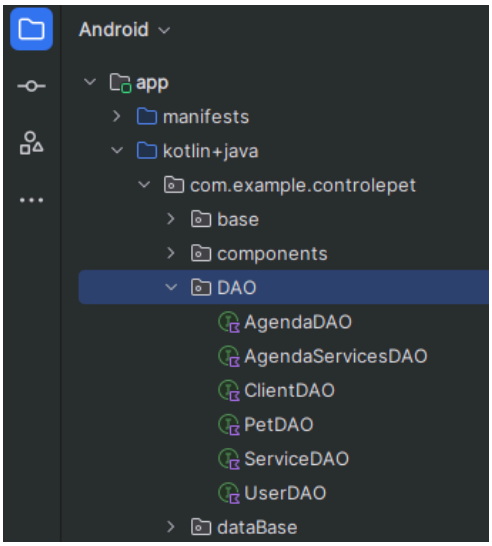
Controle Pet

Aplicativo para agendamento de serviços pet, com o objetivo de ajudar o profissional a agendar o horário de cada cliente e seu pet, com o serviços fornecidos.

Arquitetura de Software:

Desenvolvi o aplicativo para Android utilizando Jetpack Compose, assim utilizei a arquitetura MVVM para o desenvolvimento,

- DAO e Repository: Para o gerenciamento do dados, utilizei o ROOM DAO: itens salvos na pasta DAO; e repository para intermediar o acesso ao DAO
- Model para os arquivos model que representam os dados.
- ViewModel: para acesso aos dados utilizando repository e mostrar na screen
- Screens: Compose - arquivos de layout



- Figura 1: DAO

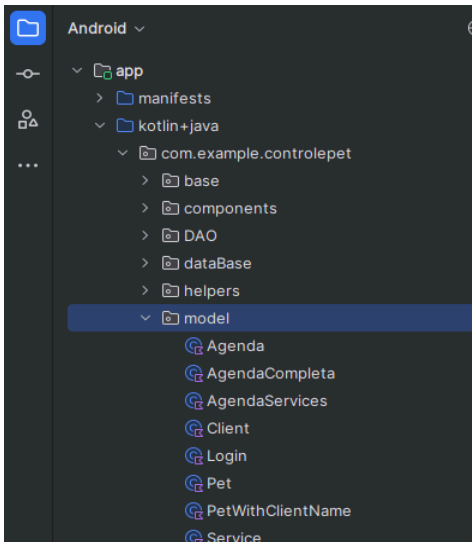


Figura 2: Model

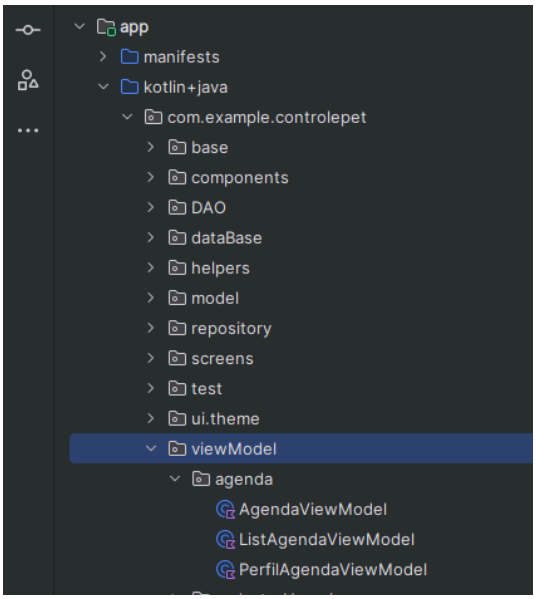


Figura 3: ViewModel

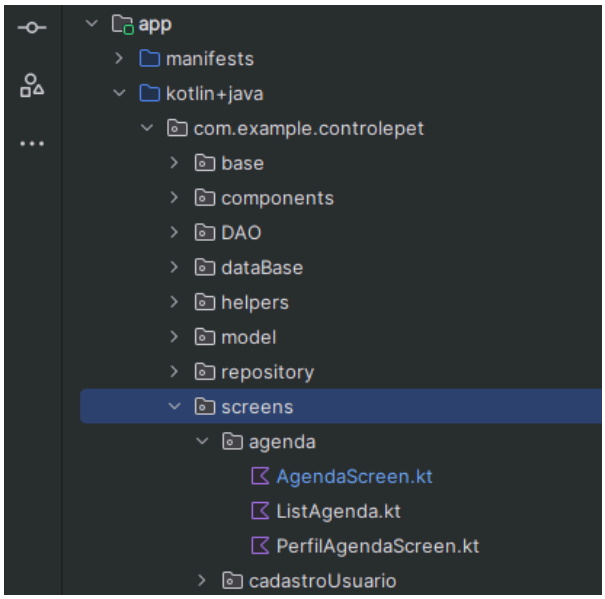


Figura 4: Screens

Clean Code: Seguir as diretrizes básicas de código limpo para garantir legibilidade e manutenção.

- Código com separação de responsabilidade
- Funções com responsabilidades pequenas
- Funções e nomes de variáveis claros
    - Funções da viewModel para agenda

```
- val listaPets:
- fun saveAgenda(servicos: List<ServiceSelecionado>,
  petId: Int)
- fun loadAgendaForEdit(agendaId: Int)
- fun onDelete(agendaId: Int) {
- open fun loadAgendaCompleta(id: Int) {
```

    - Comentários em funcionalidades para entender o que irá fazer
    - Tratamento dos erros

Injeção de Dependência: Implementar o uso de Dependency Injection para promover um código mais modular e testável.

- Classes recebendo as dependências necessárias, sem recriar dentro do código, facilitando para os testes passar as dependências tambem

```
16
17  class ListAgendaViewModel(
18      private val repo: AgendaRepository,
19      private val dispatcher: CoroutineDispatcher = Dispatchers.Main
20  ): ViewModel() {
```

E nos testes podendo passar um fakeRepository e o testDispatcher

```
82      @Test
83      fun `agendaList should emit list with items`() = runTest {
84
85          // cria o repositório com as agendas mockadas
86          fakeRepository = FakeAgendaRepository( fixedTime, listOf(agenda1, agenda2))
87          viewModel = ListAgendaViewModel(fakeRepository)
88
```

```
106     @Test
107     fun `onDelete should emit success toast message`() = runTest {
108         fakeRepository = FakeAgendaRepository(fixedTime,  listOf(agenda1, agenda2))
109         viewModel = ListAgendaViewModel(fakeRepository, testDispatcher)
110
```

Testes unitários

Teste realizados na funcionalidade de agendamento de serviços

- Realizados nas viewModel da agenda, AgendaViewModel, PerfilAgendaViewMode e LisAgendaViewModel
- Criado um FakeAgendaRepository

No AgendaViewModelTest foi criado os teste para as funções de saveAgenda para adicionar um novo item, carregar a agenda para edição.

Em ListAgendaViewModelTest criado testes para as funcionalidade de listagem de agenda com itens (AgendaList with items), listagem sem itens (AgendaList empty list), e funcionalidade de deletar o item (onDelete toast message)

E no PerfilAgendaViewModelTest criado o teste para carregar a agendaCompleta loadAgendaCompleta.