

# Programação de Dispositivos Móveis

---










Atributo	Valor
Unidade Curricular	Programação de Dispositivos Móveis
Curso	Mestrado em Engenharia Informática (MEI)
Instituição	Instituto Politécnico do Cávado e do Ave (IPCA)
Autoria	Bruno Rodrigues (a9969@alunos.ipca.pt), Daniel Martins (a17497@alunos.ipca.pt) Ricardo Dias (a11597@alunos.ipca.pt)
Data	2019-06-28

## Introdução e objetivos

---

O trabalho desenvolvido consiste na criação de uma aplicação móvel em android, e visa suportar o professor a organizar e gerir o ano lectivo, com visibilidade sobre as suas disciplinas, aulas, testes e ainda como forma de comunicação com os alunos.

Neste contexto, foram levantados os requisitos propostos pelo professor.

id	Requisitos	Classe	Estado
1	O professor deve poder entrar em sua sessão.	[P1]	
2	O professor deve ser capaz executar operações CRUD de anos, cursos, disciplinas, aulas, aulas especiais, eventos especiais.	[P1]	
3	O professor deve ser capaz de navegar nos anteriores anos, cursos, disciplinas e aulas.	[P1]	
4	O professor deve ser capaz de definir pessoas de contato com e-mails (todos os alunos ou pelo menos delegados).	[P1]	
5	Professor deve poder mudar de classe (sala ou datas / horas) e notificar (via e-mail) alunos e pessoas administrativas.	[P1]	
6	O professor deve poder copiar o conteúdo de uma disciplina para uma nova (independentemente do ano).	[P2]	
7	O professor deve poder exportar o planejamento de uma disciplina para um (ou todos) de: Excel + texto simples ou Excel + PDF.	[P2]	
8	O professor deve ser capaz de decidir se deseja ser notificado sobre aulas, eventos, reuniões de estudantes e afins.	[P2]	
9	Login com o Google.	[B1]	
10	Implementar uma ou várias estratégias de monetização.	[B2]	
11	Tornar a aplicação disponível no PlayStore ou através de um link	[B3]	
12	Crie um aplicativo simples para os alunos que interagem com o aplicativo de planejamento de professores para agendar compromissos de suporte.	[B4]	

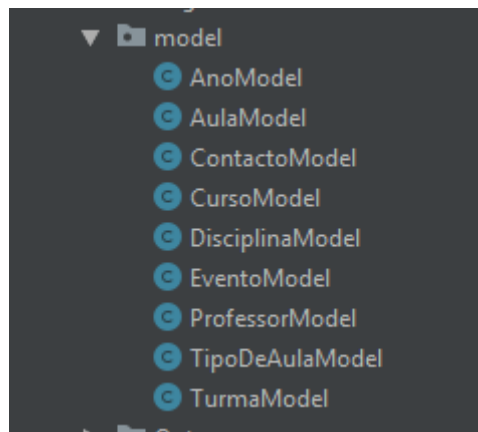
## Arquitetura

A base de dados foi definido utilizar o REALM , usa muito pouco espaço em disco e foi introduzido nas aulas, o schema foi definido para responder às exigências do programa.

Para o desenvolvimento foi utilizado o Android Studio.

## Modelo de dados

Neste ponto expõe se o modelo de dados:



### AnoModel

```
@PrimaryKeyprivate
int ID;
private String Descricao;
```

### Aula Model

```
@PrimaryKey
private Integer ID;
private Date DataDeOcorrencia;
private Integer Duracao;
private String Sala;
private Integer Tipo;
private String Sumario;
private Integer Turma;
private Integer DisciplinaId;
private boolean Important;
private int Year;
private int ProfId;
```

### CursoModel

```
private int ID;
private String Descricao;
private int Year;
private int ProfId;
```

### DisciplinaModel

```
@PrimaryKey
private Integer ID;
private String Nome;
private String Curso;
private Integer Anolectivo;
private String Acronimo;
private Integer Semestre;
// public String[] Principaistopicos = new String[20];
private RealmList<AulaModel> aulaModels;
private int Year;
private int ProfId;
```

