



ACADEMIC PLANNER

Programação de Dispositivos Móveis

Mestrado em Engenharia Informática
Professor Nuno Oliveira

Breno Garcia 17499
Rodolfo Kobe 17648
Fernando Correia 11199

Table of Contents

1	INTRODUÇÃO E OBJETIVOS	2
2	ARQUITETURA E ESTRUTURA	2
2.1	ESQUEMA DE DADOS	3
2.2	STORAGE.....	4
2.3	AUTENTICAÇÃO	4
2.4	ESTRUTURA DO PROJETO NO ANDROID STUDIO.....	4
2.5	DECISÕES DE LAYOUT	4
2.5.1	<i>Telas da aplicação.....</i>	<i>5</i>
2.6	USO DE BIBLIOTECAS DE TERCEIROS	7
3	DIFICULDADES.....	8
4	EXTRAS	9
4.1	LOGIN COM GOOGLE	9
4.1.1	<i>Autenticação com o Firebase</i>	<i>9</i>
4.1.2	<i>Resultado</i>	<i>10</i>
4.2	DISPONIBILIZAÇÃO DA APLICAÇÃO ATRAVÉS DE UM LINK.....	11
4.2.1	<i>Escolha da logo da aplicação</i>	<i>11</i>
4.2.2	<i>Escolha do plano de fundo</i>	<i>11</i>
4.2.3	<i>Alteração na configuração do Android Manifest</i>	<i>11</i>
4.3	CRIANDO A APK.....	12
4.3.1	<i>Generate Signed APK.....</i>	<i>12</i>
4.4	DISPONIBILIZAÇÃO DA APLICAÇÃO ATRAVÉS DE UM LINK.....	13
5	CONCLUSÃO.....	13
6	BIBLIOGRAFIA	14

1 Introdução e Objetivos

Para a Unidade Curricular de Programação de Dispositivos Móveis e Multissensoriais do Mestrado em Engenharia Informática do Instituto Politécnico do Cavado e Ave foi proposta a elaboração de uma aplicação mobile que permita aos docentes planear o conteúdo dos respetivos cursos que lecionam e de cada turma dos cursos. A gestão de turmas e de alunos dessas turmas também é facilitado com uma aplicação mobile que permita a um docente ter uma visão diária e atualizada da informação do seu dia e da sua semana.

Para apoio ao docente, algumas ações são realizadas para que este possa ter a sua agenda cada vez mais independente do lugar onde esteja. Simplesmente com uma ligação à Internet de forma a poder organizar todo o seu dia. A contribuição neste trabalho é uma aplicação que permita ao professor conseguir visual anos letivos, verificar as faculdades em que esta a lecionar, os cursos e as turmas correspondentes bem como todos os seus alunos inscritos.

2 Arquitetura e Estrutura

Para arquitetura da aplicação, foi discutido a possibilidade da criação de uma API em Node.JS e que seria armazenada em um serviço web e usando o MondoDB para persistência dos dados. Esta aplicação seria consumida pela aplicação Android pelo serviço Retrofit. Porém foi proposto pelo Rodolfo que fosse utilizado o Firebase pela sua facilidade de integração com Android por ambos se tratarem de um serviço Google.

Portanto, utilizamos a ferramenta Firebase da Google para gerenciar os dados da aplicação, no âmbito de base de dados, autenticação de utilizadores e armazenamento de ficheiros.

O Firebase é uma ferramenta BaaS (Backend as a Service) que já prove uma quantidade de serviços que facilita a gestão da aplicação backend.

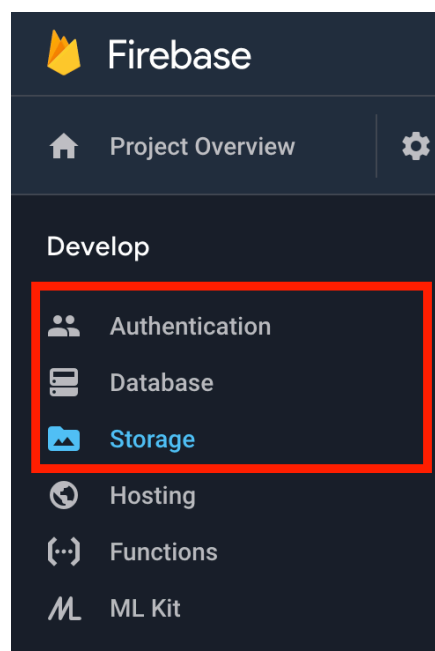


Figura 1 - Serviços utilizados no Firebase

2.1 Esquema de dados

Como padrão do Firebase, uma base de dados NoSQL, são criadas collections para estruturar os dados na base.

