Fridgy-食材管理应用(Vesion 1.0.0)

作者：周玉耀

用

户

手

册

目录

[1、 软件简介 5](#_Toc1127)

[(1) 开发背景 5](#_Toc24006)

[(2) 软件介绍 5](#_Toc664)

[(3) 面向用户 6](#_Toc13377)

[(4) 主要功能 6](#_Toc26136)

[(5) 优势创新 7](#_Toc18068)

[2、 运行环境 8](#_Toc9166)

[(1) 硬件要求 8](#_Toc11778)

[(2) 软件要求 9](#_Toc4075)

[3、 软件技术 9](#_Toc30225)

[(1) 技术栈 9](#_Toc14249)

[(2) 架构设计 11](#_Toc4023)

[(3) 数据模型 13](#_Toc1602)

[(4) 软件功能 17](#_Toc26352)

[① 食材管理 17](#_Toc30373)

[② 智能状态监控与提醒 18](#_Toc30263)

[③ AI智能建议系统 19](#_Toc24483)

[④ 个性化设置与配置 20](#_Toc20073)

[⑤ 数据管理与安全 21](#_Toc5802)

[⑥ 分析功能 22](#_Toc32540)

前 言

在家庭与小微餐饮的日常管理中，食材被遗忘、临期未用与过期浪费是最常见却最难以避免的问题。Fridgy 正是为此而生的一款智能食材管理应用：它以移动端为载体，帮助用户便捷地记录冰箱与储物柜中的食材信息，自动计算保质期状态，按时推送提醒，从而显著降低浪费、提升使用效率，并在此基础上提供与当下库存相匹配的烹饪与保鲜建议，让“吃得好、用得尽”成为轻松可行的日常习惯。

Fridgy 以“轻量、直观、智能”为核心设计理念。轻量，体现在简洁的录入流程与清晰的分区管理，用户可在数秒内完成一条食材记录，并通过品类、位置与单位字典快速规范化数据；直观，体现在概览与统计页面，通过颜色与图表呈现新鲜、临期与过期的分布，重要信息一目了然；智能，体现在本地通知的精细化提醒策略与可插拔的 AI 建议能力，系统会在临期前、到期日等关键节点发出提醒，并结合当前食材生成菜谱与保鲜方案，帮助用户将“库存”转化为“灵感”。

从价值层面看，Fridgy 并非仅是一个“记账式”的食材清单，而是一套围绕家庭与小微场景构建的“低门槛库存管理”解决方案：它让食材管理与生活节奏自然融合，减少重复劳动与无效开支，兼顾健康饮食与可持续理念。无论是注重营养与效率的家庭，还是追求成本与质量平衡的小型餐饮，都能在 Fridgy 中找到顺手、可靠且可持续的工具支持。

面向未来，Fridgy 将持续迭代：在保障隐私与离线可用的前提下，进一步引入多语言支持、个性化提醒策略与更多智能洞察，拓展与电商/物联网的连接能力，帮助用户更好地管理“从采购到消费”的全链路。我们相信，科技的温度在于细节；Fridgy 希望用克制而有效的设计，让每一次打开冰箱都更有把握与期待

# 软件简介

## 开发背景

在当今快节奏的生活中，食材管理已成为家庭和小型餐饮面临的重要挑战。据统计，全球每年约有13亿吨食物被浪费，其中家庭浪费占很大比例。主要问题包括：

1. 遗忘问题：忙碌的生活节奏导致用户经常忘记冰箱中存放的食材，特别是那些被放在角落或包装不明显的物品。

2. 保质期管理困难：不同食材的保质期差异很大，用户难以准确记忆每种食材的最佳食用时间，经常出现临期或过期食材。

3. 采购重复：由于不清楚现有库存，用户经常重复购买已有食材，造成浪费和额外支出。

4. 烹饪灵感缺乏：面对冰箱中的食材，用户往往缺乏创意，不知道如何搭配制作美味健康的餐食。

5. 存储方式不当：缺乏科学的食材存储知识，导致食材提前变质。

Fridgy 正是为解决这些痛点而开发的智能食材管理应用。它通过移动端的便捷记录、智能提醒和AI建议，帮助用户建立科学的食材管理体系，显著降低浪费，提升厨房管理效率，让食材管理变得简单而有趣。

## 软件介绍

Fridgy 是一款基于 React Native + Expo 技术栈开发的跨平台智能食材管理移动应用，支持 Android 与 iOS 双平台运行。作为一款专注于食材全生命周期管理的应用，Fridgy 集成了现代移动开发的最佳实践和人工智能技术。

