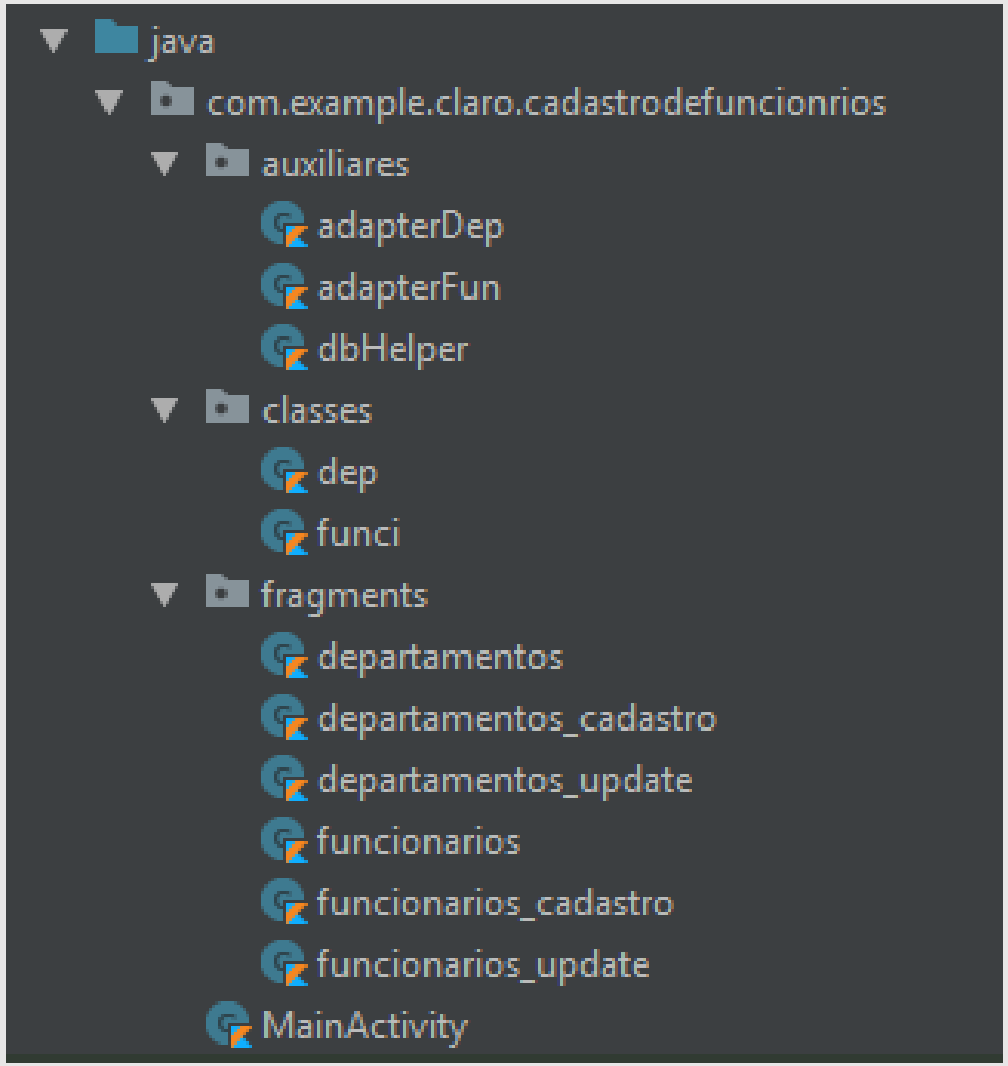
Cadastro de Funcionários

Estrutura do Projeto

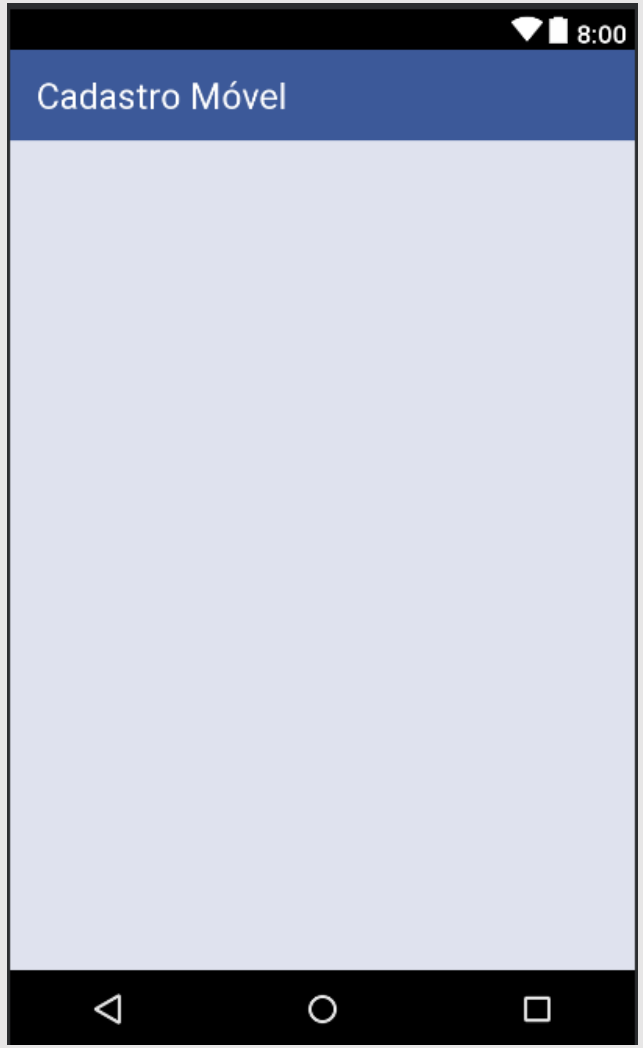


- 1 Activity
  - Main Activity
- 2 classes
  - Funcionários
  - Departamentos
- 6 Fragments
  - Departamentos
  - Cadastro Departamentos
  - Atualização Departamentos
  - Funcionários
  - Cadastro de Funcionários
  - Atualização de Funcionários
- 3 Auxiliares
  - Adaptador Lista Departamento
  - Adaptador Lista Funcionários
  - Banco de Dados

Cadastro de Funcionários

Main Activity


```
override fun onCreate(savedInstanceState: Bundle?)
{

    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    inicio()

}

fun inicio()
{
    supportFragmentManager.beginTransaction().replace(R.id.container,
        departamentos()