## EventoModel

```
@PrimaryKeyprivate
    int ID;
    private String Descricao;
    private Date DataInicio;
    private int Duracao;
    private int Year;
    private int ProfId;
```

## ProfessorModel

```
@PrimaryKey
    private int ID;
    private String IdToken;
    private String PhotoUrl;
    private String Nome;
    private String Email;
    private RealmList<DisciplinaModel> disciplinaModels;
    private RealmList<ContactoModel> Contactos;
    private int Year;
    private int ProfId;
```

## TipodeAulaModel

```
@PrimaryKeyprivate
    int ID;
    private String Descricao;
    private int Year;
    private int ProfId;
```

## ContactoModel

```
@PrimaryKey
    private Integer ID;
    private String Nome;
    private String Email;
    private String Descricao;
    private int Year;
    private int ProfId;
```

## TurmaModel

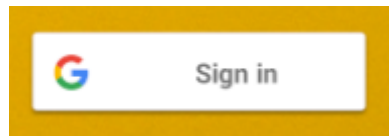
```
@PrimaryKey
    private Integer ID;
    private String Descricao;
    private RealmList<Integer> ListaContactos;
```

# Desenvolvimento

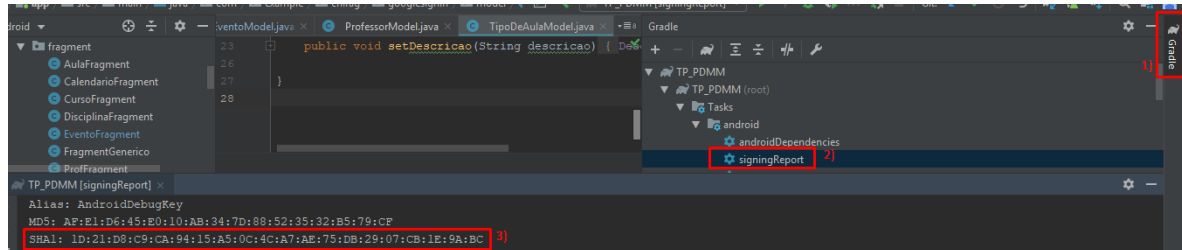
Nesta parte iremos explorar a parte de desenvolvimento do projeto, será dividido por partes com breve demonstração da método de implementação.

- Objectivo id 1

Para o Login o professor pode usar a conta do google,



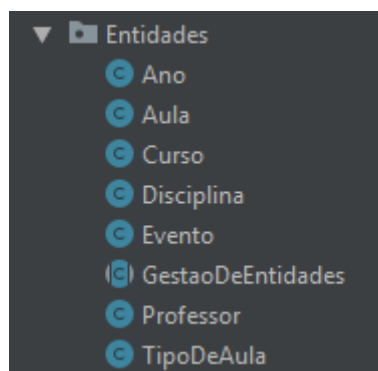
De forma a configurar o serviço de login é necessário configurar a aplicação a desenvolver no google developers, e registrar a chave SHA1 que pode ser obtida seguindo os seguintes passos.



Após login no caso do registo não estar gravado na DB é automaticamente registado a conta, evitando criação de conta.

- Objectivo id 2

As operações de CRUD foram organizadas na secção "Entidades", para cada um dos modelos e um Geral de gestão de entidades útil para o acesso mais flexível e genérico.