核心功能架构：

- 智能录入系统：支持多种录入方式，包括手动输入、扫码识别、拍照识别等，用户可根据不同场景选择最便捷的方式记录食材信息。

- 智能状态管理：系统基于食材的保质期信息，结合当前时间自动计算并更新食材状态（新鲜、临期、过期），为用户提供直观的视觉反馈。

- 个性化提醒机制：根据用户设定的提醒策略，在关键时间节点（如临期前1-3天、到期日等）推送本地通知，确保用户及时处理食材。

- AI智能建议引擎：集成多种AI模型（如DeepSeek、Zhipu等），基于用户现有食材库存，智能生成烹饪建议、营养搭配方案和保鲜技巧。

- 数据可视化分析：通过图表和统计信息，帮助用户了解食材使用模式、浪费情况，为优化采购和存储提供数据支持。

技术特色：

- 采用现代化的跨平台开发框架，确保在不同设备上的一致体验

- 支持离线使用，核心功能无需网络连接

- 数据本地存储，保护用户隐私

- 模块化架构设计，便于功能扩展和维护

## 面向用户

Fridgy 的目标用户群体广泛，主要服务于以下几类用户：

1. 家庭用户群体

- 年轻家庭：注重生活品质，希望通过科学管理提升家庭饮食健康水平

- 有孩子的家庭：需要管理多样化的食材，确保孩子营养均衡

- 老年用户：记忆力下降，需要智能提醒帮助管理食材保质期

- 独居用户：采购量小但种类多，需要精准管理避免浪费

2. 健康饮食倡导者

- 健身人群：需要精确控制食材摄入，管理蛋白质、碳水化合物等营养成分

- 素食主义者：需要管理多种植物性食材，确保营养搭配合理

- 特殊饮食需求者：如糖尿病患者、过敏体质等，需要严格管理食材选择

3. 小型餐饮经营者

- 家庭式餐厅：需要管理食材库存，控制成本，减少浪费

- 员工餐厅：需要批量管理食材，确保食品安全和营养均衡

- 咖啡厅/轻食店：需要管理易腐食材，保持食材新鲜度

- 外卖商家：需要根据订单情况合理采购和存储食材

4. 教育机构

- 学校食堂：需要科学管理学生餐食食材，确保食品安全

- 幼儿园：需要精细管理儿童餐食，注重营养搭配

5. 其他用户群体

- 食材批发商：需要管理库存，跟踪食材流转情况

- 食品研究人员：需要记录和分析食材数据

- 环保意识用户：希望通过减少食物浪费来保护环境

## 主要功能

Fridgy 提供全面的食材管理功能，涵盖从录入到使用的完整流程：

1. 智能食材管理

- 基础信息管理：支持食材名称、品类分类、存放位置（冷藏/冷冻/常温）、计量单位、数量、有效期等完整信息录入

- 图片管理：支持拍照或从相册选择食材图片，便于视觉识别和记录

- 批量操作：支持批量编辑、删除、移动食材，提高管理效率

- 搜索与筛选：提供多维度搜索功能，可按名称、品类、状态、位置等条件快速定位食材

2. 多样化录入方式

- 手动录入：提供直观的表单界面，支持快速输入食材信息

- 模板快速添加：预设常用食材模板，一键添加减少重复输入

3. 智能状态监控与提醒

- 自动状态计算：基于当前时间与保质期，自动计算食材状态（新鲜/临期/过期）

- 多级提醒策略：支持设置多个提醒时间点（如临期前3天、1天、到期日等）

- 本地通知推送：即使应用未运行也能及时收到提醒通知

- 自定义提醒内容：用户可自定义提醒消息内容，增加个性化体验

4. 数据可视化与分析

- 概览仪表板：直观显示食材总数、临期数量、过期数量等关键指标

- 品类分布图：以饼图、柱状图等形式展示不同品类食材的分布情况

- 时间趋势分析：分析食材使用模式，识别浪费高峰期

- 成本统计：记录食材采购成本，分析支出结构

- 营养分析：基于食材信息分析营养摄入情况

5. AI智能建议系统

- 烹饪建议：基于现有食材智能推荐菜谱，提供详细的制作步骤

- 营养搭配：根据营养学原理，推荐合理的食材搭配方案

- 保鲜技巧：提供针对不同食材的科学保鲜方法和存储建议

- 采购建议：分析使用模式，智能推荐需要补充的食材

- 季节性建议：结合时令特点，推荐应季食材和菜谱

6. 个性化设置中心

- 字典管理：自定义食材品类、计量单位、存放位置等字典内容

- 提醒策略配置：灵活设置不同食材的提醒时间和方式

- 界面主题：支持浅色/深色主题切换，适应不同使用环境

- 多语言支持：支持中文、英文等多种语言界面

- 数据备份与同步：支持数据导出、导入和云端同步功能

## 优势创新

Fridgy 在食材管理领域具有多项技术创新和用户体验优势：

1. 场景化设计理念

- 冰箱场景优化：专门针对家庭冰箱和储物柜的使用场景设计，考虑实际使用中的空间限制和操作习惯

- 轻量化数据模型：采用精简而高效的数据结构，减少存储空间占用，提高应用响应速度

- 高频操作优化：针对用户频繁的查看、编辑、搜索操作进行界面和交互优化，提升使用效率

- 零学习成本：采用直观的界面设计和符合用户习惯的操作流程，新用户无需学习即可上手使用

2. 智能提醒系统

- 多维度提醒策略：支持基于时间、状态、用户行为等多维度的智能提醒

- 个性化提醒设置：用户可根据个人习惯和食材特性自定义提醒时间和方式

- 渐进式提醒：采用渐进式提醒策略，从温和提醒到紧急提醒，避免用户产生抵触情绪

- 智能学习：系统会学习用户的使用习惯，自动调整提醒策略，提高提醒的精准度

3. 可扩展AI架构

- 多模型支持：支持集成多种AI模型（DeepSeek、Zhipu、GPT等），用户可根据需求选择最适合的模型

- 统一接口设计：采用统一的API接口和提示词模板，确保不同AI模型的一致体验

- 离线AI能力：支持本地AI模型，在无网络环境下也能提供基础智能建议

- 模型热切换：支持运行时切换AI模型，无需重启应用即可体验不同AI能力

4. 数据安全与隐私保护

- 本地优先存储：核心数据优先存储在本地，减少隐私泄露风险

- 加密存储：敏感信息采用加密存储，确保数据安全

- 最小权限原则：只请求必要的系统权限，避免过度授权

- 数据控制权：用户完全控制自己的数据，支持导出、删除等操作

# 运行环境

## 硬件要求

Fridgy 对硬件要求相对较低，能够适配大多数主流移动设备：

1. 设备兼容性

- Android设备：支持Android 8.0（API 26）及以上版本的智能手机和平板电脑

- 推荐配置：4GB RAM，64GB存储空间

- 最低配置：2GB RAM，32GB存储空间

