**Checkpoint 1 – Android Kotlin Developer**

**Participantes: Eloisa Araujo (94604), Julia Carvalho (96106), Leonardo Minniti (76172) e William Tedros (87429)**

**Professor Ewerton Carreira**

**Classe: PRODUTOSDAO**

Classe responsável por manipular os dados de produtos.

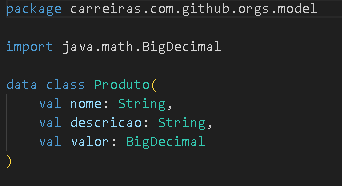


**adiciona(produto: Produto) >** Método para adicionar um produto à lista de produtos.

**buscaTodos(): List<Produto> >** Método para buscar todos os produtos. Retorna uma lista contendo todos os produtos cadastrados.

**Classe: PRODUTO**

Classe que representa um produto.



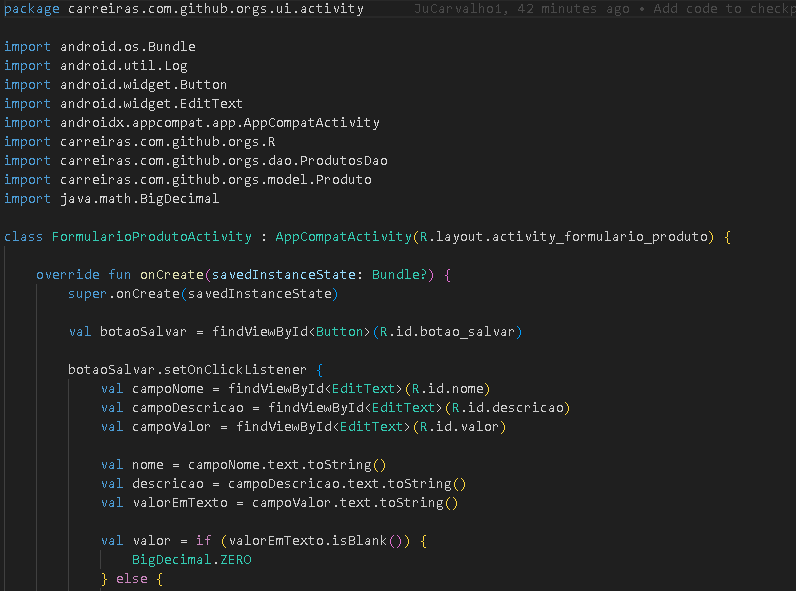
**nome: String >** Nome do produto.

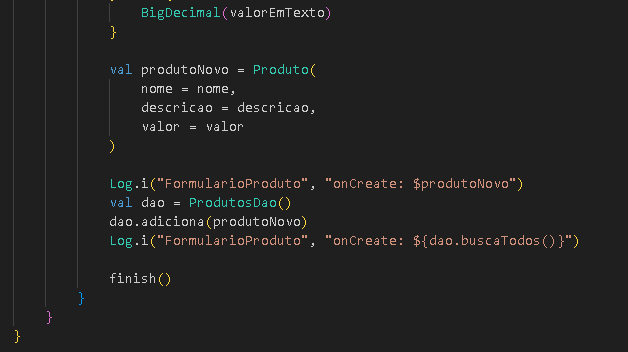
**descricao: String >** Descrição do produto.

**valor: BigDecimal >** Valor do produto.

**FormularioProdutoActivity**

Classe responsável pela atividade de formulário de produtos.





**produtos class FormularioProdutoActivity : AppCompatActivity(R.layout.activity\_formulario\_produto) >** Classe responsável pela atividade de formulário de produtos