    ).commit()
}
```



Cadastro de Funcionários

Main Activity


```
class Compartilhado : ViewModel()
{
    val selecionado = MutableLiveData<dep>()
    val funSelecionado = MutableLiveData<funci>()

    fun select(item: dep)
    {
        selecionado.value = item
    }

    fun selectFun(item: funci)
    {
        funSelecionado.value = item
    }

}
```

View Model tem como objetivo armazenar dados gerados por uma activity/fragment mesmo depois do seu fechamento

Cadastro de Funcionários

Classes

Departamentos:

```
data class dep(var nome:String, var img:Int, var sigla : String, var id : Int)
```

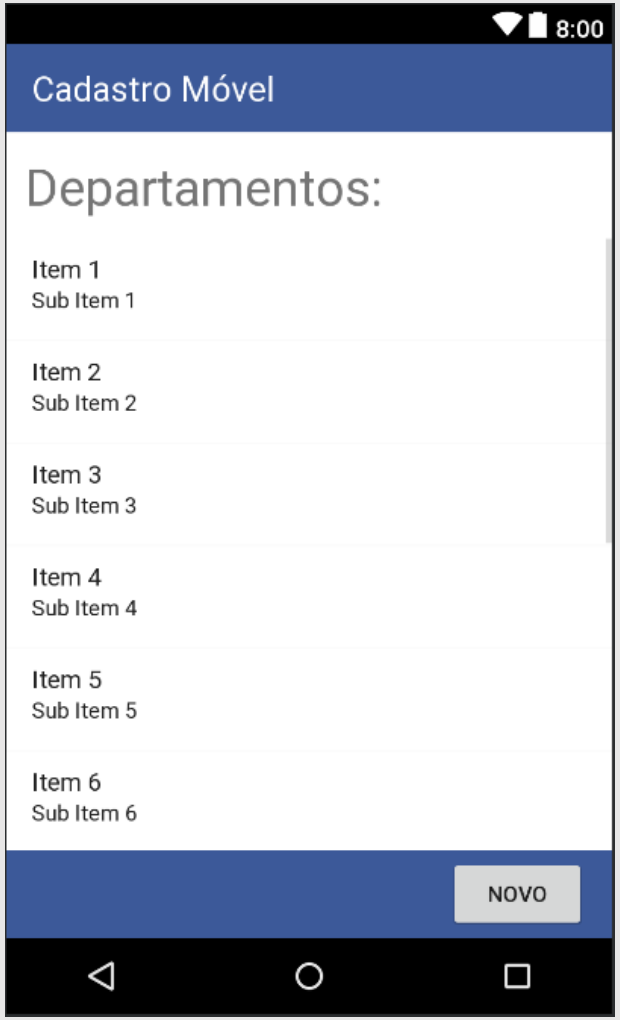
Funcionários:

```
data class funci(var nome : String, var id : Int, var rg : String, var idDpto : Int, var img : Int)
```