Na nossa aplicação, temos as seguintes *collections* para CRUD: adminpeople, classes, courses, disciplines, students, universities, users e years.

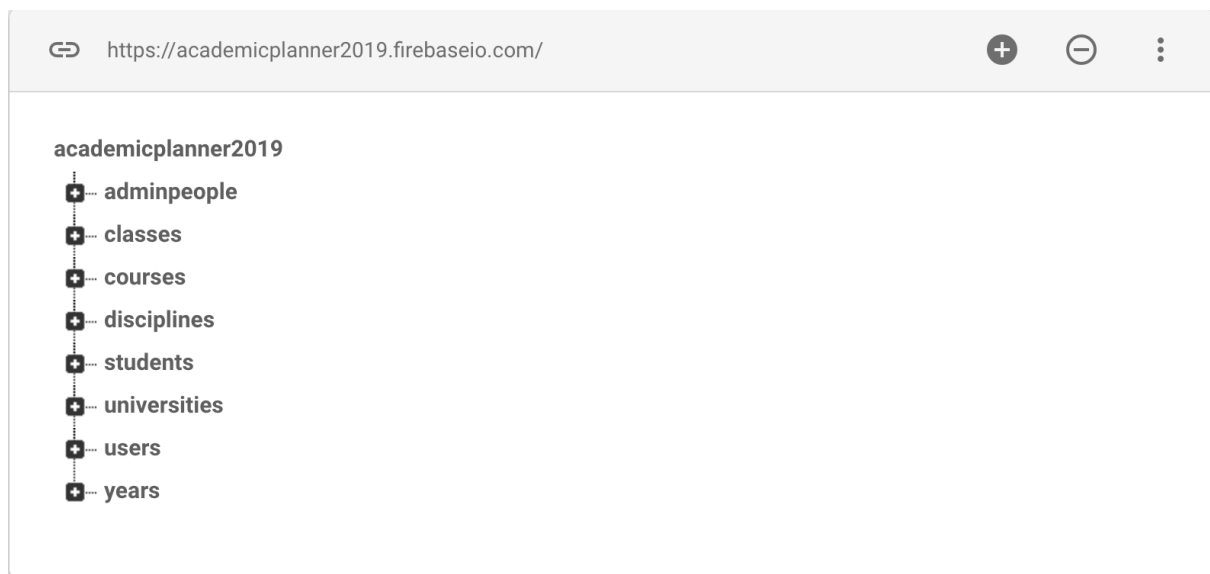


Figura 2 - Collections no Firebase

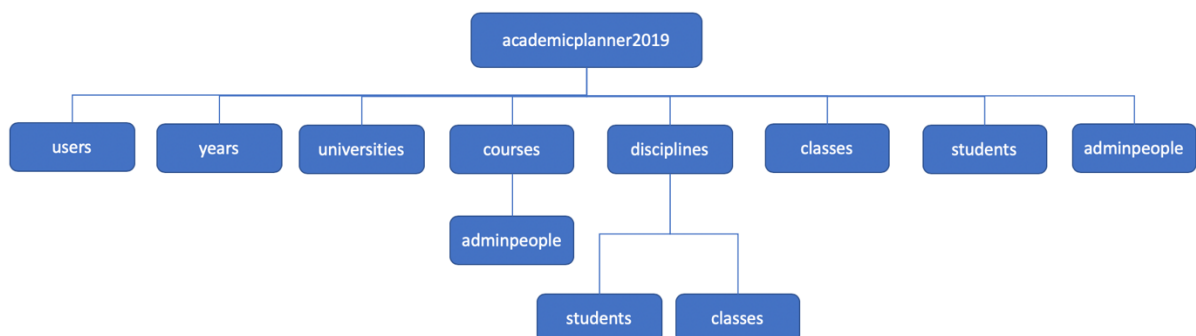


Figura 3 - Estrutura dos dados

As informações das aulas também são criadas dentro da *collection disciplinas* quando associadas à disciplina.