Ex: Entidade Ano

```
package com.example.chirag.googlelogin.Entidades;

import android.content.Context;
import android.util.Log;

import com.example.chirag.googlelogin.model.Anomodel;

import io.realm.Realm;
import io.realm.RealmQuery;
import io.realm.RealmResults;

public class Ano extends GestaoDeEntidades {
    public Anomodel entidade;
```

```

public Ano(Context context) {
    this(context, null);
}

public Ano(Context context, AnoModel model) {
    entidade = model == null ? new AnoModel() : model;
    super.context = context;
}

@Override
public RealmQuery<? extends AnoModel> BaseQuery(Realm realm) {
    return realm.where(AnoModel.class);
}

@Override
public void ExecuteCreateOrUpdate(Realm myRealm) {

    //Checks if the object already exists
    AnoModel findEntidade = BaseQuery(myRealm).equalTo("ID",
entidade.getID()).findFirst();

    if (findEntidade == null) {
        findEntidade = new AnoModel();
        findEntidade.setID(GetNextId(myRealm, AnoModel.class));
    }
    findEntidade.setDescricao(entidade.getDescricao());

    myRealm.insertOrUpdate(findEntidade);
}

@Override
public void ExecuteDelete(Realm realm) {
    RealmResults<? extends AnoModel> result = BaseQuery(realm).equalTo("ID",
entidade.getID()).findAll();

    if (result.size() == 0)
        Log.d("DataBase", "NO DATA FOUND TO DELETE");
    else
        result.deleteAllFromRealm();
}

@Override
public void ExecuteRead(Realm myRealm) {
    ExecuteRead(myRealm, null);
}

@Override
public void ExecuteRead(Realm myRealm, Integer ID) {
    setEntidade(BaseQuery(myRealm).equalTo("ID", ID == null ?
entidade.getID() : ID).findFirst());
}

private void setEntidade(AnoModel entidade) {
    this.entidade = entidade != null ? entidade : new AnoModel();
}
}

```

```

package com.example.chirag.googlelogin.Entidades;

import android.content.Context;
import android.util.Log;

import com.example.chirag.googlelogin.Outros.Enums;

import java.util.Collections;
import java.util.List;

import io.realm.Realm;
import io.realm.RealmConfiguration;
import io.realm.RealmObject;
import io.realm.RealmQuery;
import io.realm.exceptions.RealmMigrationNeededException;

public abstract class GestaoDeEntidades {
    private Realm realm;

    public Context context;

    public abstract void ExecuteCreateOrUpdate(Realm bgRealm);

    public abstract void ExecuteDelete(Realm realm);

    public abstract void ExecuteRead(Realm realm, Integer ID);

    public abstract void ExecuteRead(Realm realm);

    public abstract RealmQuery BaseQuery(Realm realm);

    public void CreateOrUpdate() {
        realm = getRealm();

        realm.executeTransaction(r -> {
            ExecuteCreateOrUpdate(realm);
        });
    }

    public void Delete() {
        Realm realm = getRealm();
        realm.executeTransaction(r -> {
            ExecuteDelete(realm);
        });
    }

    public void Read() {
        Realm realm = getRealm();
        realm.executeTransaction(r -> {
            ExecuteRead(realm);
        });
    }
}

```

```

    }

    public List<RealmObject> ReadAll(Class classToSearch) {
        try {
            realm = getRealm();

            return realm.where(classToSearch).findAll();
        } catch (Exception e) {
            Log.d("Erro on ID", "");
        } finally {
        }
        return Collections.emptyList();
    }

    public List<RealmObject> ReadAllByYear() {
        try {
            realm = getRealm();

            return BaseQuery(realm).findAll();
        } catch (Exception e) {
            Log.d("Erro on ID", "");
        } finally {
        }
        return Collections.emptyList();
    }

    }

    public Realm getRealm() {
        if (realm == null || realm.isClosed()) {
            try {
                realm = Realm.getDefaultInstance();
            } catch (IllegalStateException e) {
                if (e.getMessage().contains("Call `Realm.init(Context)` before creating a RealmConfiguration")) {
                    Realm.init(context);
                    realm = Realm.getDefaultInstance();
                } else {
                    throw e;
                }
            } catch (RealmMigrationNeededException r) {
                Realm.deleteRealm(getRealmConfiguration());
                realm = getRealm();
            }
        }

        return realm;
    }

    public static RealmConfiguration getRealmConfiguration() {
        return new RealmConfiguration.Builder().name("myrealm2.realm").build();
    }

    }

    Integer GetNextId(Realm realm, Class classToSearch) {
        Integer number = 0;
        try {
            number = realm.where(classToSearch).max("ID").intValue();
        } catch (Exception e) {
            Log.d("Erro on ID", "");
        }
    }