Cadastro de Funcionários

Fragmento - Departamentos


```
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View?
{

    super.onCreate(savedInstanceState)
    model = activity?.run {ViewModelProviders.of( activity: this).get(MainActivity.Compartilhado::class.java)
    } ?: throw Exception("Invalid Activity")

    val fgView = inflater.inflate(R.layout.departamentos, container,  attachToRoot: false)

    val listView = fgView.lista_dep as ListView

    var nomes: ArrayList<dep> = ArrayList()

    db = dbHelper(activity!!.applicationContext)

    nomes = db.listarDpto()
```



Cadastro de Funcionários

Fragmento - Departamentos

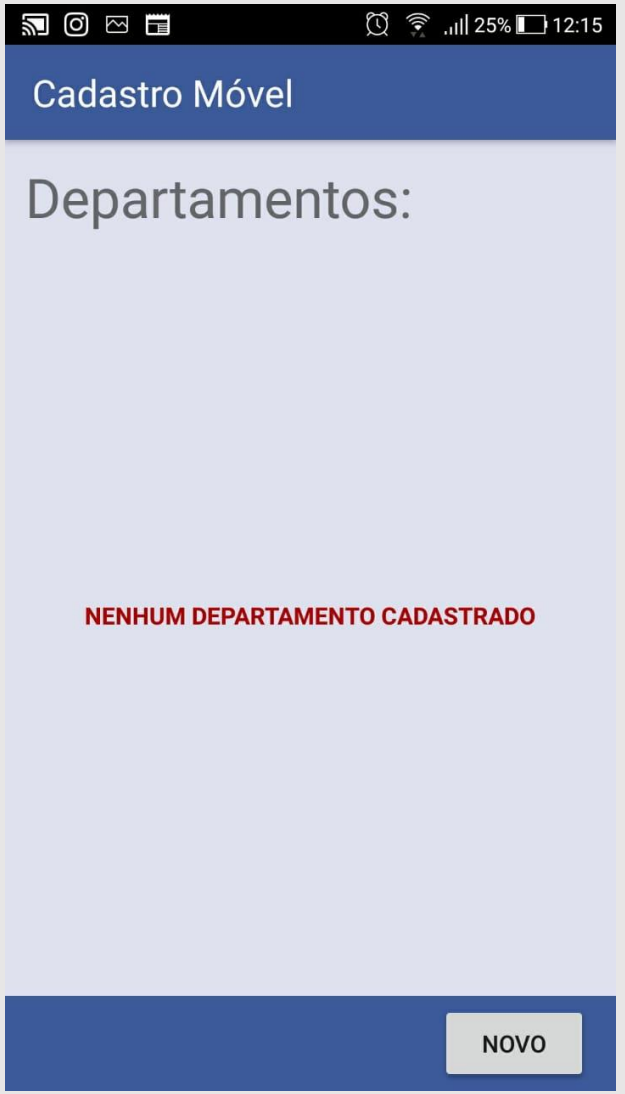

```
if (nomes.count() > 0) {
    listView.adapter = adapterDep(
        activity!!.applicationContext,
        nomes,  mcontext: this
    )


    fgView.lista_dep.setOnItemClickListener() { adapterView, view, i, l ->

        val dptoAtual: dep = nomes[i]

        model.select(dptoAtual)

        fragmentManager?.beginTransaction()?.replace(
            R.id.container,
            funcionarios()
        )?.remove(this)?.commit()
    }

}
```

Cadastro de Funcionários

Fragmento - Departamentos


```
else
{
    fgView.txt_sem_cad.visibility = View.VISIBLE
    listView.adapter = adapterDep(
        activity!!.applicationContext,
        nomes, mcontext: this
    )
}
```

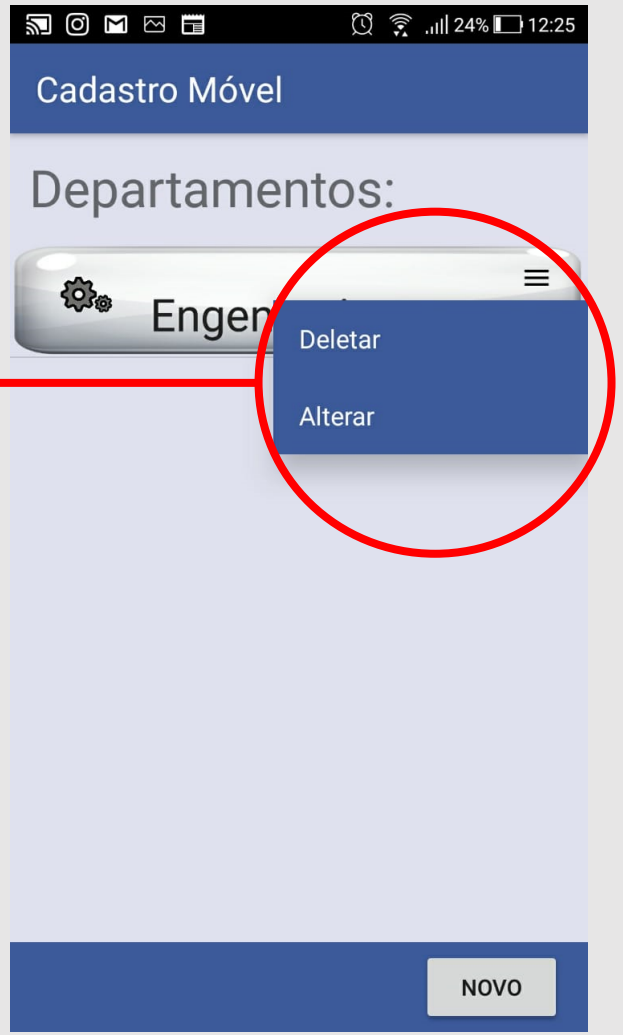
```
fgView.bt_nv_dep.setOnClickListener { view ->

  fragmentManager?.beginTransaction()?.replace(
        R.id.container,
        departamentos_cadastro()
   )?.remove(this)?.commit()
}
```



