT, " + "day TEXT, " +
9                  "month TEXT, " + "year TEXT, " +
10                 "hour TEXT, " + "minutes TEXT, " +
11                 "message TEXT )";
12         String createConversationsTable = "CREATE TABLE conversations ( " +
13                 "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
14                 "dniAddressee TEXT, " + "dniRemitter TEXT )";
15         //Creando BBDD
16         db.execSQL(createMessagesTable);
17         db.execSQL(createConversationsTable);}
18     /*Operaciones para obtener mensajes, conversaciones, ...*/
19     //...
20 }//class
```

Interacción con la base de datos en *Firebase*

Base de datos en la nube

1. En `app/build.gradle` añadir:
   - `compile 'com.firebase:firebase-client-android 2.2.1'`
2. Importar los servicios a usar en los archivos de clase (`java`).
3. En `onCreate()` incorporar:
   - `Firebase.setAndroidContext(this)`
4. Colocar una referencia a la base de datos o a una de sus tablas.
5. Crear la consulta.
6. situar un *oyente* (`listener`).

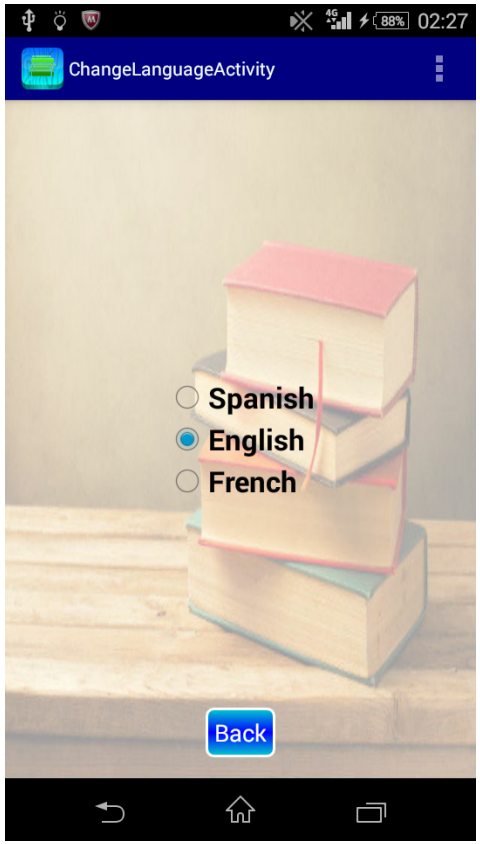
Ejemplo interacción con la base de datos (Obtener datos)

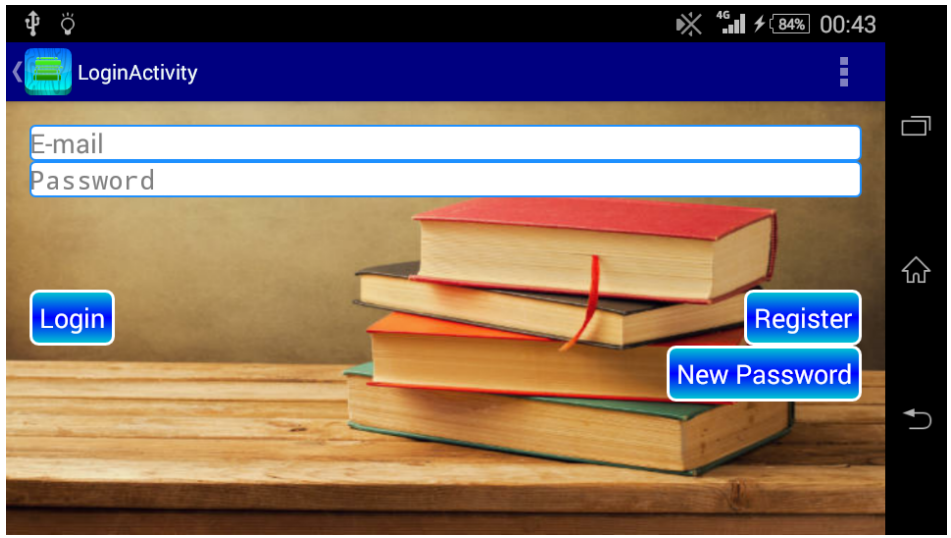
```
1  //import ChildEventListener, DataSnapshot, Firebase, FirebaseError y Query;
2  public class StudentActivity extends ListActivity {
3      /*...*/ Firebase aluRef;
4      public void onCreate (Bundle savedInstanceBundle) {
5          super.onCreate(savedInstanceBundle);
6          Firebase.setAndroidContext(this);
7          aluRef = new Firebase (getString(R.string.studentRef));/*...*/}
8      public void preparingData(){ //Obtener los Student de la misma clase
9          Query getClassmates = aluRef.orderByChild(getString(R.string.
               bbdd_center)).equalTo(school);
10         getClassmates.addChildEventListener(new ChildEventListener() {
11             @Override
12             public void onChildAdded(DataSnapshot dataSnapshot,String s){}
13             @Override
14             public void onChildChanged(DataSnapshot dataSnapshot,String s){}
15             @Override
16             public void onChildRemoved(DataSnapshot dataSnapshot){}
17             @Override
18             public void onChildMoved(DataSnapshot dataSnapshot, String s){}
19             @Override
20             public void onCancelled(FirebaseError firebaseError){} });
21         }}//class
```

Ejemplo interacción con la base de datos (Almacenar datos)