- 屏幕分辨率：支持720p及以上分辨率，推荐1080p或更高

- iOS设备：支持iOS 13.0及以上版本的iPhone和iPad

- iPhone：支持iPhone 6s及以上机型

- iPad：支持iPad Air 2及以上机型

- 推荐配置：3GB RAM，64GB存储空间

2. 存储空间要求

- 应用安装：约50-80MB（包含核心功能模块）

- 用户数据：根据使用情况动态增长

- 基础使用：约10-50MB（纯文本数据）

- 图片存储：每张图片约1-5MB，建议预留200-500MB

- 缓存数据：约20-100MB（AI模型缓存、临时文件等）

- 总建议空间：至少预留500MB可用存储空间

3. 网络连接要求

- 基础功能：支持离线使用，无需网络连接

- AI功能：需要稳定的网络连接（Wi-Fi或4G/5G）

- 上传速度：建议≥1Mbps

- 下载速度：建议≥2Mbps

- 数据同步：需要网络连接进行云端备份和同步

- 应用更新：需要网络连接下载应用更新

## 软件要求

Fridgy 的软件环境要求包括操作系统、权限配置和开发工具等方面：

操作系统要求

- Android平台：

- 最低版本：Android 8.0（API Level 26）

- 推荐版本：Android 10.0（API Level 29）及以上

- 支持架构：ARM64-v8a、ARMv7、x86\_64

- 特殊要求：需要支持Google Play Services（用于推送通知）

- iOS平台：

- 最低版本：iOS 13.0

- 推荐版本：iOS 15.0及以上

- 支持架构：ARM64

- 特殊要求：需要支持ARKit（用于增强现实功能，可选）

# 软件技术

## 技术栈

Fridgy 采用现代化的技术栈，确保高性能、可维护性和跨平台兼容性：

1. 前端框架与运行时

- React 19.0.0：最新版本的React框架，提供最新的特性和性能优化

- React Native 0.79.5：跨平台移动应用开发框架，实现一套代码多平台运行

- Expo SDK 53.0.22：提供丰富的原生功能封装和开发工具链

- TypeScript 5.8.3：提供静态类型检查，提高代码质量和开发效率

2. 路由与导航系统

- expo-router 5.1.5：基于文件系统的路由管理，支持嵌套路由和动态路由

- react-native-screens 4.11.1：原生屏幕管理，提升导航性能

- react-native-safe-area-context 5.4.0：安全区域管理，适配各种屏幕尺寸

3. UI组件与交互

- react-native-paper 5.12.0：Material Design风格的UI组件库

- react-native-gesture-handler 2.24.0：高性能手势处理库

- react-native-reanimated 3.17.4：高性能动画库，支持60fps流畅动画

- react-native-svg 15.11.2：矢量图形支持，用于图标和图表渲染

- expo-linear-gradient 14.1.5：渐变背景支持

4. 原生功能模块

- expo-camera 16.1.11：相机功能，支持拍照和扫码

- expo-notifications 0.31.4：本地和远程通知管理

- expo-file-system 18.1.11：文件系统操作，支持文件读写

- expo-image-picker 16.1.4：图片选择器，支持相册和拍照

- expo-device 7.1.4：设备信息获取

- expo-constants 17.1.7：应用常量管理

- expo-secure-store 14.2.4：安全存储，用于敏感数据加密存储

- expo-haptics 14.1.4：触觉反馈支持

- expo-location 18.1.6：地理位置服务

- expo-linking 7.1.7：深度链接支持

- expo-sharing 13.1.5：文件分享功能

5. 状态管理与数据存储

- zustand 4.4.0：轻量级状态管理库，提供简洁的API

- @react-native-async-storage/async-storage 2.1.2：异步存储，用于本地数据持久化

- @react-native-community/datetimepicker 8.4.1：日期时间选择器

- @react-native-community/netinfo 11.4.1：网络状态检测

6. 工具库与辅助功能

- date-fns 2.30.0：现代JavaScript日期处理库

- babel-plugin-module-resolver 5.0.2：模块路径解析

- react-native-web 0.20.0：Web平台支持

7. 开发工具与质量保证

- ESLint 8.0.0：代码质量检查工具

- Prettier 3.0.0：代码格式化工具

- Jest 29.0.0：单元测试框架

- @testing-library/react-native 12.8.0：React Native测试工具

- TypeScript ESLint：TypeScript代码检查

8. 构建与部署

- EAS Build：Expo应用构建服务

- EAS Submit：应用商店提交服务

- Metro：React Native打包工具

- Hermes：JavaScript引擎（Android）

## 架构设计

Fridgy 采用分层架构设计，确保代码的可维护性、可扩展性和高性能：

1. 整体架构模式

- 分层架构：采用经典的分层架构模式，将应用分为表现层、业务逻辑层、数据访问层

- 模块化设计：每个功能模块独立开发，通过接口进行交互

- 组件化开发：UI组件高度复用，减少代码冗余

- 插件化扩展：预留扩展接口，支持功能模块的动态加载

2. 目录结构与职责划分

src/

├── app/ # 应用入口和路由配置

│ ├── \_layout.tsx # 根布局组件

│ ├── index.tsx # 首页

│ └── (tabs)/ # Tab导航页面

│ ├── overview/ # 概览页面