Cadastro de Funcionários

Fragmento - Departamentos - menuDep


```
fun menuDep (dep : dep, v: View)
{

    val opcoes = PopupMenu(ContextThemeWrapper(context?.applicationContext, R.style.popup), v)

    val dptoAtual : dep = dep

    model.select(dptoAtual)
```



Cadastro de Funcionários

Fragmento - Departamentos - menuDep


```
opcoes.setOnMenuItemClickListener { item ->
    when (item.itemId)
    {
        R.id.menu_deletar ->
        {

            val builder = AlertDialog.Builder(activity as Context)

            builder.setTitle("ATENÇÃO: ")

            builder.setMessage("Deseja mesmo deletar ${dep.nome}?")

            builder.setPositiveButton( text: "SIM") { dialog, which ->

                Log.d( tag: "CURA",  msg: "ALERTA : SIM")

                db.deletarDpto(dep.component4())
                val dptoAtual : Int = model.selecionado.value!!.component4()

                db.deletarFun(dptoAtual)

                fragmentManager?.beginTransaction()?.replace(R.id.container,
                    departamentos()
                )?.remove(this)?.commit()
```

Cadastro de Funcionários

Fragmento - Cadastro Departamento


```
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {

    val fgView = inflater.inflate(R.layout.departamentos_cadastro, container,  attachToRoot: false)

    db = dbHelper(activity!!.applicationContext)

    val imgs : IntArray = intArrayOf (R.drawable.d1, R.drawable.d2, R.drawable.d3, R.drawable.d4, R.drawable.d5)

    var i = 0
```

```
fgView.bt_cad_img_ant.setOnClickListener { view ->

    i--

    if(i == -1) i = 4

    img_dpto.setImageResource(imgs[i])

}
```

```
fgView.bt_cad_voltar.setOnClickListener { view ->

    fragmentManager?.beginTransaction()?.replace(
        R.id.container,
        departamentos()
    )?.remove(this)?.commit()

}
```

Cadastro de Funcionários

Fragmento - Cadastro Departamento


```
fgView.bt_cad_inserir.setOnClickListener { view ->


    var nome : String = nome_dpto.text.toString()
    var sigla : String = sigla_dpto.text.toString()
    var img = imgs[i]
    val regex = Regex( pattern: ".*\\d+.*")

    if (nome.matches(regex) || nome == "" || sigla == "")
    {
        Toast.makeText(activity, text: "Dados Inseridos Inválidos", Toast.LENGTH_SHORT).show()
    }

    else
    {
        db.inserirDpto(nome, sigla, img)
        fragmentManager?.beginTransaction()?.replace(R.id.container,
            departamentos()
        )?.remove(this)?.commit()
    }

    fgView.hideKeyboard()
}
```

