IPCA INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE

Relatório de Trabalho Prático

The Teacher Planning App – Android App

Mestrado em Engenharia Informática

Disciplina:

Programação de Dispositivos Móveis

Professor:

Nuno Oliveira

Trabalho elaborado por:

Rui Oliveira - Nº 3698

Tiago Silva - Nº 6130

Pedro Oliveira - Nº 8684

André Monteiro - Nº 16202

23/07/2019

1) Sumário

O presente relatório descreve a actividade desenvolvida pelo grupo de trabalho para a disciplina de Programação de Dispositivos Móveis onde, ao longo do semestre, foi desenvolvida uma aplicação dirigida a dispositivos móveis, na plataforma Android, cujo intuito central é, potencialmente, ajudar um docente na gestão da sua vida académica, nomeadamente na administração das várias disciplinas que este poderá lecionar, integradas nos mais diversos cursos, assim como, facilitar e promover o contato mais directo e prático com os alunos, evitando, por exemplo, que o docente tenha que que recorrer a emails dirigidos à turma, sob a forma de avisos/alterações/notificações. Facilita assim na definição e nas eventuais alterações das salas de aula onde deverão ser lecionadas as suas disciplinas. A nossa disciplina, integrada no programa curricular do mestrado em Eng. Informática, assume-se como uma mais valia na compreensão do vasto mundo do desenvolvimento de aplicações para dispositivos móveis oferecendo, para muitos de nós, um primeiro contacto com a plataforma Android, cobrindo de forma transversal diferentes tópicos, de diferentes pertinências, que poderam ser explorados por nós neste trabalho prático. Entendemos que tivemos a possibilidade de observar, apreender e implementar diversas técnicas para o desenvolvimentos das aplicações mobile, tendo por base o IDE Android Studio, bem como adquirir o lingo próprio do desenvolvimento da programação para Android.

2) Introdução e objectivos

A plataforma Android é uma plataforma aberta, com base em Linux, extremamente versátil e é, hoje em dia, a maior referência em termos de sistema operativo para aplicações móveis em todo o mundo, com cerca de 76%[1] de market share. Não foi a primeira plataforma de aplicações em dispositivos móveis mas rapidamente se assumiu como a principal, tendo vindo a revelar cada vez mais e mais a sua força pela mão da Google desde 2005.

Uma das suas vantagens é a abertura e flexibilidade que permite altos níveis de customizações, tanto por parte dos developers como dos utilizadores finais, facilitando também uma transversal integração das aplicações nos mais diferentes dispositivos que suportam a plataforma Android, como é o caso, para além dos telemóveis e tablets, de televisores, carros, relógios, alarmes, etc...A plataforma Android rivaliza assim a com a segunda plataforma mais adotada (cerca de 22% de market share[2]), o iOS, da Apple, e suas respectivas derivações.

O objectivo da aplicação em que trabalhamos foi desenvolver os nossos conhecimentos na plataforma de programação Android, apreender a complexa utilização/navegação de um dos seus softwares de desenvolvimento, o Android Studio, desenvolver um visual/design simples com uma boa 'user experience' bem como procurar ultrapassar os diversos desafios que se foram apresentando com a natural evolução das ideias para o UI da aplicação. Entendemos que esta cumpre os objectivos que nos foram propostos acabando por ser uma forma simples/fundamental e relativamente intuitiva, para que um ou vários professores possam fazer uma gestão/organização das suas diferentes disciplinas, integradas em eventuais diversos cursos, ao longo dos anos, com base nas necessidades potencialmente dinâmicas da suas agendas.