│ ├── statistics/ # 统计页面

│ └── settings/ # 设置页面

├── components/ # 可复用UI组件

│ ├── ui/ # 基础UI组件

│ ├── ingredients/ # 食材相关组件

│ └── ai/ # AI功能组件

├── services/ # 业务服务层

│ ├── ai/ # AI服务适配

│ ├── api/ # API接口服务

│ └── notifications/ # 通知服务

├── store/ # 状态管理

│ ├── ingredients/ # 食材状态

│ └── settings/ # 设置状态

└── utils/ # 工具函数

├── constants/ # 常量定义

├── helpers/ # 辅助函数

└── types/ # 类型定义

3. 表现层（Presentation Layer）

- 页面组件：`src/app/`，基于expo-router的文件系统路由

- 概览页面：展示食材总览和关键指标

- 扫描页面：提供扫码和拍照功能

- 统计页面：数据可视化和分析报告

- 设置页面：用户偏好和应用配置

- UI组件：`src/components/`，高度复用的界面组件

- 基础组件：Button、Card、Badge等通用组件

- 业务组件：IngredientCard、EditModal、AIModal等

- 复合组件：结合多个基础组件的复杂界面

4. 业务逻辑层（Business Logic Layer）

- 服务层：`src/services/`，封装业务逻辑和外部服务

- AI服务：统一AI接口，支持多模型切换

- API服务：网络请求和数据同步

- 通知服务：本地通知调度和管理

- 权限服务：系统权限请求和处理

- 状态管理：`src/store/`，使用Zustand管理应用状态

- 食材状态：食材列表、筛选条件、编辑状态

- 设置状态：用户偏好、提醒策略、主题设置

- 全局状态：网络状态、加载状态、错误信息

5. 数据访问层（Data Access Layer）

- 本地存储：AsyncStorage和SecureStore

- 食材数据：存储在AsyncStorage中

- 敏感信息：使用SecureStore加密存储

- 缓存数据：临时数据和图片缓存

- 网络服务：API调用和数据同步

- RESTful API：标准HTTP接口

- 数据同步：增量同步和冲突解决

- 离线支持：本地优先，网络补充

6. 通知调度系统

- 调度策略：

- 基于食材到期日计算提醒时间

- 支持多级提醒（临期前3天、1天、到期日）

- 用户可自定义提醒时间和方式

- 通知管理：

- 应用启动时重建通知计划

- 数据变更时更新相关通知

- 系统重启后通过引导恢复通知

- 权限处理：

- 渐进式权限请求

- 权限被拒绝时的降级方案

- 用户友好的权限说明

7. AI适配架构

- 统一接口：`services/ai`定义标准接口

- analyze：食材分析接口

- advise：建议生成接口

- 统一的输入输出格式

- 多模型支持：

- DeepSeek：主要AI服务提供商

- Zhipu：备选AI服务提供商

- Mock：离线演示和测试

- 智能切换：

- 根据网络状态自动切换

- 用户可手动选择AI模型

- 支持运行时热切换

8. 异常处理与容错机制

- 权限处理：

- 相机权限：提供手动输入替代方案

- 通知权限：使用应用内提醒作为备选

- 存储权限：提供数据导出功能

- 网络异常：

- 离线模式：核心功能本地可用

- 重试机制：网络恢复后自动重试

- 用户提示：清晰的错误信息和解决建议

- 数据异常：

- 数据校验：输入数据格式验证

- 数据恢复：损坏数据的自动修复

- 备份机制：定期数据备份和恢复

## 数据模型

Fridgy 采用以"食材"为中心的数据模型设计，确保数据的完整性和扩展性：

1. 核心实体模型

Ingredient（食材实体）

```typescript

interface Ingredient {

// 基础标识

id: string; // 唯一标识符（UUID）

name: string; // 食材名称

description?: string; // 食材描述（可选）

// 分类信息

category: string; // 品类分类（蔬菜、肉类、乳制品等）

subcategory?: string; // 子分类（可选）

tags: string[]; // 标签数组（有机、进口、特价等）

// 存储信息

location: string; // 存放位置（冷藏、冷冻、常温、冷冻室上层等）

storageMethod?: string; // 存储方法（密封、通风、避光等）

// 数量与单位

quantity: number; // 数量

unit: string; // 计量单位（个、g、ml、袋、盒等）

originalQuantity?: number; // 原始采购数量（可选）

originalUnit?: string; // 原始单位（可选）

// 时间信息

purchaseDate?: Date; // 采购日期（可选）

expireAt: Date; // 到期/最佳食用日期

openedAt?: Date; // 开封日期（可选）

createdAt: Date; // 创建时间

updatedAt: Date; // 更新时间

// 媒体信息

imageUri?: string; // 食材图片URI

barcode?: string; // 条码信息（可选）

// 状态信息

status: 'fresh' | 'expiring' | 'expired'; // 当前状态

isOpened: boolean; // 是否已开封

isFavorite: boolean; // 是否收藏

// 扩展信息

price?: number; // 价格（可选）

source?: string; // 来源（超市、菜市场、网购等）

notes?: string; // 备注信息

}