Cadastro de Funcionários

Fragmento - Atualização Departamento

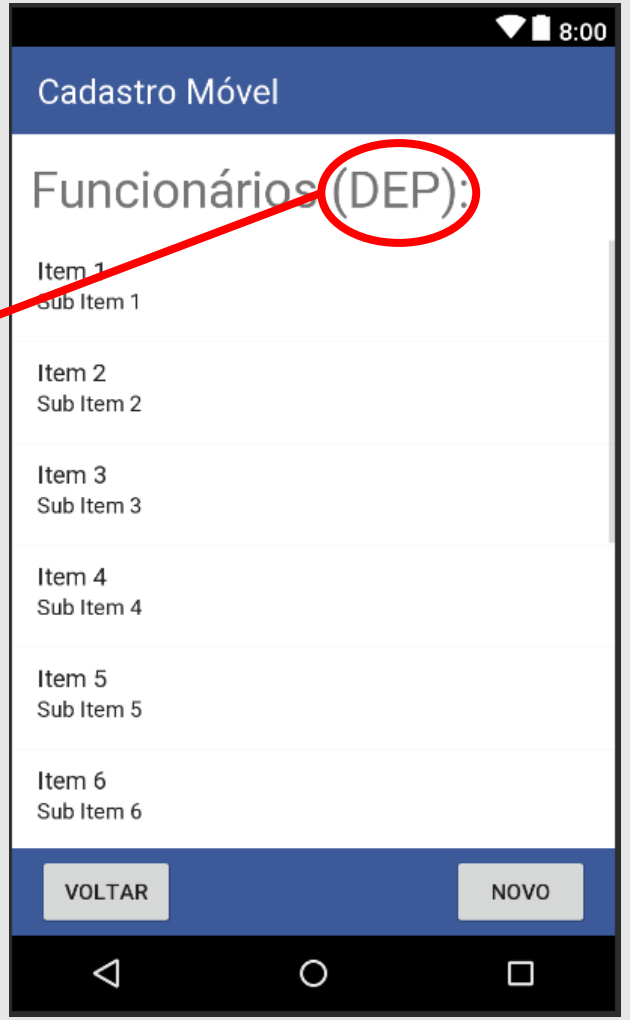

```
val dptoAtual = model.selecionado.value


fgView.nome_dpto_up.setText(dptoAtual?.nome)
fgView.sigla_dpto_up.setText(dptoAtual?.sigla)
fgView.img_dpto_up.setImageResource(dptoAtual!!.img)

when (dptoAtual!!.img)
{
    R.drawable.d1 -> i = 0
    R.drawable.d2 -> i = 1
    R.drawable.d3 -> i = 2
    R.drawable.d4 -> i = 3
    R.drawable.d5 -> i = 4
}
```

```
else
{
    db.atualizarDep(dptoAtual.id, name, sigla, img)
    fragmentManager?.beginTransaction()?.replace(R.id.container,
        departamentos()
    )?.remove(this)?.commit()
}
```

Cadastro de Funcionários

Fragmento - Funcionários


```
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {

    super.onCreate(savedInstanceState)
    model = activity?.run { this: FragmentActivity
        ViewModelProviders.of( activity: this).get(MainActivity.Compartilhado::class.java)
    } ?: throw Exception("Invalid Activity")

    val fgView = inflater.inflate(R.layout.funcionarios, container,  attachToRoot: false)

    val listView = fgView.lista_fun as ListView

    db = dbHelper(activity!!.applicationContext)

    val dptoAtual : dep = model.selecionado.value as dep

    fgView.titulo_fun.setText("Funcionarios (${dptoAtual.component3()}):")

    nomesFunc = db.listarFun(dptoAtual.component4())
```



Cadastro de Funcionários

Fragmento - Funcionários


```
if (nomesFunc.count() > 0)
{
    listView.adapter = adapterFun(activity!!.applicationContext,nomesFunc,  mcontext: this)
}
```

```
else
{
    fgView.txt_sem_cad_fun.visibility = View.VISIBLE
}
```

```
fgView.bt_lista_func_voltar.setOnClickListener { view ->

    fragmentManager?.beginTransaction()?.replace(
        R.id.container,
        departamentos()
    )?.remove(this)?.commit()

}
```

```
fgView.bt_nv_fun.setOnClickListener { view ->

    fragmentManager?.beginTransaction()?.replace(
        R.id.container,
        funcionarios_cadastro()
    )?.remove(this)?.commit()

}
```

Cadastro de Funcionários

Fragmento - Funcionários


```
fun apagar(funci: funci)
{
    db.deletarFunId(funci.component2())

    fragmentManager?.beginTransaction()?.replace(R.id.container,
        funcionarios()
    )?.remove(this)?.commit()
}
```

```
fun modificar(funci: funci)
{
    model.selectFun(funci)

    fragmentManager?.beginTransaction()?.replace(R.id.container,
        funcionarios_update()
    )?.remove(this)?.commit()
}
```

