

# Этап 5. ARCHITECTURE.md — MoodTail

Версия: MVP 1.0

Дата: 2025-07-31

Автор: Nikita Voitkus

---

## 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:**
    - `MoodIconView`
    - `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:**

- $\geq 3$  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` )
  - `error` (в планах)
  - `success`
- Планируется `ViewState` enum для управления состоянием UI



## Структура проекта

```
MoodTail/  
├── Core/  
│   ├── Models/  
│   │   └── MoodEntry.swift  
│   ├── Storage/  
│   │   ├── MoodStorageProtocol.swift  
│   │   ├── CoreDataMoodStorage.swift  
│   │   └── MockMoodStorage.swift  
│   └── Services/  
│       ├── AdviceEngine.swift  
│       └── AdviceRules.swift  
├── Modules/  
│   ├── MoodLogger/  
│   │   ├── MoodLoggerView.swift  
│   │   └── MoodLoggerViewModel.swift
```

```

|   |   | HistoryTracker/
|   |   | | HistoryTrackerView.swift
|   |   | | HistoryTrackerViewModel.swift
|   |
|   | SharedUI/
|   | | MoodIconView.swift
|   | | SaveButton.swift
|   | | EmptyStateView.swift
|   | | MoodRowView.swift
|   | | FeedbackBanner.swift (в планах)
|   |
|   | Infrastructure/
|   | | MoodTailContainer.swift
|   |
|   | MoodTailApp.swift
|   | PreviewContent/

```

## Временные решения и TODO

- ☐ ViewState enum во ViewModel
- ☐ CloudKit-реализация (переход через `MoodStorageProtocol`)
- ☐ `FeedbackBanner` / `ToastView` для совета
- ☐ Поддержка мульти-питомцев
- ☐ iPad-адаптивность
- ☐ CI/CD (GitHub Actions)
- ☐ Полная локализация: `ru`, `en`, `pl`
- ☐ Accessibility: VoiceOver, Dynamic Type
- ☐ Создать `DEV_GUIDE.md` для новых участников команды

## Диаграмма модулей (Mermaid)

```
graph TD
  A[MoodLogger] -->|uses| B[MoodStorageProtocol]
  C[HistoryTracker] -->|uses| B
  C --> D[AdviceEngine]
  D --> E[AdviceRules]
  A --> D
  B --> F[CoreDataMoodStorage]
  B --> G[MockMoodStorage]
```



## 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.