```



```

    } finally {
        return number == null ? 1 : ++number;
    }

}

public void Navegar(Enums.Navegar move, final int idRef) {
    Realm realm = getRealm();

    realm.executeTransaction(r -> {
        int idToSearch = 0;
        switch (move) {
            case Anterior: {
                idToSearch = idRef - 1;
                break;
            }
            case Seguinte: {
                idToSearch = idRef + 1;
                break;
            }
            case Este: {
                idToSearch = idRef;
                break;
            }
        }

        ExecuteRead(realm, idToSearch);
    });
}

public Context getContext() {
    return context;
}

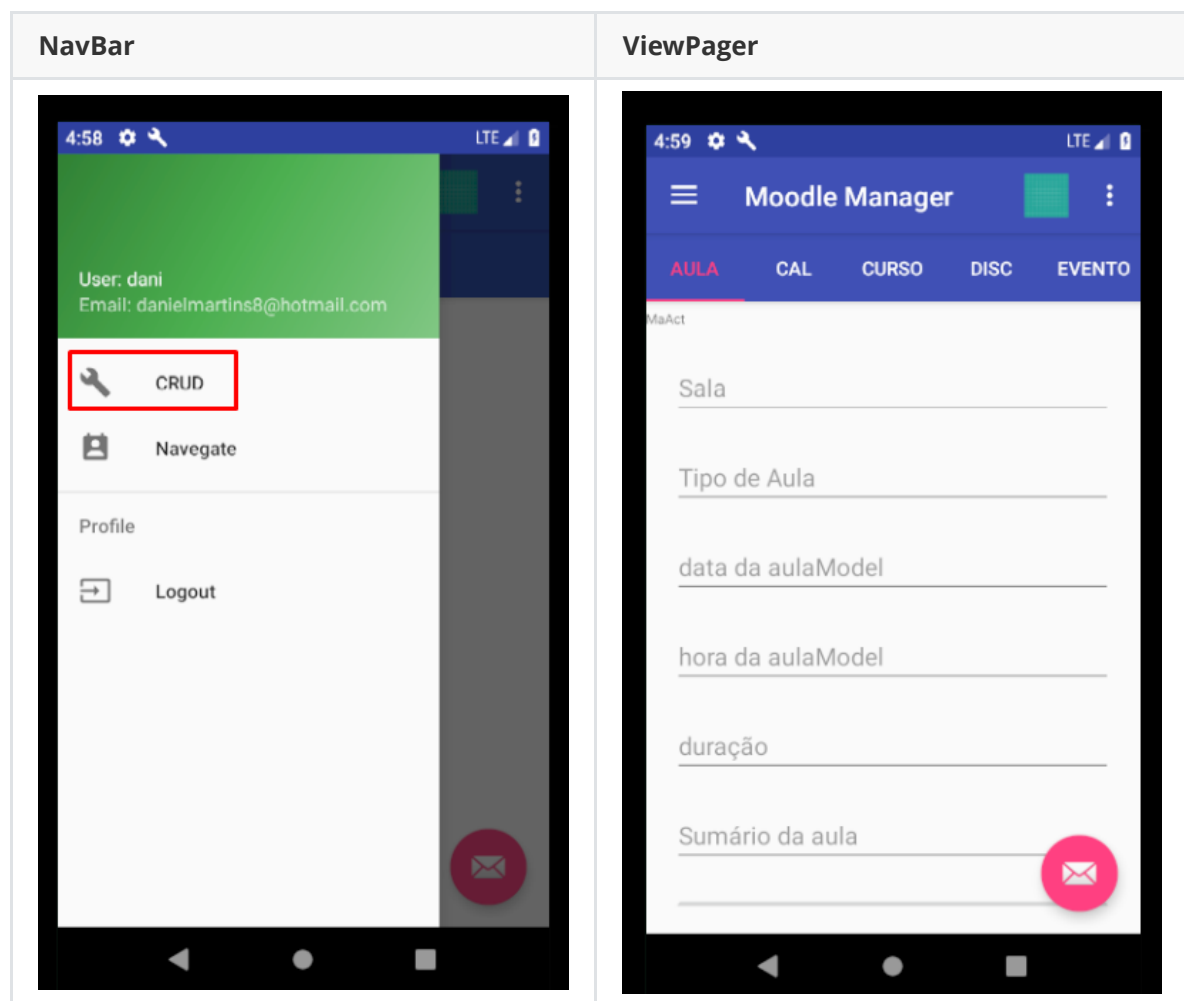
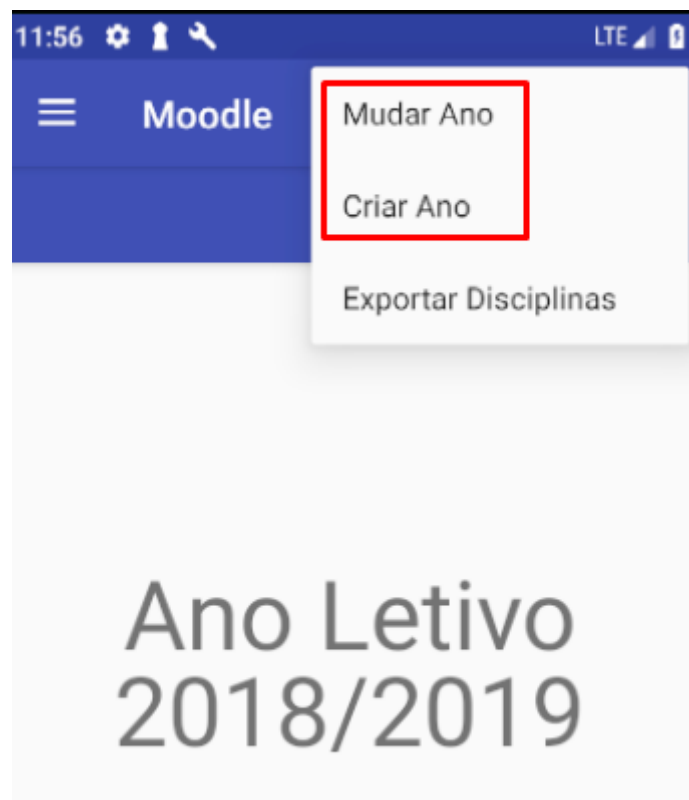
public void setContext(Context context) {
    this.context = context;
}
}