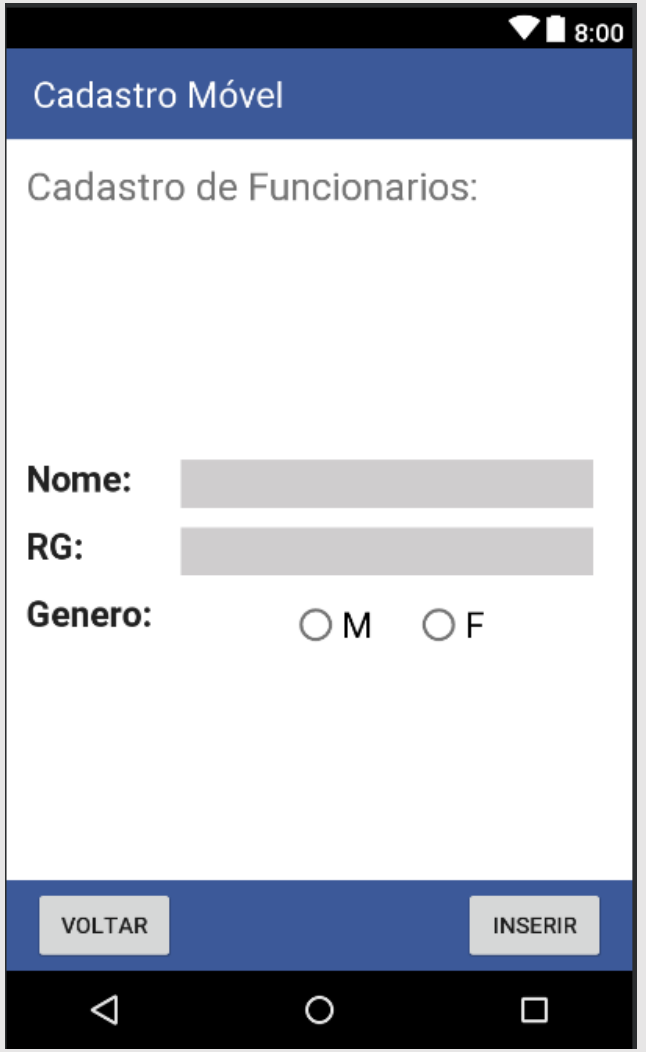
```
fun alert(funci: funci)
{
    Log.d( tag: "CURA",  msg: "ALERTA")

    val builder = AlertDialog.Builder(activity as Context)
```

Cadastro de Funcionários

Fragmento - Funcionários Cadastro


```
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {

    super.onCreate(savedInstanceState)
    model = activity?.run { this: FragmentActivity
        ViewModelProviders.of( activity: this).get(MainActivity.Compartilhado::class.java)
    } ?: throw Exception("Invalid Activity")

    val fgView = inflater.inflate(R.layout.funcionarios_cadastro, container,  attachToRoot: false)

    db = dbHelper(activity!!.applicationContext)

    val dptoAtual : Int = model.selecionado.value!!.component4()
```

```
fgView.bt_fun_inserir.setOnClickListener { view ->

    val nome : String = func_nome.text.toString()
    val rg : String = func_rg.text.toString()

    if (fgView.feminino.isChecked) img = R.drawable.female
    else if (fgView.masculino.isChecked) img = R.drawable.male
```



Cadastro de Funcionários

Fragmento - Funcionários Cadastro


```
var nomesFunc: ArrayList<funci> = ArrayList()

nomesFunc = db.listarFun( id: -1)

var duplicado : Int = 0

for (i in 0 .. nomesFunc.count() - 1)
{
    if (nomesFunc[i].component3() == rg) duplicado = 1
}

Log.d( tag: "CURA",  msg: "IMG: $img")

if (nome.matches(regex)) Toast.makeText(activity,  text: "O Nome não pode conter numeros", Toast.LENGTH_SHORT).show()

else if (nome == "") Toast.makeText(activity,  text: "O campo Nome não pode ser vazio", Toast.LENGTH_SHORT).show()

else if (rg == "") Toast.makeText(activity,  text: "O campo RG não pode ser vazio", Toast.LENGTH_SHORT).show()

else if (duplicado == 1) Toast.makeText(activity,  text: "numero do RG já consta no cadastro", Toast.LENGTH_SHORT).show()

else if (img == 0) Toast.makeText(activity,  text: "Gênero não selecionado", Toast.LENGTH_SHORT).show()
```

Cadastro de Funcionários

Fragmento - Funcionários Atualização


```
val funcSelecionado = model.funSelecionado.value

fgView.func_nome_up.setText(funcSelecionado?.component1())
fgView.func_rg_up.setText(funcSelecionado?.component3())

if (funcSelecionado?.img == R.drawable.female) fgView.feminino_up.isChecked = true
else fgView.masculino_up.isChecked = true
```

