**Samuel Guerra de Aquino**  *N 34*

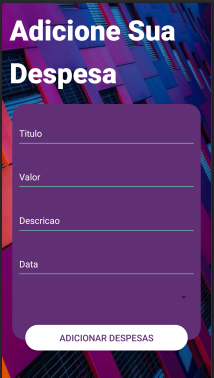
**Mariana Santos Fernandes** *N 28*

**Controle de Gasto**

*A ideia do App seria você fazer a inserção de seu Salário Líquido e sempre que houver alguma despesa iremos registrar para o usuário se der conta de sua situação financeira, ajudando na* administração *de seu capital.*

***Cadastro de Gasto***

*Design*



*Foi necessário criar a tela* ***Cadastro de Gasto*** *para o usuário conseguir cadastrar seu gasto, para assim o app fazer um balanço monetário e informar ao usuário seus Gasto, em porcentagens e dados uteis que serão implementados para ajudar tanto em uma educação financeira quanto no entendido do gasto em si.*

***Conexao.java***

*(banco)*

public class Conexao extends SQLiteOpenHelper {  
 public static final String *name* = "bdControleDeGasto.db";  
 public static final int *version* = 1;  
 //com ajuda do openHelper a classe de criação do banco é criada e configurada da maneira desejada por mim  
 public Conexao(@Nullable Context context) {  
 super(context, *name*, null, *version*);  
 }@Override  
 public void onCreate(SQLiteDatabase db) {

//aqui usamos uma função na qual executa um comando sql, e aqui eu coloquei a criação da tabela juntamente com seus atributos  
 db.execSQL("" +  
 "CREATE TABLE tbGasto (" +  
 "idGasto integer primary key autoincrement," +  
 "tituloGasto Varchar(40) not null," +  
 "descGasto Varchar(60) not null," +  
 "valorGasto Varchar(60) not null," +  
 "dataGasto Varchar(10) not null," +  
 ")");  
 }  
 @Override  
 public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {  
 }  
}// um molde da orientação a objeto com atributos e classes de Getters e Setters  
public class Gasto {  
 private String titulo, descricao, valor, data;  
 public String getTitulo() {  
 return titulo;  
 }  
 public void setTitulo(String titulo) {  
 this.titulo = titulo;  
 }  
 public String getDescricao() {  
 return descricao;  
 }  
 public void setDescricao(String descricao) {  
 this.descricao = descricao;  
 }  
 public String getValor() {  
 return valor;  
 }  
 public void setValor(String valor) {  
 this.valor = valor;  
 }  
 public String getData() {  
 return data;  
 }  
 public void setData(String data) {  
 this.data = data;  
 }  
}

***GastoDAO.java***

*(União entre banco e molde)*

// Classe que junta classe conexao e o molde de Gasto  
  
public class GastoDAO {  
  
 private Conexao c;  
 private SQLiteDatabase banco;  
  
 public GastoDAO(Context context){  
 c = new Conexao(context);  
   
 //primeiro criamos o objeto c de conexao e interpretamos eles e jogamos na variável banco  
   
 banco = c.getWritableDatabase();  
 }  
 //essa função inicia os inserts  
 public long insert(Gasto gasto){  
 //chamamos uma função interna que realiza as funções de insert  
   
 ContentValues inserindo = new ContentValues();  
   
 //e com a classe molde pegamos os valores inseridos no formulario e colocamos o atributo da tabela que colocaremos  
   
 inserindo.put("tituloGasto", gasto.getTitulo());  
 inserindo.put("descGasto", gasto.getDescricao());  
 inserindo.put("valorGasto", gasto.getValor());  
 inserindo.put("dataGasto", gasto.getData());  
   
 //e com a função insert na qual eu boto a tabela e a função inserindo para inserir todos os valores resgatados e devolvo no return o código  
   
 return banco.insert("tbGasto", null, inserindo);  
  
 }

***MainActivity.java***

*(Tela Cadastro de Gasto)*

//botão para adicionar  
public void addDespesa(View view) {  
 //declaro as classes explicadas anteriormente

Gasto gasto = new Gasto();  
 GastoDAO GDAO = new GastoDAO(this);  
  