**onCreate(savedInstanceState: Bundle?) >** Método de inicialização da atividade.

**savedInstanceState >** O estado da instância salva anteriormente.

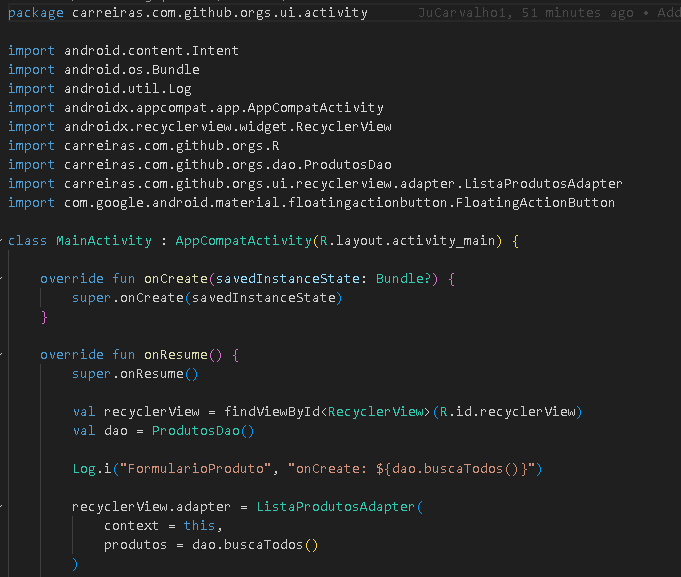
Descrição do Comportamento:

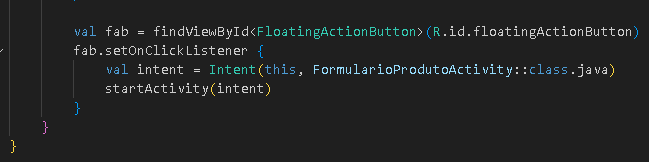
* Configura o listener para o clique no botão de salvar.
* Captura os dados inseridos nos campos de entrada.
* Cria um novo objeto Produto com os dados inseridos.
* Adiciona o novo produto ao banco de dados.
* Finaliza a atividade.

**botaoSalvar.setOnClickListener >** Configuração do listener para o clique no botão salvar

**MainActivity**

Classe principal da activity principal (MainActivity).





**onCreate(savedInstanceState: Bundle?) >** Método de inicialização da atividade.

**onResume() >** Método chamado quando a atividade está sendo retomada.

Descrição do Comportamento:

- Obtém uma instância do DAO de produtos.

- Configura o adaptador para o RecyclerView com a lista de produtos obtida do DAO.

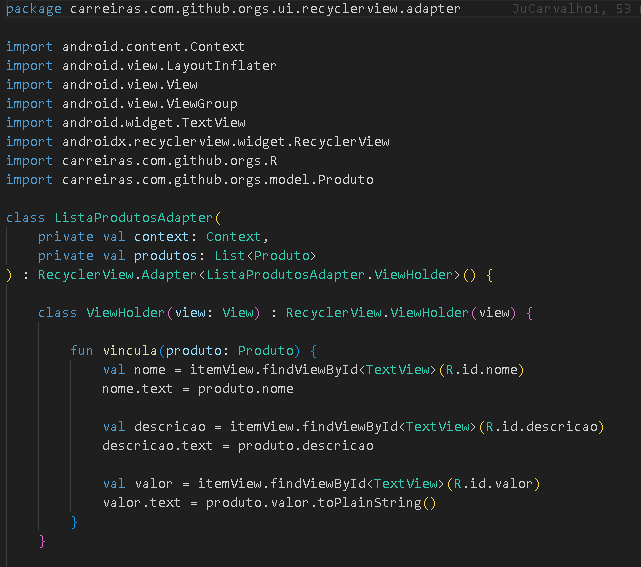
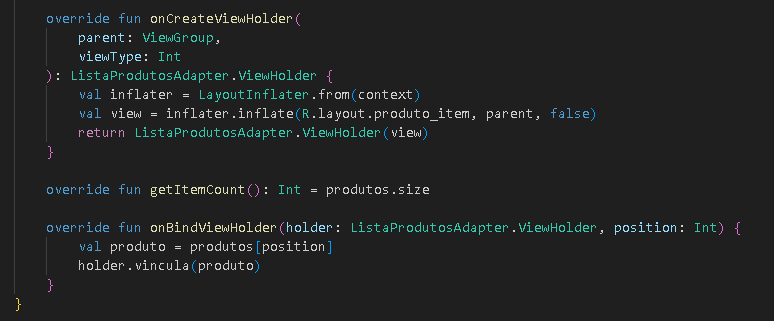
- Configura um listener para o clique no FloatingActionButton para abrir a atividade de formulário de produtos.

**Bundle >** classe que tem tudo que preciso pra manter o estado da ACTIVITY

"?"-> pra forçar o NULO

**ListaProdutosAdapter**

Adaptador para exibir a lista de produtos em um RecyclerView.