Cadastro de Funcionários

Auxiliar - Adaptador Lista Departamento

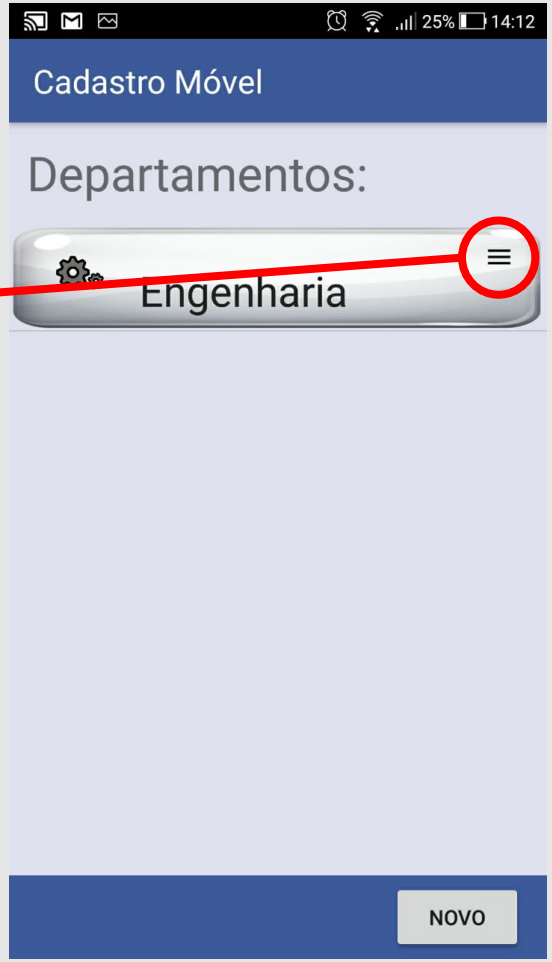

```
class adapterDep (var context: Context, var dataSource: ArrayList<dep>, var mcontext : departamentos) : BaseAdapter()
{

    private class ViewHolder(row: View?) {
        var txtName: TextView? = null
        var imgAvatar: ImageView? = null

        init {
            this.txtName = row?.findViewById(R.id.nome_dpto_list) as TextView
            this.imgAvatar = row?.findViewById(R.id.imgFundo) as ImageView
        }
    }
```

```
override fun getView(position: Int, convertView: View?, parent: ViewGroup?): View
{
    val view: View?
    val viewHolder: ViewHolder
    if (convertView == null) {
        val layout = LayoutInflater.from(context)
        view = layout.inflate(R.layout.dep_item, parent,  attachToRoot: false)
        viewHolder = ViewHolder(view)
        view.tag = viewHolder

    }

    else
    {
        view = convertView
        viewHolder = view.tag as ViewHolder
    }
```

Cadastro de Funcionários

Auxiliar - Adaptador Lista Departamento


```
    val elemento : dep = getItem(position) as dep
    viewHolder.txtName?.text = elemento.nome
    viewHolder.imgAvatar?.setImageResource(elemento.img)

    view?.opcDep?.setOnClickListener { view ->

        mcontext.menuDep(elemento, view)

    }


    return view as View
}

override fun getItem(position: Int): Any
{
    return dataSource.get(position)
}
```



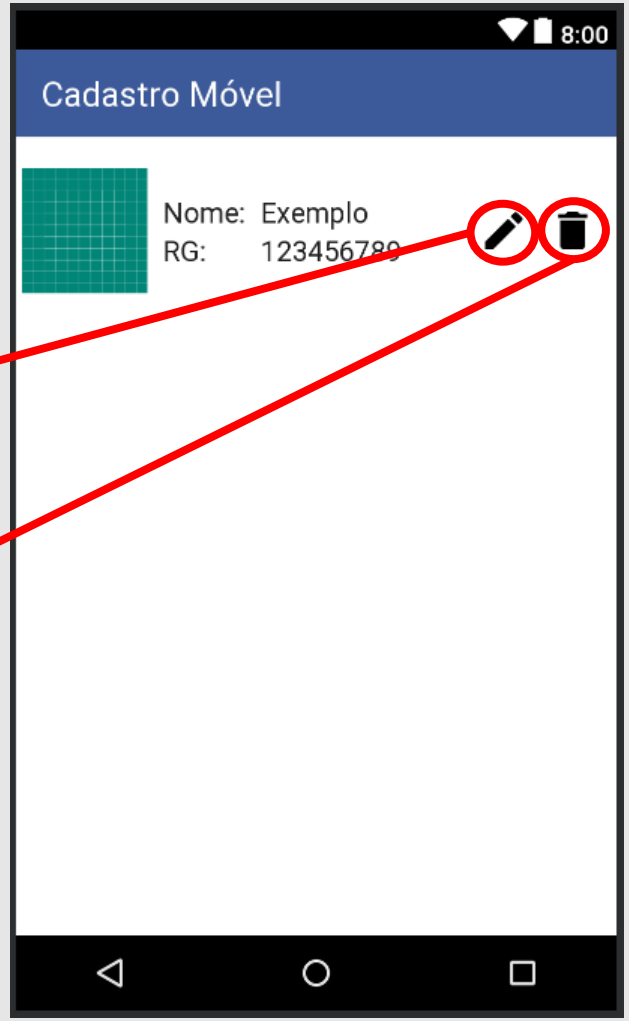
Cadastro de Funcionários

Auxiliar - Adaptador Lista Funcionários


```
private class ViewHolder(row: View?) {
    var txtName: TextView? = null
    var rg: TextView? = null
    var imgAvatar: ImageView? = null

    init {
        this.txtName = row?.findViewById(R.id.nome_fun) as TextView
        this.rg = row?.findViewById(R.id.rg) as TextView
        this.imgAvatar = row?.findViewById(R.id.foto) as ImageView
    }
}
```

```
    view?.func_alterar?.setOnClickListener{view ->

        val novof : funci = dataSource.get(position)

        mcontext.modificar(novof)
}
```

```
view?.func_remover?.setOnClickListener { view ->

    val novof: funci = dataSource.get(position)

    mcontext.alert(novof)

}
```