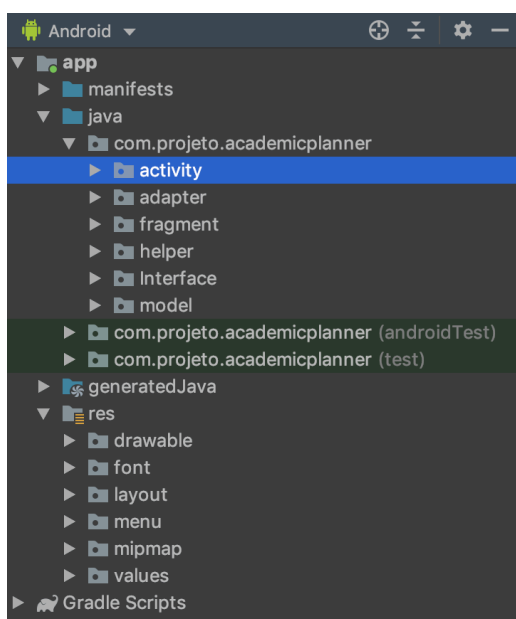
Para associarmos as pessoas do time administrativo, adicionamos um nó dentro da *collection* cursos. E é feito o mesmo no caso de estudantes com disciplinas, criamos um nó dentro da *collection* disciplina para que consigamos estruturar o envio de e-mails.

2.2 Storage

O *storage* do Firebase foi utilizado na aplicação para armazenar a foto do perfil de cada um dos utilizadores.

2.3 Autenticação

O Firebase facilita na implementação da autenticação para aplicação por integrar todos os serviços da plataforma Google.



2.4 Estrutura do projeto no Android Studio

Para o desenvolvimento da aplicação foi utilizado o Android Studio e o código foi desenvolvido na linguagem Java.

É possível verificar na imagem ao lado a estrutura do projeto dividida em camadas, conforme indicado nas boas práticas de desenvolvimento.

Abaixo um breve context do que há em cada uma das pastas.

Activity – uso para organização das activities.

Adapter – uso dos adapters para manutenção dos recylcers viewer.

Fragment – como activities, utilizados como uma

camada entre layout e código.

Helper – estrutura utilizada para gestão de ficheiros de configuração de acesso ao Firebase, autenticação do Google e JavaMail.

Interface – utilizadas para obrigar que sejam implementados métodos nas classes.

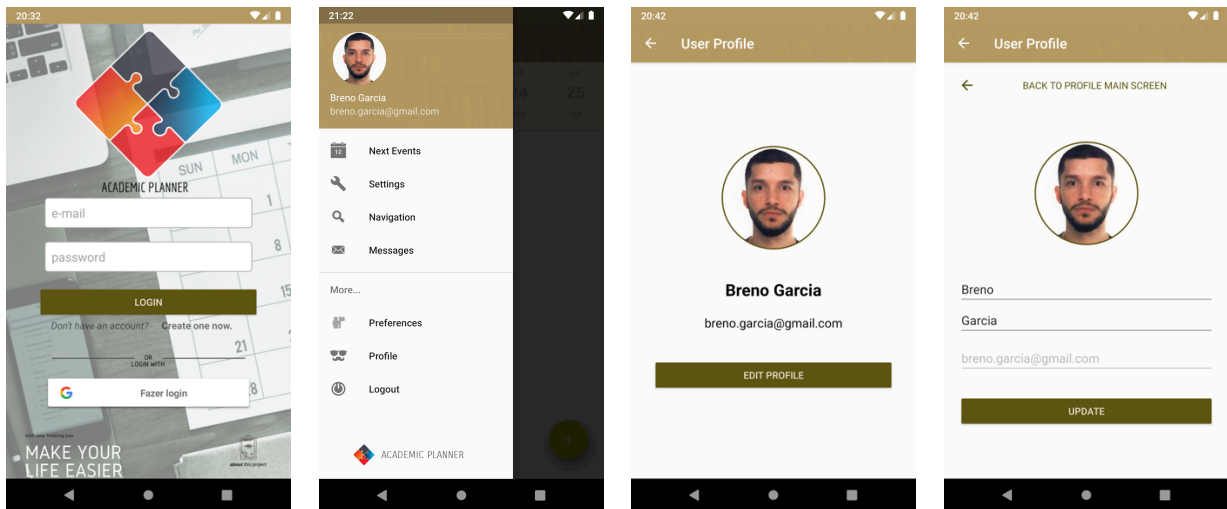
Model – são utilizados para geração dos *schemas* para o NoSQL.

