# Этап 5. ARCHITECTURE.md — MoodTail

Версия: MVP 1.0

Дата: 2025-07-31

Автор: Nikita Voitkus

## **6** Цель архитектуры

MoodTail — это iOS-приложение для владельцев собак, помогающее отслеживать и понимать эмоциональное состояние питомца. Архитектура проекта строится на принципах масштабируемости, модульности, тестируемости и простоты поддержки.

## 🍣 Архитектурный шаблон

- **Шаблон**: MVVM
- UI: SwiftUI (разделение на View / Component)
- Логика: Выделена в сервис AdviceEngine и AdviceRules
- **Хранение:** Core Data через MoodStorageProtocol
- **DI:** Простая реализация через MoodTailContainer
- **Масштабируемость:** модульная структура + возможный переход на CloudKit без изменений бизнес-логики

## 🧩 Модули приложения

- 1. MoodLogger
  - View: MoodLoggerView.swift
  - ViewModel: MoodLoggerViewModel.swift

- Model: MoodEntry.swift
- UI-компоненты:
  - MoodlconView
  - SaveButton
- Сценарии:
  - Выбор эмоции
  - Опциональный ввод заметки
  - Сохранение с валидацией
  - Анимация подтверждения с withAnimation

#### 2. HistoryTracker

- **View:** HistoryTrackerView.swift
- ViewModel: HistoryTrackerViewModel.swift
- UI-компоненты:
  - MoodRowView
  - EmptyStateView
- Сценарии:
  - История эмоций в виде списка
  - Фильтрация по дате и эмоции
  - Просмотр деталей записи

#### 3. AdviceEngine

- Файл: AdviceEngine.swift
- Служебный модуль логики: AdviceRules.swift
- **ФУНКЦИЯ:** getAdvice(for history: [MoodEntry]) → String?
- Правила:

- ∘ ≥ 3 тревожных эмоции подряд
- отсутствие позитивных эмоций
- Интеграция: вызывается из ViewModel, логика изолирована

## **| Хранение данных**

#### **MoodEntry** (Core Data)

- id: UUID
- mood: MoodType (enum)
- note: String?
- date: Date

#### MoodStorageProtocol

```
protocol MoodStorageProtocol {
  func save(_ entry: MoodEntry) throws
  func fetchAll() → [MoodEntry]
}
```

- Реализация: CoreDataMoodStorage
- Моки: MockMoodStorage для тестов и Preview

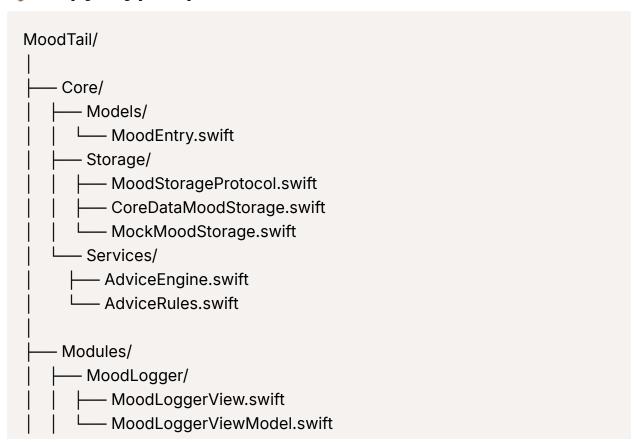
## **/** Тесты

- AdviceEngineTests.swift
- MoodStorageTests.swift
- HistoryFilterTests.swift
- Используются in-memory Core Data и моки
- В планах: ViewModelTests.swift

## Навигация и UI-состояния

- Используется NavigationStack , TabView
- Экраны:
  - MoodLoggerView
  - HistoryTrackerView
- Поддерживаемые состояния UI:
  - loading
  - empty ( EmptyStateView )
  - o error (в планах)
  - o success
- Планируется ViewState enum для управления состоянием UI

## 📦 Структура проекта



├── MoodIconView.swift
│ ├── SaveButton.swift
│
│
│ └── FeedbackBanner.swift (в планах)
   Infrastructure/ 
MoodTailApp.swift
PreviewContent/

Secretary of the Control of the Cont	Временные решения и TODO
☐ Vi	iewState enum во ViewModel
☐ CI	loudKit-реализация (переход через MoodStorageProtocol)
Fe	eedbackBanner / ToastView ДЛЯ СОВЕТА
П	оддержка мульти-питомцев
□ iP	ad-адаптивность
☐ CI	I/CD (GitHub Actions)
П	олная локализация: ru , en , pl
□ Ac	ccessibility: VoiceOver, Dynamic Type
☐ C	оздать DEV_GUIDE.md для новых участников команды

## 🌌 Диаграмма модулей (Mermaid)

```
\begin{array}{l} \text{graph TD} \\ & \text{A[MoodLogger]} \rightarrow \big| \text{uses} \big| \text{ B[MoodStorageProtocol]} \\ & \text{C[HistoryTracker]} \rightarrow \big| \text{uses} \big| \text{ B} \\ & \text{C} \rightarrow \text{D[AdviceEngine]} \\ & \text{D} \rightarrow \text{E[AdviceRules]} \\ & \text{A} \rightarrow \text{D} \\ & \text{B} \rightarrow \text{F[CoreDataMoodStorage]} \\ & \text{B} \rightarrow \text{G[MockMoodStorage]} \end{array}
```

## **Ш**Итог

MoodTail построено на зрелой, масштабируемой архитектуре. Проект готов к выходу в TestFlight после финальной стабилизации и покрытия ключевых путей юнит-тестами. Все критические зоны отмечены в TODO и подготовлены к доработке без архитектурных изменений.