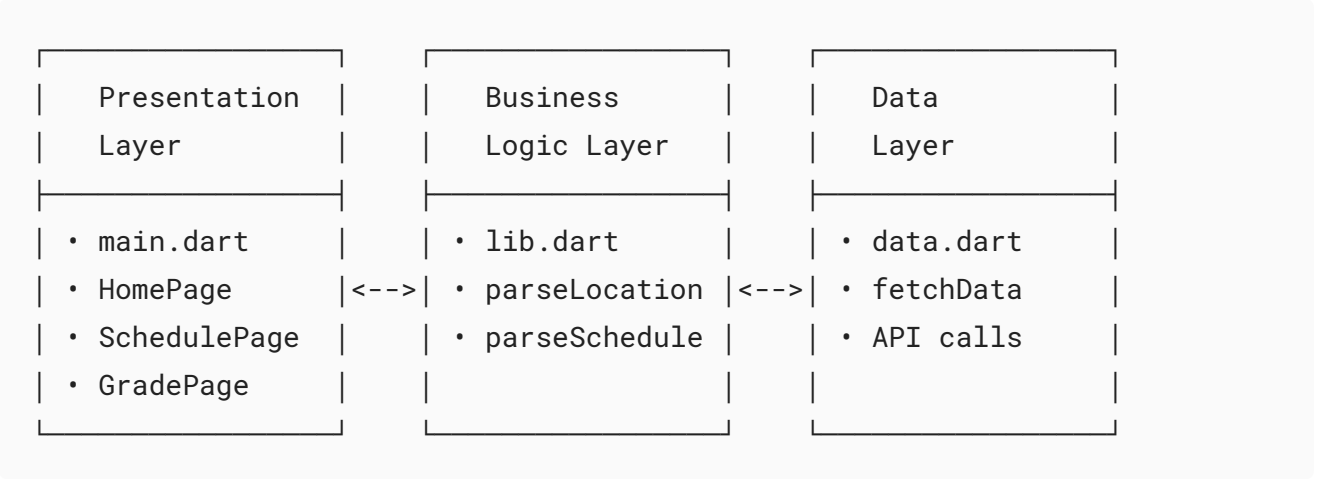
Dash Duck 架构设计文档

1. 整体架构

1.1 架构模式

采用 **MVVM (Model-View-ViewModel)** 架构模式，结合 Flutter 的声明式UI特性。



1.2 关键设计决策

决策1: 使用Flutter框架

原因:

- 跨平台开发，一套代码支持iOS和Android
- 热重载提高开发效率
- 丰富的UI组件库

决策2: 模块化设计

原因:

- 便于维护和扩展
- 职责分离，降低耦合
- 支持单独测试

决策3: 智能位置解析

创新点:

- 自动识别校区、建筑物、教室
- 使用正则表达式提取关键信息

- 生成简洁易读的位置标识

2. 核心模块设计

2.1 数据层 (data.dart)

- **职责**: 数据获取和预处理
- **接口**: fetchScheduleData(), parseSchedule()
- **扩展性**: 支持多数据源集成

2.2 业务逻辑层 (lib.dart)

- **职责**: 业务规则处理
- **核心算法**: 位置解析算法
- **创新特性**: 中山大学校区智能识别

2.3 表现层 (main.dart)

- **职责**: UI展示和用户交互
- **设计模式**: 组件化设计
- **用户体验**: 响应式布局，暗色主题

3. 技术栈选择

层次	技术选择	理由
前端框架	Flutter	跨平台，高性能
开发语言	Dart	类型安全，异步支持
状态管理	StatefulWidget	轻量级，适合小型应用
测试框架	flutter_test	官方支持，功能完整
构建工具	Flutter CLI	标准化构建流程