**class ListaProdutosAdapter >** Adaptador para exibir a lista de produtos em um RecyclerView

**class ViewHolder(view: View) : RecyclerView.ViewHolder(view) >** ViewHolder que armazena as referências aos elementos da interface do item de produto

**override fun onCreateViewHolder >** Método chamado quando o ViewHolder é criado

**override fun onBindViewHolder >** Método chamado para exibir os dados do produto em uma posição específica

**onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder >** Método chamado quando o ViewHolder é criado. Retorna um novo ViewHolder que contém a visualização do item de produto.

**- parent >** O ViewGroup pai no qual o novo ViewHolder será adicionado após ser vinculado.

**- viewType >** O tipo de visualização do novo ViewHolder.

**getItemCount(): Int >** Retorna o número total de itens na lista de produtos.

**onBindViewHolder(holder: ViewHolder, position: Int) >** Método chamado para exibir os dados do produto em uma posição específica.

**- holder >** O ViewHolder a ser atualizado.

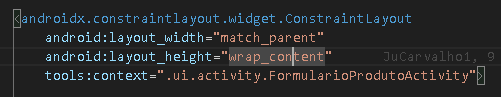
**- position >** A posição do item na lista.

**ViewHolder(view: View) >** ViewHolder que contém as referências aos elementos da interface do item de produto.

**vincula(produto: Produto) >** Método para vincular os dados do produto aos elementos da interface.

**- produto >** O objeto Produto a ser exibido.

**XML**



**ConstraintLayout >** é um tipo de layout que permite posicionar e dimensionar widgets de forma flexível e eficiente, através de restrições relativas entre elementos na interface.

**android:layout\_width >** Mexe na largura do elemento.

**android:layout\_height >** Mexe na altura do elemento.

**"match\_parent" >** ocupa todo o espaço do seu elemento pai.

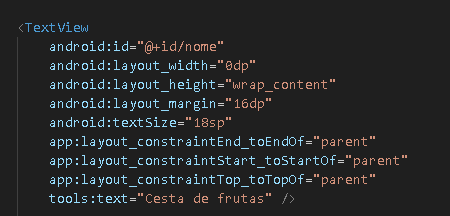
**"wrap\_content" >** ocupa exatamente o conteúdo do componente, sem espaço extra.



**constraintEnd\_toEndOf="parent" >** Define que o lado direito do componente deve ser alinhado com o lado direito do seu elemento pai.

**constraintStart\_toStartOf="parent" >** Especifica que o lado esquerdo do componente deve ser alinhado com o lado esquerdo do seu elemento pai.

**constraintTop\_toTopOf="parent" >** Faz com que o topo do componente seja alinhado com o topo do seu elemento pai.



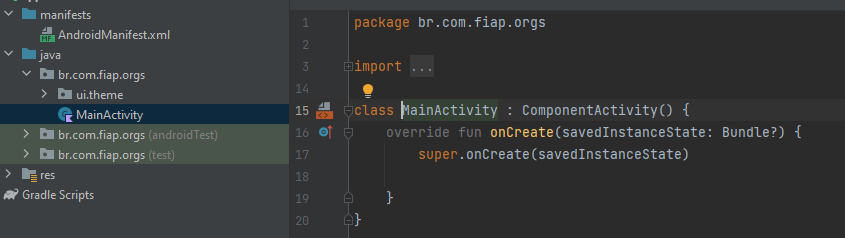
**TextView >** é um componente de interface do Android usado para exibir texto para o usuário. Ele pode ser personalizado.

**View >** Qualquer objeto que esteja dentro de uma activity, exemplo uma caixa de texto.

**OUTRAS ANOTAÇÕES**

****

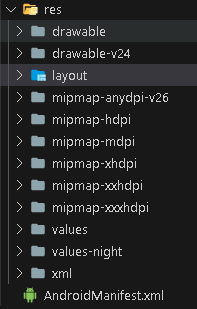
**ComponentActivity >** Abstração que contém tudo que precisa para trabalhar, funções, métodos. Tem que sempre ser herdado



**CICLO DE VIDA** > Começa com oneCreate e vem passando um passo por vez

On create, primeiro estágio de ciclo de vida, só rodar uma vez.

Método create, dentro da classe pai.



**PASTA RES** > recursos, tudo que vou utilizar para desenvolver, arquivos xml, layout, cores...