 //seto dentro do objeto os dados do formulário

gasto.setTitulo(edTitulo.getText().toString());  
 gasto.setDescricao(edDescricao.getText().toString());  
 gasto.setValor(edValor.getText().toString());  
 gasto.setData(edData.getText().toString());  
  
 //uso o método criado na classe GastoDAO para realizar o insert e atribuo o retorno(o código) a uma variável  
 long getId = GDAO.insert(gasto);

//após o resgate do id cadastrado do banco eu apresento em um toast para ser informado da conclusao confirmada da inserção

Toast.*makeText*(this, (int) getId, Toast.*LENGTH\_LONG*).show();  
  
  
}

*Com essas Classes conseguimos executar os* ***Inserts*** *assim implanto um banco de dados já funcionou em armazenamento assim deixando o app bem mais dinâmico.*

***Selects e Delete***

***PessoaDao.java***

*(conexão entre banco e objeto molde)*

//essa funcao faz uma busca no banco devolvendo um objeto list  
public List<gastos> listar() {  
 //declaramos um arraylist  
 List< gastos> gastos = new ArrayList<>();  
 //criamos um objeto do tipo Cursor para usarmos como direcao no banco, nela selecionamos a tabela e os atributos desejados  
 Cursor cursor = banco.query("tbGasto", new String[] {"idGasto","tituloGasto", "descricaoGasto",  
 "valorGasto", "dataGasto"},  
 null, null, null, null, null);  
 //aqui temos um laco no qual com os dados adquiridos pegamos e jogamos dentro do objeto "pessoa" e adicionamos ao nosso molde Gasto  
 while (cursor.moveToNext()){  
 Gasto gasto = new Gasto();  
 gasto.setId(cursor.getInt(0));  
 gasto.setTitulo(cursor.getString(1));  
 gasto.setDescricao(cursor.getString(2));  
 gasto.setValor(cursor.getString(3));  
 gasto.setData(cursor.getString(4));  
  
 //criamos o objetos  
 gastos.add(gasto);  
 }  
 //e retornamos o objeto no qual devemos receber num arraylist para recebermos diversos valores e trata-los  
 return gastos;  
}  
//uma funcao curta para deletar que pede apenas o codigo da linha q deseja excluir(usamos funcoes internas do sqlite para essa exclusao(metodo .delete))  
public void deletar(int id){  
 banco.delete("tbGasto", "id = " + id, null);  
}

***Gasto.java***

*(modelo)*

//formatamos os valores para ficar mais agradavel  
@Override  
public String toString(){  
 String retorno = String.*format*("Id: %d\nNome: %s\nE-mail: %s\nTelefone: %s\nAssunto: %s\nMensagem: %s", id, nome, email, telefone, assunto, mensagem);  
 return retorno;  
}

***Select.java***

*(classe para realizar as buscas)*

//utilizamos a funcao listar para consultarmos dados do banco jogando num arraylist chamado pessoas  
pessoas = dao.listar();  
  
ArrayAdapter<Pessoa> adapter = new ArrayAdapter <Pessoa> (this, android.R.layout.*simple\_list\_item\_1*, pessoas);  
  
listview.setAdapter(adapter);

//funcao para deletar dados  
public void deletar(View view) {  
 //criamos um alert  
 AlertDialog.Builder mensagem = new AlertDialog.Builder(this);  
 mensagem.setTitle("Deletar");  
 mensagem.setMessage("Qual o id da linha que deseja deletar?");  
 mensagem.setNeutralButton("Cancelar", null);  
 //criamos um edit text no alert e jogamos numa variavel input para usarmos como referencia para a exclusao, usando o codigo como referencia  
 final EditText input = new EditText(this);  
 input.setRawInputType(InputType.*TYPE\_CLASS\_NUMBER*);  
 mensagem.setView(input);  
  
 //caso confirme ele executara esse bloco de codigo  
 mensagem.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 //pegano o dado (id) para referenciarmos na rotina  
 int id = Integer.*parseInt*(input.getText().toString());  
 //e utilizamos a funcao deletar usando o id que resgatamos  
 dao.deletar(id);  
 }  
 });  
 mensagem.show();