```

2. 辅助数据模型

Category（品类模型）

```typescript

interface Category {

id: string;

name: string; // 品类名称

icon: string; // 图标标识

color: string; // 主题颜色

defaultExpiryDays: number; // 默认保质期天数

storageLocation: string; // 默认存储位置

isActive: boolean; // 是否启用

}

```

Unit（单位模型）

```typescript

interface Unit {

id: string;

name: string; // 单位名称

symbol: string; // 单位符号

type: 'weight' | 'volume' | 'count' | 'length'; // 单位类型

conversionFactor?: number; // 转换系数（可选）

}

```

NotificationRule（通知规则模型）

```typescript

interface NotificationRule {

id: string;

name: string; // 规则名称

daysBeforeExpiry: number; // 到期前天数

isEnabled: boolean; // 是否启用

message: string; // 通知消息模板

soundEnabled: boolean; // 是否启用声音

vibrationEnabled: boolean; // 是否启用震动

}

```

3. 状态计算逻辑

状态判定算法

```typescript

function calculateIngredientStatus(ingredient: Ingredient): 'fresh' | 'expiring' | 'expired' {

const today = new Date();

const expireDate = new Date(ingredient.expireAt);

const daysUntilExpiry = Math.ceil((expireDate.getTime() - today.getTime()) / (1000 \* 60 \* 60 \* 24));

if (daysUntilExpiry < 0) {

return 'expired'; // 已过期

} else if (daysUntilExpiry <= 3) { // 3天内到期

return 'expiring'; // 临期

} else {

return 'fresh'; // 新鲜

}

}