2.5 Decisões de Layout

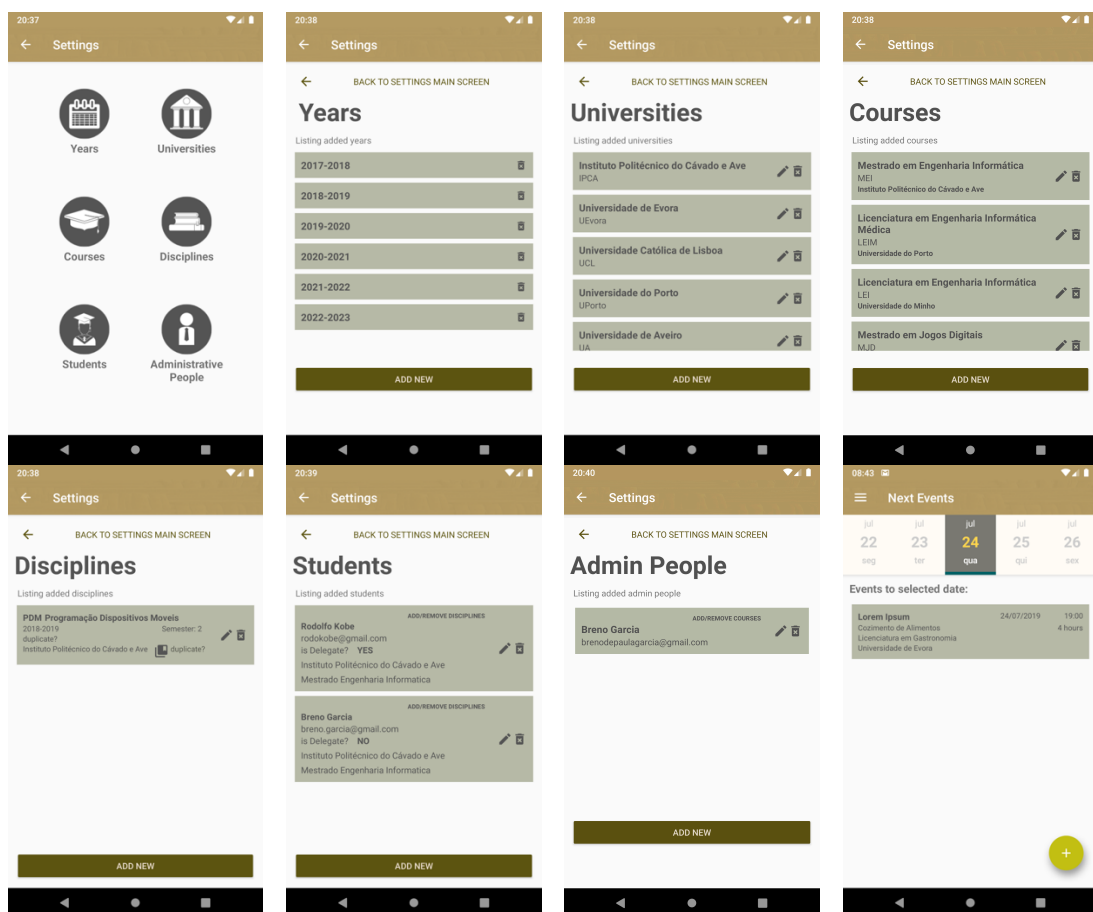
Este foi o ponto do trabalho que mais dificultou. A falta de mockups de base para criação da interfaces, foi um gargalo no desenvolvimento.

