TodoList 应用需求与设计文档

1. 项目概述

TodoList 是一个专为Android平台开发的待办事项管理应用，由 Flutter 前端和 GraphQL 后端构成。应用允许用户创建、管理和组织待办事项，并支持数据与配置的服务器同步。该应用通过.apk文件直接安装，不通过应用商店分发。

2. 系统架构

技术栈

- **前端**：Flutter (Dart)
  - 依赖：graphql, provider, shared_preferences, intl
- **后端**：Go 语言 GraphQL 服务
  - 使用 gqlgen 框架
- **数据库**：SurrealDB (本地开发环境)
- **目标平台**：Android 5.0及以上设备

系统组件

1. **客户端 (Flutter)**
   - 用户界面
   - 本地存储
   - GraphQL 客户端
   - 同步管理器
2. **服务端 (Go)**
   - GraphQL API
   - 数据库连接器
   - 同步服务
3. **数据库 (SurrealDB)**
   - 存储待办事项数据

3. 功能规格

3.1 待办事项管理

- 创建新的待办事项
- 编辑现有待办事项
- 删除待办事项
- 为待办事项添加以下属性：
  - 标题（必填）
  - 描述（可选）
  - 类别标签
  - 截止日期/时间
  - 地点
  - 状态（完成/未完成）

3.2 待办事项分类

- 支持自定义类别标签
- 按类别组织待办事项
- 按类别筛选待办事项

3.3 待办事项查看

- 按时间顺序查看待办事项
- 按类别筛选待办事项
- 按时间范围筛选待办事项
- 按完成状态筛选待办事项

3.4 待办事项状态管理

- 通过点击复选框修改待办事项状态（完成/未完成）

3.5 服务器连接

- 配置服务器地址和端口
- 测试连接可用性
- 支持离线操作

3.6 数据同步

- 将本地待办事项与配置的服务器同步
- 支持手动触发同步
- 支持离线操作，网络恢复后可同步

4. 数据模型

4.1 待办事项 (TodoItem)

```
{
  id: String,
  title: String,
  description: String (optional),
  category: String,
  dueDate: DateTime (optional),
  location: String (optional),
  isCompleted: Boolean,
  createdAt: DateTime,
  updatedAt: DateTime
}
```

4.2 类别 (Category)

```
{
  id: String,
  name: String,
  color: String
}
```

4.3 服务器配置 (ServerConfig)

```
{
  address: String,
  port: Number,
  isConnected: Boolean
}
```

5. API 设计 (GraphQL)

5.1 主要查询 (Queries)

- `getTodos`：获取所有待办事项
- `getTodosByCategory`：按类别获取待办事项
- `getTodosByStatus`：按状态获取待办事项
- `getCategories`：获取所有类别

5.2 主要变更 (Mutations)

- `createTodo`：创建新的待办事项
- `updateTodo`：更新待办事项
- `deleteTodo`：删除待办事项
- `toggleTodoStatus`：切换待办事项状态
- `createCategory`：创建新的类别
- `syncTodos`：同步待办事项

6. 用户界面设计

6.1 主要界面

- **主屏幕**：显示待办事项列表，支持筛选和排序
- **创建/编辑屏幕**：用于添加或编辑待办事项
- **类别管理界面**：管理类别标签
- **设置界面**：配置服务器和同步选项

6.2 用户交互流程

1. 用户打开应用，查看待办事项列表
2. 用户可添加新的待办事项
3. 用户可点击待办事项前的复选框切换完成状态
4. 用户可点击待办事项查看或编辑详情
5. 用户可向左滑动待办事项删除
6. 用户可在设置中配置服务器连接

6.3 服务器配置界面

- 允许用户输入服务器地址（IP或域名）
- 允许用户输入端口号
- 提供连接测试功能

- 显示连接状态

7. 同步流程

1. 客户端检测网络连接
2. 客户端向服务器发送本地的待办事项数据
3. 服务器处理更改并更新数据库
4. 服务器返回最新的数据
5. 客户端更新本地数据库

8. 安全考虑

- 本地存储数据安全
- 基本的数据验证

9. 性能考虑

- 优化本地数据查询
- 高效使用本地存储
- 减少网络请求频率

10. 部署与分发

- 生成签名的APK文件用于直接安装
- 不通过Google Play商店发布
- 通过项目负责人分发APK安装包

11. 可扩展性考虑

- 模块化设计允许添加新功能
- 遵循Flutter最佳实践