Cadastro de Funcionários

Auxiliar - Banco de Dados


```
class dbHelper (context: Context) : SQLiteOpenHelper(context, dbname, factory, version)
{
    override fun onCreate(db: SQLiteDatabase?)
    {

        val query = "create table departamentos(id integer primary key autoincrement," + "name varchar(30), sigla varchar(5), img integer)"
        val query2 = "create table funcionarios(id integer primary key autoincrement," + "nome varchar(30), rg varchar(10), idpto integer, foto integer)"

        db?.execSQL(query)
        db?.execSQL(query2)

    }
```

```
companion object
{
    internal val dbname = "newDB"
    internal val factory = null
    internal val version = 1
}
```

```
fun inserirDpto (name: String, sigla: String, img: Int)
{
    val db: SQLiteDatabase = writableDatabase
    val values = ContentValues()
    values.put("name",name)
    values.put("sigla",sigla)
    values.put("img",img)

    db.insert( table: "departamentos",  nullColumnHack: null, values)
    db.close()
}
```

```
fun atualizarDep( id: Int, name: String, sigla: String, img: Int)
{
    val db: SQLiteDatabase = writableDatabase
    val values = ContentValues()
    values.put("name",name)
    values.put("sigla",sigla)
    values.put("img",img)

    db.update( table: "departamentos", values,  whereClause: "id == $id",  whereArgs: null)
    db.close()
}
```

Cadastro de Funcionários

Auxiliar - Banco de Dados


```
fun listarDpto() : ArrayList<dep>
{
    Log.d( tag: "CURA",  msg: "QUERY DEPARTAMENTOS")

    val db = writableDatabase
    val query = "SELECT * FROM departamentos"
    val search = db.rawQuery(query,  selectionArgs: null)
    var lista : ArrayList<dep> = ArrayList()

    if (search.moveToFirst()) {
        do {
            var Dpto = dep(
                search.getString( columnIndex: 1),
                search.getInt( columnIndex: 3),
                search.getString( columnIndex: 2),
                search.getInt( columnIndex: 0)
            )


            lista.add(Dpto)
        } while (search.moveToNext())
    }

    search.close()

    return lista
}
```

```
fun listarFun(id : Int) : ArrayList<funci>
{
    Log.d( tag: "CURA",  msg: "QUERY FUNCIONARIOS")

    val db = writableDatabase
    var query = "SELECT * FROM funcionarios WHERE idpto = '$id'"
    if (id == -1) query = "SELECT * FROM funcionarios"
    val search = db.rawQuery(query,  selectionArgs: null)
    var lista : ArrayList<funci> = ArrayList()


        if (search.moveToFirst()) {
            do {
                var nfunci = funci(
                    search.getString( columnIndex: 1),
                    search.getInt( columnIndex: 0),
                    search.getString( columnIndex: 2),
                    search.getInt( columnIndex: 3),
                    search.getInt( columnIndex: 4)
                )
                lista.add(nfunci)
            } while (search.moveToNext())
        }

    search.close()

    return lista
}
```

Cadastro de Funcionários

Auxiliar - Banco de Dados


```
fun inserirFun(nome: String, rg: String, idDpto: Int, foto: Int)
{
        val db: SQLiteDatabase = writableDatabase
        val values = ContentValues()
        values.put("nome", nome)
        values.put("rg", rg)
        values.put("idpto", idDpto)
        values.put("foto", foto)

        db.insert( table: "funcionarios",  nullColumnHack: null, values)
        db.close()
}
```

```
fun deletarFun( id : Int)
{
    val db = this.writableDatabase

    db.delete( table: "funcionarios",  whereClause: "idpto == + $id",  whereArgs: null)

    db.close()
}
```

```
fun atualizarFun(id : Int, name : String, rg : String, img : Int)
{
    val db: SQLiteDatabase = writableDatabase
    val values = ContentValues()
    values.put("nome",name)
    values.put("rg",rg)
    values.put("foto",img)

    db.update( table: "funcionarios", values,  whereClause: "id == $id",  whereArgs: null)

    db.close()
}
```

```
fun deletarFunId( id : Int)
{
    val db = this.writableDatabase

    db.delete( table: "funcionarios",  whereClause: "id == + $id",  whereArgs: null)

    db.close()
}
```

Cadastro de Funcionários

Links

Código:

https://github.com/Felpz13/CadastroFuncionarios

Aplicativo:

https://drive.google.com/file/d/1OpDqN_qkJ_0lXQAnrrG89Ep0bRjT-pU5/view