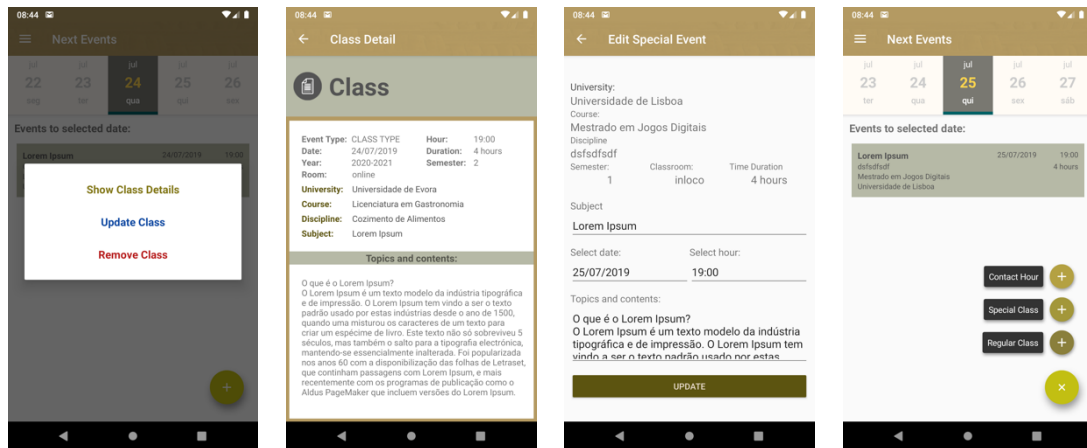
2.5.1 Telas da aplicação

2.5.1.1 Login, Menu de Navegação e Perfil do Utilizador



2.5.1.2 Configurações e navegação





2.5.1.3 Outras telas

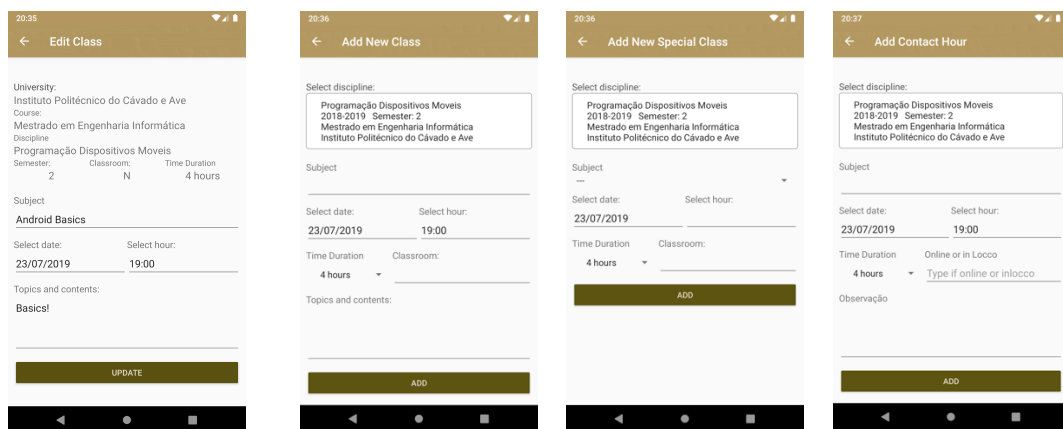


Figura 4 Exemplo editar aula e adicionar aula, aula especial e contato com professor

2.5.1.4 Envio de emails

Para envio de email utilizamos a biblioteca Javax Mail.

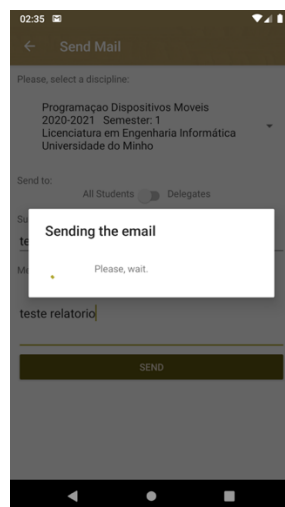


Figura 5 - Enviando email

2.5.1.4.1 Mensagens diretas aos alunos ou delegados:

No menu mensagens, é possível enviar mensagem para alunos ou delegados de uma disciplina. Ao enviar a mensagem é exibido uma AlertDialog com progressBar até finalizar o envio.