```

4. 数据关系设计

一对多关系

- 一个用户可以有多个食材记录

- 一个品类可以对应多个食材

- 一个食材可以有多个通知记录

多对多关系

- 食材与标签：一个食材可以有多个标签，一个标签可以对应多个食材

- 食材与菜谱：一个食材可以用于多个菜谱，一个菜谱可以包含多个食材

5. 数据验证规则

输入验证

- 食材名称：必填，长度1-50字符

- 数量：必填，大于0的数值

- 到期日期：必填，不能早于当前日期

- 品类：必填，必须从预定义列表中选择

- 存储位置：必填，必须从预定义列表中选择

业务规则

- 开封后的食材保质期自动缩短

- 临期食材优先显示在列表顶部

- 过期食材自动标记但保留记录

- 重复食材自动合并或提示

6. 数据索引策略

主要索引

- 按状态索引：快速筛选新鲜/临期/过期食材

- 按品类索引：快速按品类分组显示

- 按到期日期索引：快速排序和提醒

- 按存储位置索引：快速按位置查看

复合索引

- 状态+到期日期：优化临期提醒查询

- 品类+状态：优化统计页面查询

- 用户+创建时间：优化数据同步查询

7. 数据迁移与版本控制

版本管理

- 数据模型版本号：每次结构变更递增

- 向后兼容：新版本支持旧数据格式

- 自动迁移：检测版本差异并自动升级

数据备份

- 增量备份：只备份变更的数据

- 定期全量备份：确保数据安全

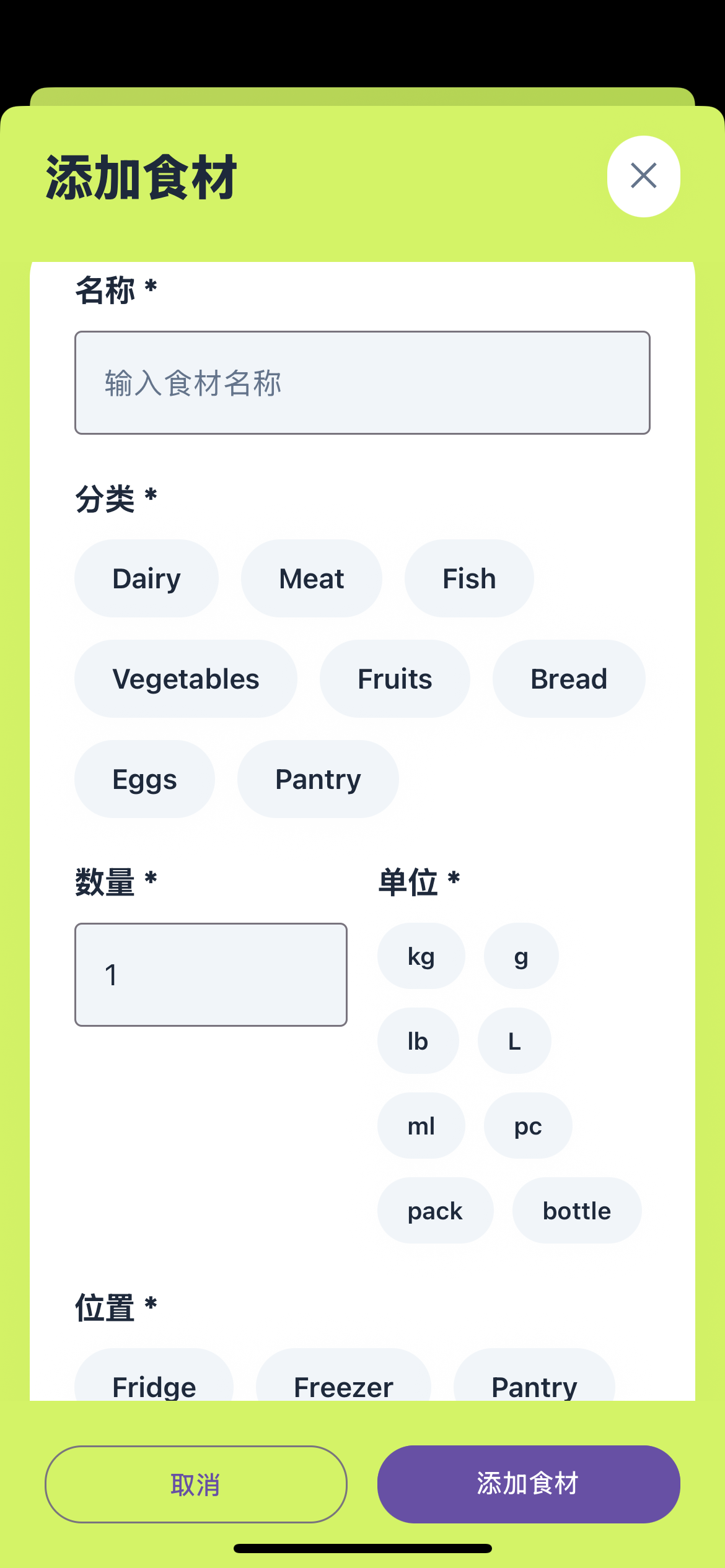