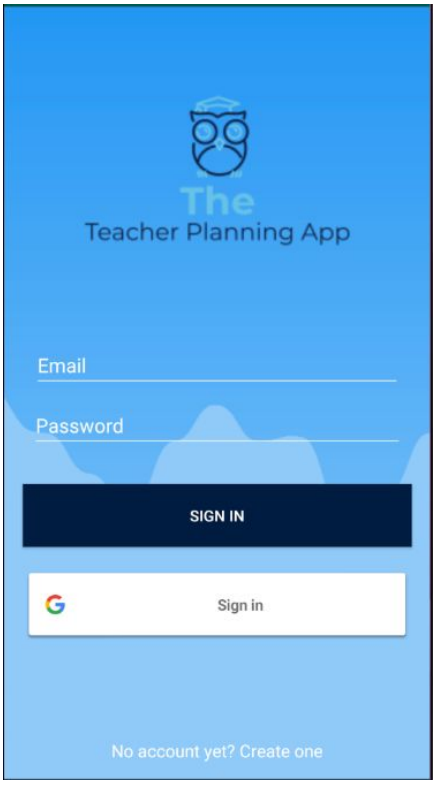
[1] http://gs.statcounter.com/os-market-share/mobile/worldwide
[2] http://gs.statcounter.com/os-market-share/mobile/worldwide

3) Arquitectura da solução e da aplicação

A principal decisão em termos de arquitectura da solução foi a definição da base de dados e respectiva forma de guardar e obter dados registados pelos utilizadores. Decidimos implementar uma API RESTfull, utilizando a framework Spring e a base de dados MySql. A decisão da não utilização da base de dados Realm deveu-se à maior familiarização com a definição do modelo de dados relacional bem como à menor complexidade associada na criação dos serviços REST, fruto da experiência profissional de alguns elementos do grupo. O facto de podermos fazer queries SQL diretamente à base de dados também permitiu que fosse mais fácil detectar detectar problemas e erros no nosso desenvolvimento/dados. Entendemos também que este tipo de arquitetura oferece maior flexibilidade e escalabilidade embora possa se tornar um pouco mais complexo.

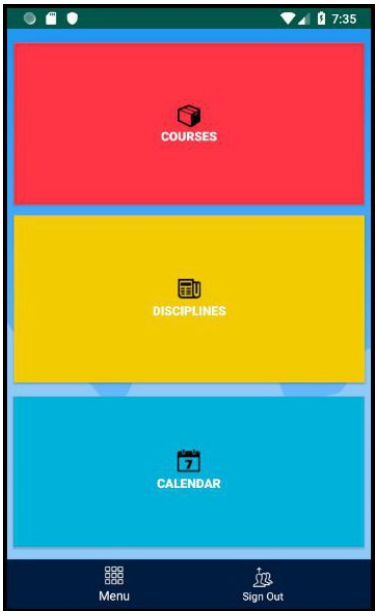
Em termos da arquitetura da aplicação, esta, após a entrada do ecrã inicial - Splash Screen - de login/registo, subdivide-se em três grandes áreas,com seguintes menus:

1 - Splash Screen de entrada:



2.Áreas:

2.1 - Courses
2.2 - Disciplines
2.3 - Calendar



É também mantido ao longo de todos os ecrãs da aplicação uma barra no footer que permite retroceder novamente para o Menu inicial (sem necessidade de utilizar um botão de retroceder) bem como um botão para fazer Sign Out da aplicação.

Nos sub-menus/ecrãs seguintes é possível adicionar/remover/editar os Cursos, adicionar/remover/editar as disciplinas associadas aos respectivos Cursos, adicionar/remover/editar as aulas associadas às respectivas disciplinas e finalmente visualizar/editar as aulas/eventos, em cada dias, no Calendário/agenda.

4) Modelos

Algumas das classes principais que são fundamentais para a organização e execução do ViewModel na aplicação:

- **MainActivity.java**

  Onde estão definidos as áreas de texto e os botões de login/registo com os respectivos Intents que permitem a sua correcta execução.

- **Menu.java**

  Permite a criação do menu e a sua consequente navegação, utilizando os três botões, ‘Courses’, ‘Disciplines’ e ‘Calendar’, para as respectivas activities.

- **CoursesActivity.java**

  Permite interacções e as operações CRUD para a àrea dos ‘Courses’.

- **DisciplinesActivity.java**

  Permite as interacções e as operações CRUD para a àrea das ‘Disciplines’.

- **MyCalendarActivity.java**
  Permite as interacções e as operações CRUD para a àrea do ‘Calendar’.