```

- Objectivo id 3

De forma a navegar e aceder às operações CRUD foi utilizado ViewPager com as diferentes Fragments de cada model com respetiva interação.

Mudar de ano:



ViewPagerAdapter

```
package com.example.chirag.googlelogin.adapters;

import android.os.Bundle;
import android.support.annotation.Nullable;
```

```

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

import com.example.chirag.googlelogin.fragment.AulaFragment;
import com.example.chirag.googlelogin.fragment.ContactoFragment;
import com.example.chirag.googlelogin.fragment.CursorFragment;
import com.example.chirag.googlelogin.fragment.DisciplinaFragment;
import com.example.chirag.googlelogin.fragment.EventoFragment;
import com.example.chirag.googlelogin.fragment.TipoDeAulaFragment;
import com.example.chirag.googlelogin.fragment.TurmasFragment;

import java.time.Year;

public class ViewPagerAdapter extends FragmentPagerAdapter {

    private Integer Year = 0;
    private Integer ProfId = 0;

    public ViewPagerAdapter(FragmentManager fm, int year, int profId) {
        super(fm);
        this.Year = year;
        this.ProfId = profId;
    }

    @Override
    public Fragment getItem(int position) {
        try {
            Bundle bundle = new Bundle();
            bundle.putInt("Year", this.Year);
            bundle.putInt("ProfId", this.ProfId);

            switch (position) {
                case 0:
                    AulaFragment aulaFragment = new AulaFragment();
                    aulaFragment.setArguments(bundle);
                    return aulaFragment;
                case 1:
                    CursorFragment cursoFragment = new CursorFragment();
                    cursoFragment.setArguments(bundle);
                    return cursoFragment;
                case 2:
                    DisciplinaFragment disciplinaFragment = new
DisciplinaFragment();
                    disciplinaFragment.setArguments(bundle);
                    return disciplinaFragment;
                case 3:
                    EventoFragment eventoFragment = new EventoFragment();
                    eventoFragment.setArguments(bundle);
                    return eventoFragment;
                case 4:
                    TurmasFragment turmasFragment = new TurmasFragment();
                    turmasFragment.setArguments(bundle);
                    return turmasFragment;
                case 5:
                    TipoDeAulaFragment tipoDeAulaFragment = new
TipoDeAulaFragment();
                    tipoDeAulaFragment.setArguments(bundle);

```

```

        return tipoDeAulaFragment;
    case 6:
        ContactoFragment contactoFragment = new ContactoFragment();
        contactoFragment.setArguments(bundle);
        return contactoFragment;

    }

} catch (Exception e) {
    e.printStackTrace();
}
return null;
}

/**
 * number of fragments
 * @return
 */
@Override
public int getCount() {
    return 7;
}

@Nullable
@Override
public CharSequence getPageTitle(int position) {

    String nome = null;

    switch (position) {
        case 0:
            nome = "Aula";
            return nome;
        case 1:
            nome = "Curso";
            return nome;
        case 2:
            nome = "Disc";
            return nome;
        case 3:
            nome = "Evento";
            return nome;
        case 4:
            nome = "Turma";
            return nome;
        case 5:
            nome = "TipoAula";
            return nome;
        case 6:
            nome = "Contacto";
            return nome;
    }
    return null;
}
}

```

- Objectivo id 4

O envio de email é realizado através do icon desenvolvido na class email.

```
public class Email {

    private String[] Para;
    private String Assunto;
    private String Mensagem;

    public Email(String[] para, String assunto, String mensagem) {
        setPara(para);
        setAssunto(assunto);
        setMensagem(mensagem);
    }
    public Email(){

    }

    public String[] getPara() {
        return Para;
    }

    public void setPara(String[] para) {
        Para = para;
    }

    private String getAllPara() {
        return TextUtils.join(";", getPara());
    }

    public String getAssunto() {
        return Assunto;
    }

    public void setAssunto(String assunto) {
        Assunto = assunto;
    }

    public String getMensagem() {
        return Mensagem;
    }

    public void setMensagem(String mensagem) {
        Mensagem = mensagem;
    }

    private Uri setEmail() {
        return Uri.parse("mailto:?subject=" + getAssunto() + "&body=" +
getMensagem() + "&to=" + getPara());
    }

    public void SendEmail(Context context) {
        Intent mailIntent = new Intent(Intent.ACTION_VIEW);
        mailIntent.setData(SetEmail());
    }
}
```

```

        context.startActivity(Intent.createChooser(mailIntent, "Send mail..."));
    }
}