2.5.1.4.2 Envio de email ao realizar alteração de data ou hora de uma aula.

Foi implementado no arquivo ClassUpdateActivity.class e foi configurado uma mensagem padrão de troca de dia ou hora da aula.

O envio do email é executado em background e o email é enviado para estudantes associados à disciplina e para *admin people* associadas ao curso.

2.5.1.5 Duplicação de disciplina

Foi implementado e demonstrado ao professor na defesa a duplicação da disciplina e o sua execução em tempo real no Firebase.

2.6 Uso de bibliotecas de terceiros

Para o desenvolvimento utilizamos algumas bibliotecas para atender as necessidades da aplicação.

Abaixo a lista das bibliotecas de terceiros no âmbito de componentes de interface.

```
/**
 * searchable spinner
 */
implementation "com.toptoeche.searchables spinner:searchables spinnerlibrary:1.3.1"

/**
 * Circle ImageView used in profile
 */
implementation 'de.hdodenhof:circleimageview:3.0.0'

/**
 * Picasso library
 */
implementation 'com.squareup.picasso:picasso:2.71828'

/**
 * Calendar
 */
implementation 'devs.mulham.horizontalcalendar:horizontalcalendar:1.3.4'
implementation 'com.android.support:recyclerview-v7:28.0.0'

/**
 * N fabs
 */
implementation 'com.github.clans:fab:1.6.4'
```


3 Dificuldades

Implementação de Parcelable para passar dados da tela do calendário onde aparecem as aulas para as activities de visualização e edição de aulas.

Era de nosso conhecimento a necessidade do uso do putExtra para passar informação de uma intent para outra, mas neste caso era todos os dados de uma classe que deveria popular items em outra activity.

Implementamos Parcelable na model Classes, na NavMainActivity passamos os dados com o putExtra e nas outras activities inicializamos uma intent e nela usamos .getParcelableExtra e o parâmetro enviado na NavMainActivity.

4 Extras

4.1 Login com Google

Como um dos extras a serem implementados na aplicação, escolhemos o login com o Google pela facilidade de implementação utilizando o backend *Firebase*.

A autenticação através da conta da Google é realizada através do *Firebase* que já tem uma SDK própria que nos ajuda muito o trabalho e o desenvolvimento, tendo que apenas seguir a API utilizada na API. É necessário liberar no *Firebase* o tipo de autenticação.




Provider	Status
 Email/Password	Enabled
 Phone	Disabled
 Google	Enabled

Figura 6 Tipos de autenticação que liberamos no *Firebase*

Para implementação, precisamos adicionar duas dependências da biblioteca do *Firebase Authentication* para o Android e dos serviços do Google Play ao arquivo Gradle no nível da aplicação do módulo.

```
implementation 'com.google.firebase:firebase-auth:17.0.0'
implementation 'com.google.android.gms:play-services-auth:16.0.1'
```

4.1.1 Autenticação com o *Firebase*

Para integrar o Login do Google, configuramos o objeto *GoogleSignInOptions* e chamamos o *requestIdToken*.

```
/**
 *          GOOGLE          LOGIN          CONFIGURATION          BEGIN
 */
GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();

mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
mAuth = FirebaseAuth.getInstance();
```

Enviamos o ID do cliente para o método *requestIdToken*, conforme documentação OAuth 2.0.

Depois de integrar o Login do Google, assim ficou o código:

```
private void firebaseAuthWithGoogle(GoogleSignInAccount acct) {
    AuthCredential credential = GoogleAuthProvider.getCredential(acct.getIdToken(),
null);
    mAuth.signInWithCredential(credential)
```

```

        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's
                    information
                    toastMsgLong("Sign In with Google Success");
                    user = mAuth.getCurrentUser();
                    startActivity(new Intent(getApplicationContext(),
                    NavMainActivity.class));
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w(TAG, "signInWithCredential:failure",
                    task.getException());
                    toastMsgLong("Authentication Failed.");
                }
            }
        });
    }
}