- **IUserApi.java**
  Onde estão definidos os pedidos REST que é possível ver no ponto 5 deste relatório.

5) Serviços - API

Construímos os nossos webservices utilizando o cliente RESTful para Android RETROFIT2 para os pedidos HTTP. Os principais pedidos feitos à base de dados que foram definidos em termos de API, nos respectivos endpoints, são os seguintes:

- Ao nível dos utilizadores:

```
@POST("user/createUser")
Call<ResponseBody> createUser(@Field("username") String title,
                              @Field("password") String body,
                              @Field("email") String email);

@POST("user/login")
Call<ResponseBody> login(@Query("email") String email,
@Query("password") String password);
```

- Ao nível dos cursos:

```
@GET("getCourses/courses")
Call<List<Courses>> getCoursesByIdUser(@Query("user_id")Long user_id);

@POST("getCourses/createCourse")
   Call<ResponseBody> createUser(@Field("user_id") long user_id,
                                 @Field("title") String title);

@GET("getCourses/courseByTitle")
        Call<List<Courses>> courseByTitle(@Query("user_id")Long
user_id,@Query("title")String title );

@DELETE("getCourses/deleteCourse/{course_id}")
Call<ResponseBody> deleteCourse(@Path("course_id") long course_id);
```

- Ao nível das disciplinas:

```
@GET("getSubjects/subjectByCourse")
Call<List<Subjects>> subjectByCourse(@Query("course_id")Long course_id);

@POST("getSubjects/createSubject")
Call<ResponseBody> createSubject(@Field("course_id") long user_id,
                                 @Field("title") String title);

@DELETE("getSubjects/deleteSubject/{id}")
Call<ResponseBody> deleteSubject(@Path("id") long id);

@GET("getSubjects/subjectByTitle")
Call<List<Subjects>> subjectByTitle(@Query("title")String title );
```

- Ao nível das aulas:

```
@POST("getClasses/createClass")
 Call<ResponseBody> createClass(@Field("subject_id") long subject_id,
                @Field("room_id") long room_id,
                @Field("date") String date,
                @Field("dateToComapare") String dateToCompare,
                @Field("title") int title);

@GET("getClasses/classesByDate")
Call<List<Classes>> getClassesByDate(@Query("date") String date,
@Query("user_id") Long user_id);

@GET("getClasses/classesByDateAndSubject")
Call<List<Classes>> getClassesByDateSubjectId(@Query("subject_id") Long
subject_id, @Query("date") String date, @Query("user_id") Long user_id);

@DELETE("getClasses/deleteClass/{id}")
Call<ResponseBody> deleteClass(@Path("id") long id);
```

- Ao nível das salas:

```
@GET("getRooms/allRooms")
Call<List<Rooms>> allRooms();
```

6) Login via Google Account - OAuth 2.0

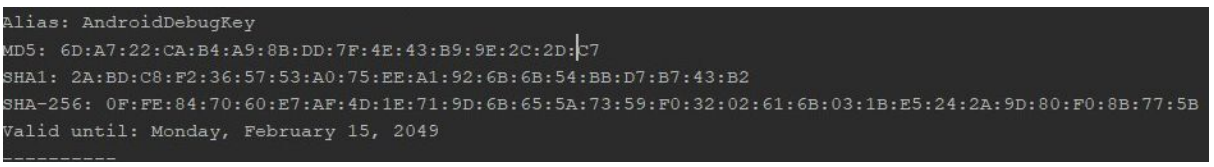
A Google possui um serviço para que as mais diversas aplicações android possam fazer uma autenticação, por forma a, muitas vezes, o utilizador poder guardar os dados da app na cloud da Google, sendo que este tipo de serviço é cada vez mais usado pelas grandes empresas online, evitando assim, por exemplo, perder um utilizador de uma aplicação, devido à barreira que possa existir no facto de o utilizador não que querer registar mais uma vez. A facilidade e rapidez que o utilizador poderá ter com este tipo de autenticação em muito superam a inconveniência de um novo preenchimento de um formulário de registo online.
Existe também a vantagem que este tipo de login irá permitir uma consistência de dados entre aparelhos que utilizem a mesma conta potenciando assim a experiência do utilizador.