- 云端同步：支持多设备数据同步

## 软件功能

### 食材管理

- 新增/编辑/删除食材；支持品类、单位、位置等字样选择。

- 批量快速添加（快捷模态），减少重复输入。

### 智能状态监控与提醒

- 状态自动计算：

- 实时状态更新：基于当前时间自动计算食材状态

- 多状态支持：新鲜、临期、过期三种状态

- 视觉标识：不同颜色和图标区分食材状态

- 状态历史：记录状态变化历史，便于分析

- 智能提醒系统：

- 多级提醒：支持设置多个提醒时间点

- 个性化策略：不同食材可设置不同的提醒策略

- 本地通知：使用expo-notifications实现本地推送

- 提醒历史：记录提醒发送历史，避免重复提醒

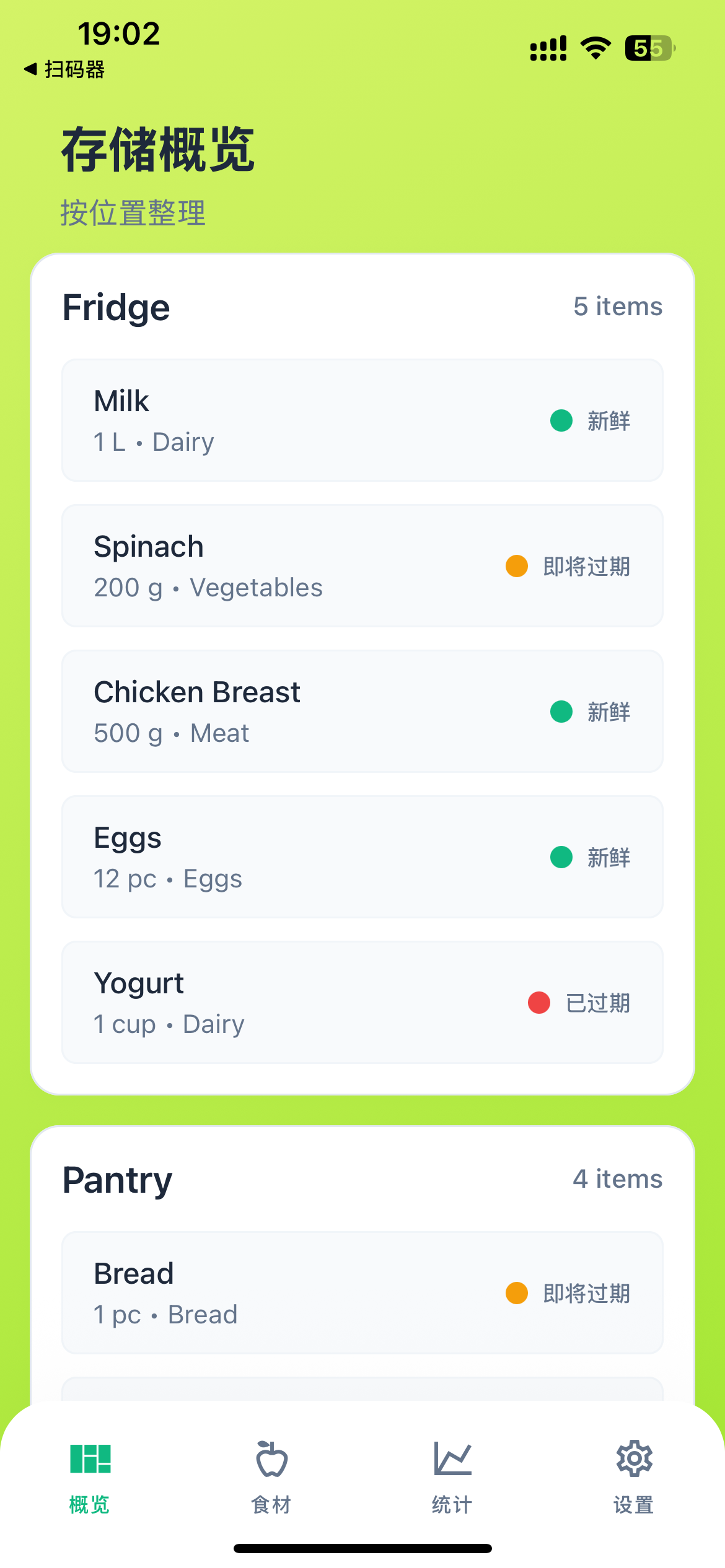
- 概览与统计：

- 仪表板：直观显示关键指标和统计数据

- 图表展示：饼图、柱状图、趋势图等多种图表

- 分类统计：按品类、状态、位置等维度统计

- 趋势分析：分析食材使用和浪费趋势

### AI智能建议系统

烹饪建议：

- 菜谱推荐：基于现有食材推荐合适菜谱

- 营养搭配：根据营养学原理推荐搭配方案

- 制作步骤：提供详细的制作步骤和技巧

- 难度评估：标注菜谱制作难度和时间

- 保鲜建议：

- 存储建议：针对不同食材提供最佳存储方法

- 保鲜技巧：提供延长食材保质期的实用技巧