```

4.1.2 Resultado

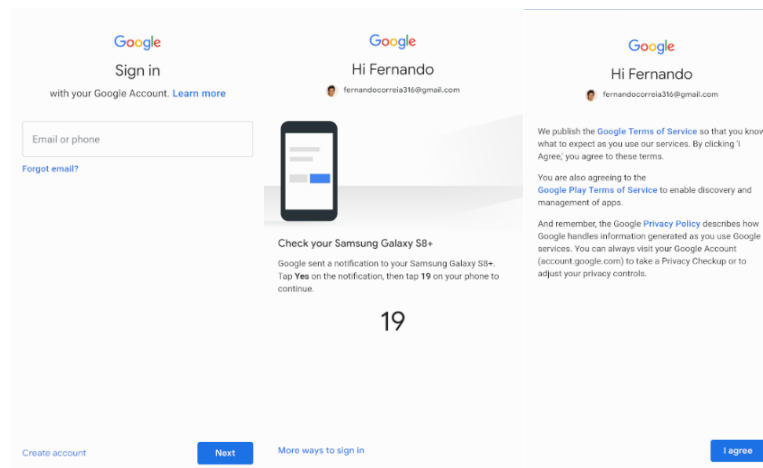


Figura 7 Login Google

As figuras acima representam o fluxo para autenticação do registo através de um email da Google utilizando a API do Firebase para ligação ao Gmail.

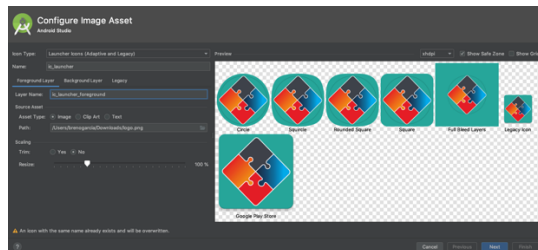
O utilizador clica no ícone do Gmail na tela do login já mostrado anteriormente, insere o email da Google e se não tiver email deve criar pois outro email não vai funcionar, e após isso simplesmente segue o fluxo sendo redirecionado para a aplicação com o seu login efetuado.

4.2 Disponibilização da aplicação através de um link

4.2.1 Escolha da logo da aplicação

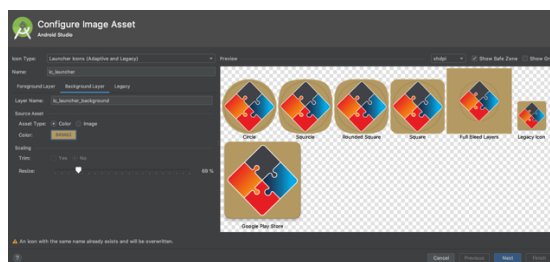
Mantivemos o tipo do ícone como *“Adaptive and Legacy”* para suportar versões anteriores e versões mais novas do Android.

Selecionamos a imagem da logo da aplicação e obtivemos o resultado da imagem abaixo:



4.2.2 Escolha do plano de fundo

Para o plano de fundo, acessamos a aba *“Background Layer”* e utilizamos em *Source Asset* a cor em *“Asset Type”*. Inserimos o hexadecimal `#b49a62` que corresponde à cor `colorPrimaryDark` da aplicação.



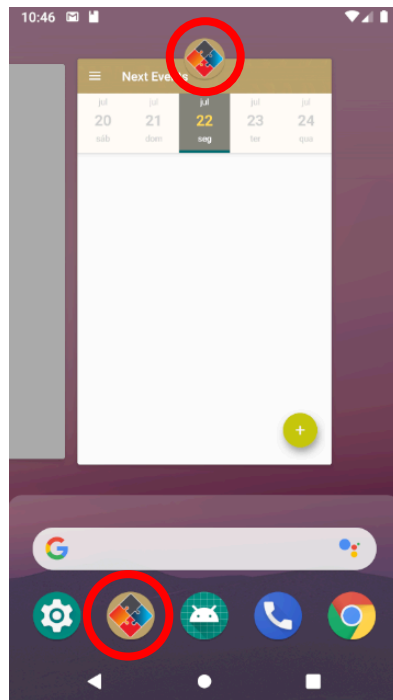
Posteriormente clicamos nos botões *Next* e *Finish*.

4.2.3 Alteração na configuração do Android Manifest

Para aplicação utilizar o ícone criado anteriormente, foi adicionada a entrada abaixo no `AndroidManifest.XML`.

```
android:icon="@mipmap/ic_launcher"
```

Para conferir o resultado, executamos a aplicação e conferimos como ficou.