Para suportar este tipo de serviço de autenticação segura a Google desenvolveu uma plataforma, *Google Identity Platform*, onde os developers podem gerir as keys necessárias para associação às apps que desejamos garantir este serviço. Este serviço assenta num protocolo standard da industria web chamado OAuth 2.0.

Decidimos então implementar a funcionalidade para o utilizador poder fazer login via conta Google (email/password) na nossa aplicação.

Como pré requisitos, no ficheiro build.gradle garantimos que o repositório da google - google() - estava incluído e posteriormente, já a nível da configuração da aplicação, adicionou-se a dependência/implementação:

Configuramos as credenciais da nossa aplicação gerando a nossa chave SHA-1 na consola e posteriormente, na página de developer autorizamos um novo Client ID, com a respectiva fingerprint:



De seguida, decidimos tomar como opção mostrar o botão de ‘Sign in with Google’ imediatamente por baixo, do botão de ‘Sign in’ normal da aplicação, por forma a que o utilizador possa ter acesso visual às duas opções e assim tomar a sua decisão.

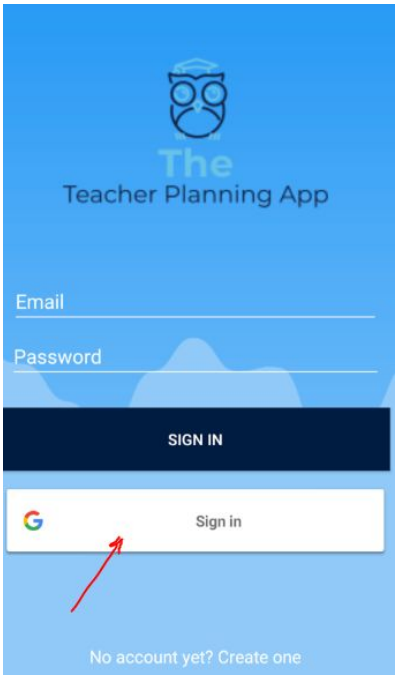
Adicionamos assim ao ficheiro ‘activity_login.xml’ o botão, com seguintes detalhes:

```
<Button
    android:id="@+id/login"
    android:layout_width="359dp"
    android:layout_height="58dp"
    android:layout_marginStart="160dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="163dp"
    android:layout_marginBottom="8dp"
    android:background="#0e1e40"
    android:text="Sign In"
    android:textColor="#fff"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.494"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.677" />

<com.google.android.gms.common.SignInButton
    android:id="@+id/signInButton"
    android:layout_width="356dp"
    android:layout_height="66dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.809" />
```

Por fim na classe Login.java, implementou-se o método de Callback Assíncrono para autenticação nos serviços do Google play.

Concluiu-se assim o layout do splash screen de entrada, ficando com a implementação do botão ‘sign in with google’, com o seguinte resultado final:



7) Conclusão

As principais conclusões retiradas dos desenvolvimento desta aplicação em particular e da apreensão dos conhecimentos de desenvolvimento de aplicações móveis na plataforma Android são as seguintes:

Importância das boas práticas em Android nomeadamente nos elementos das aplicações, as ‘Activities’ e os ‘Services’, a correcta gestão do ciclo de vida das actividades de uma aplicação, as tipologias de layout, os diferentes ficheiros na estrutura do projecto, as decisões de arquitetura em termos de persistência de dados, a facilidade de customização das aplicações e por fim a verdadeira abertura e gratuitidade da plataforma de desenvolvimento Android, com a sua enorme comunidade.

8) Bibliografia

- Beginner's Guide to Android App Development: A Practical Approach for Beginners, Serhan Yamacli, CreateSpace Independent Publishing Platform, 2017

- The Busy Coder’s Guide to Android Development, Mark L. Murphy, CommonsWare, LLC, 2014

- Android App Development, Hervé J. Franceschi, Jones & Bartlett Learning, 2017