```
1  //...
2  public class AddChildActivity extends Activity {
3      Firebase childRef;
4      @Override
5      protected void onCreate(Bundle savedInstanceState) {
6          Firebase.setAndroidContext(this);
7          childRef=new Firebase(getString(R.string.studentRef));/*...*/}
8      public void addChild (View vew) {
9          if (!haveEmptyFields()) { //Obtencion de datos del XML
10             Query existStudent = childRef.//...;
11             existStudent.addListenerForSingleValueEvent(new
                   ValueEventListener() {
12                 @Override
13                 public void onDataChange(DataSnapshot dataSnapshot) {
14                     //Student not Exist
15                     if (dataSnapshot.getValue() == null) {
16                         Map<String, Object> studentMap = new HashMap<>();
17                         studentMap.put(getString(R.string.bbdd_name), name);
18                         uuid = UUID.randomUUID().toString();
19                         childRef.child(uuid).setValue(studentMap); }
20                 }/*...*/ }); }} }//class
```

Idiomas y Formato Horizontal





Índice

Problemas

Problemas

1. *Android Studio* no render Target.
2. Fallo al encontrar `com.android.support:appcompat-v7:16.+`.
3. Fallo al encontrar *Java*
4. Duplicidad en la dependencias de los paquetes.
5. Fallo al encontrar `com.android.support:support-v13`.
6. Arregle `Gradle` (Fix `Gradle`).
7. *SDK* no encontrado.
8. Consultas a la espera de modificación de datos.

Fichero build.gradle

```
1  // Top-level build file where you can add configuration options common to
       all sub-projects/modules.
2
3  buildscript {
4      repositories {
5          jcenter()
6      }
7      dependencies {
8          classpath 'com.android.tools.build:gradle:1.0.0'
9
10         // NOTE: Do not place your application dependencies here; they belong
11         // in the individual module build.gradle files
12     }
13 }
14
15 allprojects {
16     repositories {
17         jcenter()
18     }
19 }
```

Fichero app/build.gradle

```
1  apply plugin: 'com.android.application'
2  android { //...
3      defaultConfig { /*...*/ }
4      buildTypes {
5          release {
6              minifyEnabled false
7              proguardFiles getDefaultProguardFile('proguard-android.txt'), '
                   proguard-rules.pro'
8          }
9      }
10     packagingOptions {
11         exclude 'META-INF/LICENSE'
12         exclude 'META-INF/LICENSE-FIREBASE.txt'
13         exclude 'META-INF/NOTICE'
14     }
15 }
16 dependencies {
17     compile fileTree(dir: 'libs', include: ['*.jar'])
18     compile 'com.android.support:appcompat-v7:21.0.3'
19     compile 'com.android.support:support-v4:21.0.3'
20     compile 'com.android.support:support-v13:21.0.3'
21     compile 'com.firebase:firebase-client-android:2.2.1'
22 }
```

Índice

Summary and conclusions

Conclusions

The realization of this project will apply the expertise gained during the years of study, allowing real assimilation of the necessary skills and acquire new knowledge. This experience discloses the level of involvement that involves creating a fully functional application in a real environment. The analysis of other similar applications can realize the state of the market for the type of application built.

An application program may seem simple, but it takes dedication, time and hard work.

Gracias por su atención

SchoolApp

- Repositorio en *github*:
  `https://github.com/alu0100403619/TrabajoFinDeGrado`

Gonzalo J. García Martín
alu0100403619@ull.edu.es