```

Objectivo id 7

A extração do ficheiro para análise, está na classe excel.

```

public class Excel {

    public static boolean ExportDisciplinaModel(List<RealmObject> disciplinas,
Context context, String fileName, String extension) {
        String newLine = System.getProperty("line.separator");

        if (!extension.equals("csv") && !extension.equals("txt"))
            throw new IllegalArgumentException("File extension must be csv or
txt.");

        if (disciplinas.size() == 0) {
            Toast.makeText(context, "Não foi selecionada nenhuma disciplina para
exportar para "+extension+ ".", Toast.LENGTH_SHORT).show(); //Show shadow text
            return false;
        }

        if (!Useful.iswriteStoragePermissionGranted(context))
            return false;

        StringBuilder txt = new StringBuilder();

        txt.append("Prof,Year,Id,Nome,Curso,Acronimo,Semestre" + newLine);
        for (RealmObject c : disciplinas) {
            DisciplinaModel disc =
((com_example_chirag_googlesignin_model_DisciplinaModelRealmProxy) c);
            txt.append(disc.getProfId() + "," + disc.getYear() + "," +
disc.getID() + "," + disc.getNome() + "," + disc.getCurso() + "," +
disc.getAcronimo() + "," + disc.getSemestre() + "," + newLine);

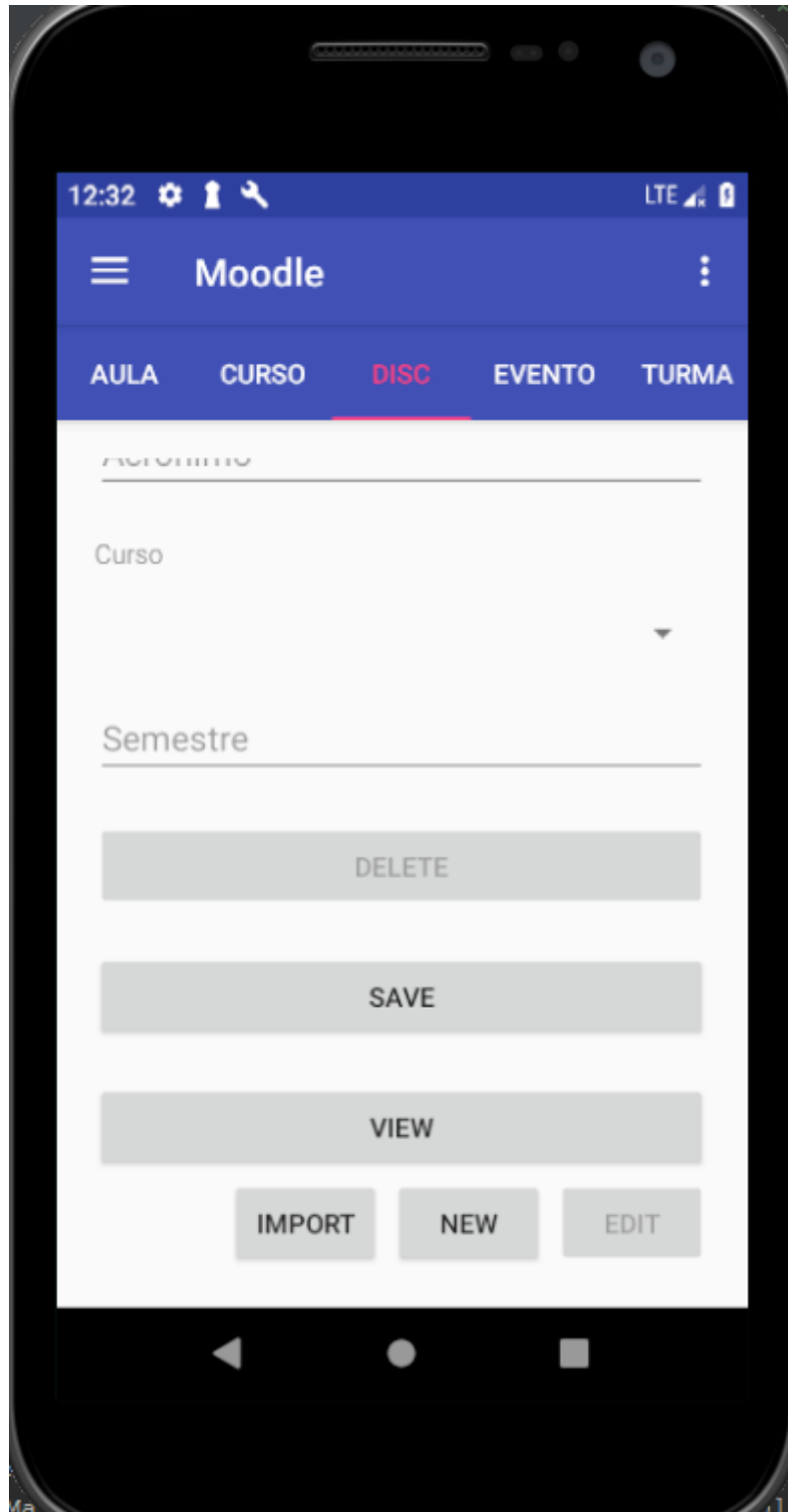
        }

        if (Useful.CreatFile(context, fileName, extension,
txt.toString().getBytes())) {
            Toast.makeText(context, "Ficheiro exportado!",
Toast.LENGTH_SHORT).show();
            return true;
        }
        return false;
    }
}