4.3 Criando a APK

4.3.1 Generate Signed APK

Através do menu Build > Generate Signed Bundle / APK iniciamos a configuração para criação da APK.

Criamos uma Key Store conforme mostra a imagem abaixo:

Key store path:

Password: Confirm:

Key

Alias:

Password: Confirm:

Validity (years):

Certificate

First and Last Name:

Organizational Unit:

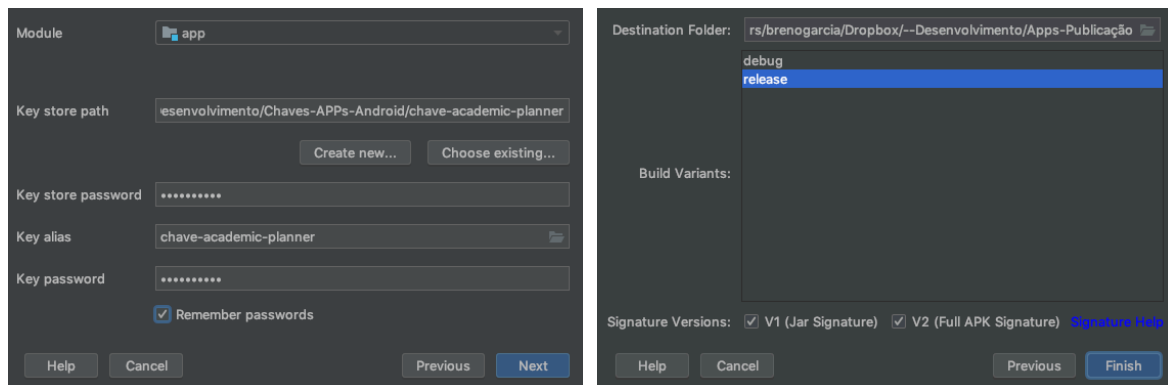
Organization:

City or Locality:

State or Province:

Country Code (XX):

E então selecionamos um local para salvar o apk, a forma de publicação como release e a versão de assinatura (ambas: V1(para funcionar em versões anteriores) e V2 (garante instalação mais rápida do app)).



Renomeamos o arquivo de app-release.apk para academic-planner.apk.

4.4 Disponibilização da aplicação através de um link

Escolhemos por disponibilizar a nossa aplicação através de um link. Para isto, copiamos o ficheiro da aplicação para o Google Drive e geramos um link de compartilhamento.

Link da APK da aplicação:

<https://drive.google.com/file/d/14RVGHXylB4RD97dX8jU0IVbpso63eSxO/view>

5 Conclusão

No geral, a aplicação **Academic Planner** gere bem as necessidades de que um docente possa ter no que toca a marcação de aula, marcação de horas de atendimento com alunos, verificação de salas disponíveis para determinada aula. Este auxílio é também suportado com autenticação através da conta da Google bem como disponibilização da aplicação para download através de um Link.

Com isto achamos que a aplicação cumpre bem o seu propósito e que um utilizador real, neste caso um docente, conseguiria gerir o seu trabalho minimamente com as mais recentes tecnologias.

Como trabalho futuro a equipa tem as sugestões de:

- Implementar um resumo seja em pdf, word ou excel com a lista de atividades que foram feitas no dia bem como tudo o que tem para fazer no resto da semana ou mês.
- Um chat para interagir com os seus alunos para esclarecimento de dúvidas e questões relacionados com horários.
- Autenticação do utilizador com conta de Facebook ou Twitter.
- Possibilidade de interação direta do professor com o seu próprio Twitter ou Facebook incorporado na aplicação tal como partilhar horários ou noticias para o seu Facebook pessoal ou por Tweet.

Outras...

6 Bibliografia

<https://developer.android.com/index.html>

<https://firebase.google.com/docs/auth/android/google-signin?hl=pt>

<https://firebase.google.com/docs/android/setup?authuser=0>

<https://www.udemy.com/course/curso-de-desenvolvimento-android-oreo/>

<https://spatidarblog.wordpress.com/2017/01/03/send-mail-internally-using-javamail-api/>