```

## Maquetes

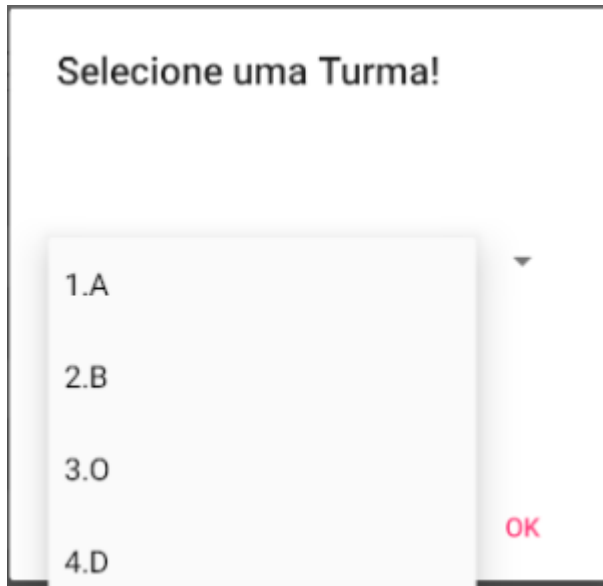
Nesta secção iremos demonstrar os exemplos finais dos dashboards de navegação.



Para a navegação foram criados os botões com determinadas funções:

- **Delete** - Apaga o registo da BD.
- **Save** - Grava o registo na BD.
- **View** - Consulta a BD e mostra os elementos criados.

Ex:



- **Import** - Permite selecionar elementos criados em anos transatos de modo a importar e copiar o mesmo registo.
- **New** - Para limpar os campos e criar de forma a criar novo registo.
- **Edit** - Após selecionar registo da BD permite a sua alteração.

## Conclusões e trabalho futuro

---

Após a conclusão deste trabalho foi possível concluir a grande evolução de todos os membros do grupo, totalmente inexperientes em programação android.

A nível das dificuldades que encontramos, inicialmente utilização do git gerou alguns problemas, após atualizações, muito devido às configurações do gradle.

A tentativa de utilizar a agenda e apresentar o calendário com marcações de aulas foi um desafio, após algumas tentativas sem sucesso, decidiu-se alterar a estratégia de forma a cumprir com os requisitos propostos.

Este projeto permitiu cada membro do grupo ultrapassar o as suas próprias limitações e munir cada um de mais uma valência para o mercado de trabalho o que pode resultar em futuras oportunidades.

## Referências

---

- <https://developer.android.com/studio/intro>
- <https://developer.android.com/guide/topics/ui/layout/recyclerview>
- <https://realm.io/blog/realm-for-android/>