- 处理建议：临期食材的处理和利用建议

- 季节性建议：结合时令特点提供建议

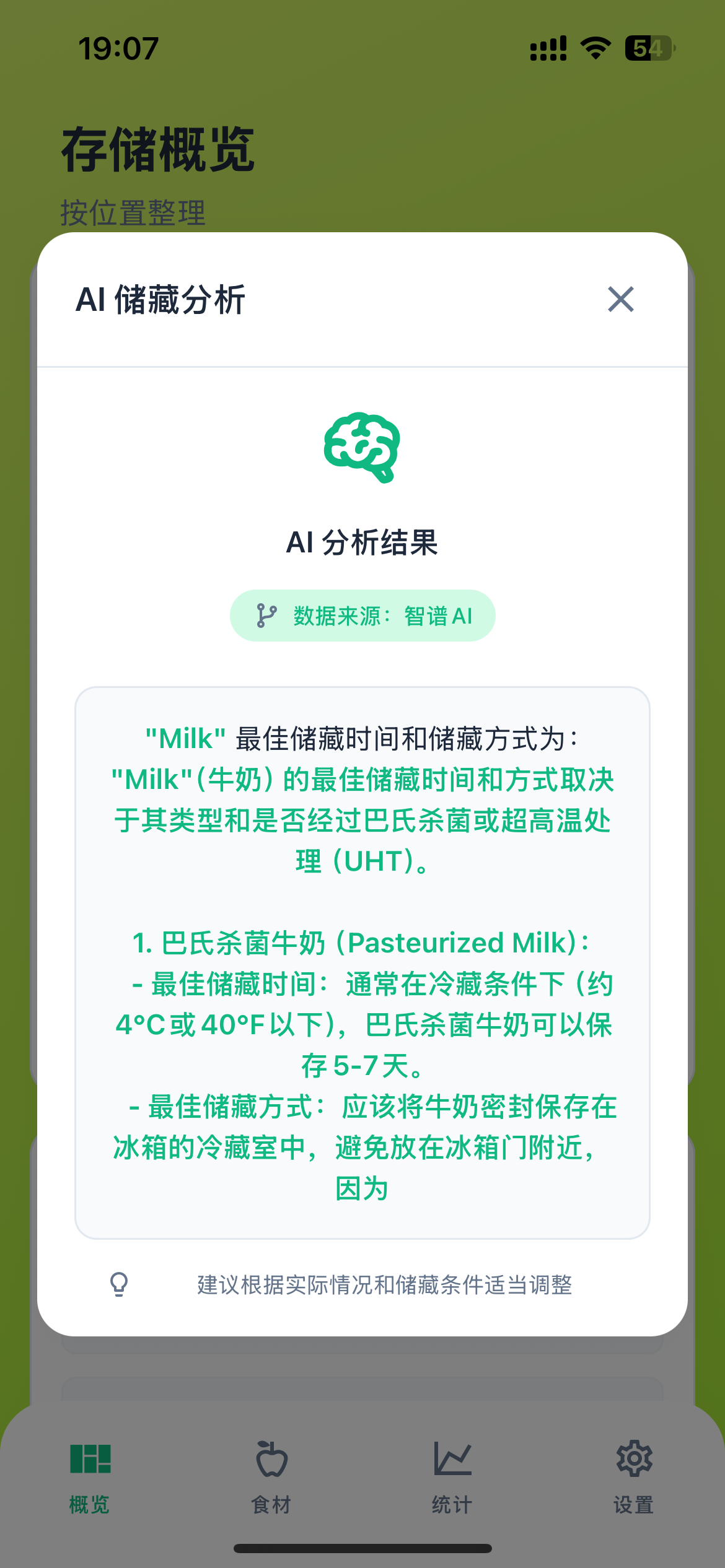
- 采购建议：

- 库存分析：分析当前库存，推荐需要补充的食材

- 采购清单：生成智能采购清单，避免遗漏

- 价格比较：提供价格参考和购买建议

- 季节性采购：推荐应季食材和最佳采购时机

### 个性化设置与配置

字典管理：

- 品类管理：自定义食材品类，支持添加、编辑、删除

- 单位管理：管理计量单位，支持单位转换

- 位置管理：自定义存储位置，适应不同家庭环境

- 标签管理：创建和管理自定义标签

- 提醒策略配置：

- 默认策略：设置默认的提醒时间和方式

- 个性化策略：为特定食材设置专门的提醒策略

- 提醒模板：创建和编辑提醒消息模板

- 免打扰时间：设置免打扰时间段

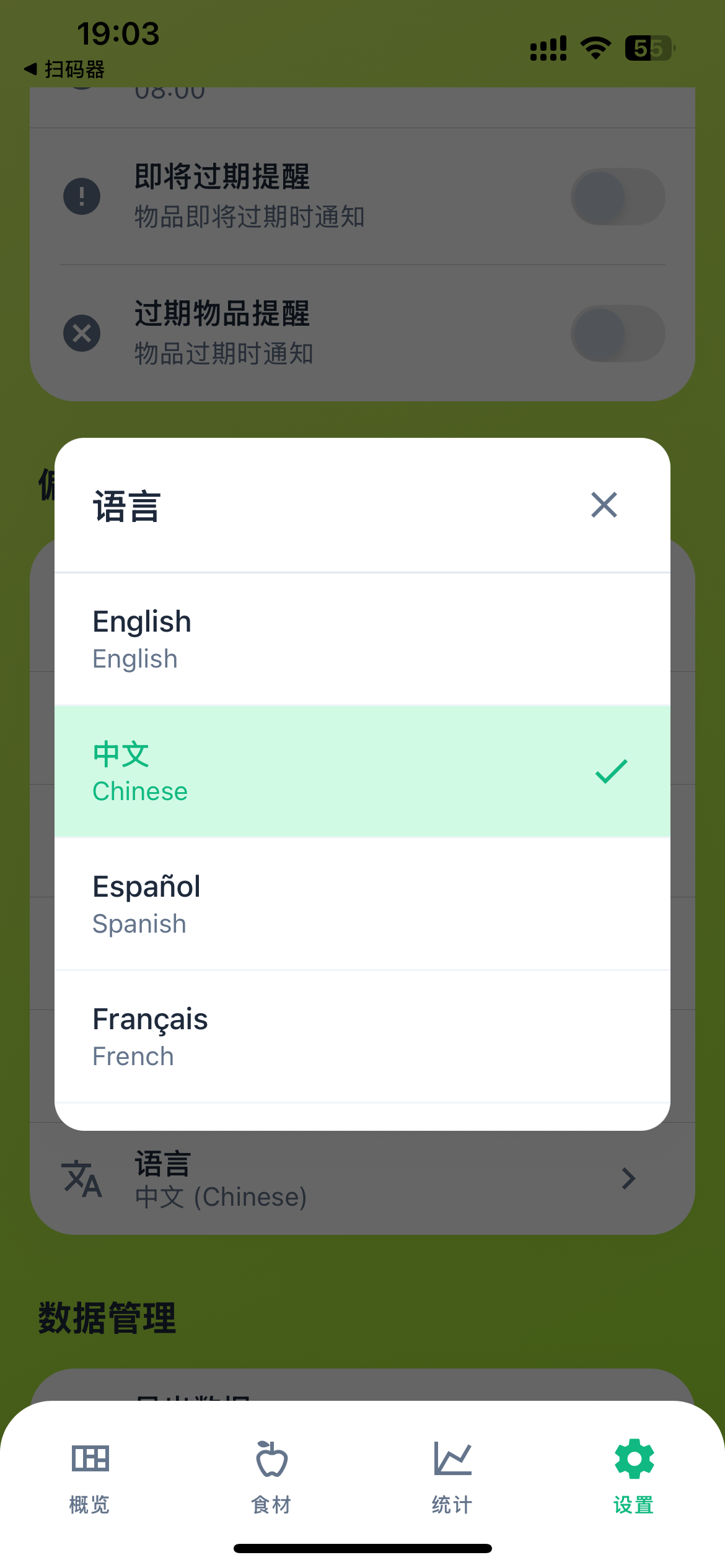
- 界面与主题：

- 主题切换：支持浅色、深色主题

- 字体大小：可调节字体大小，适应不同用户需求

- 布局设置：自定义界面布局和显示选项

- 多语言支持：支持中文、英文等多种语言



### 数据管理与安全

本地存储：

- AsyncStorage：存储食材数据和用户设置

- SecureStore：加密存储敏感信息如API密钥

- 数据压缩：自动压缩存储数据，节省空间

- 数据清理：定期清理临时文件和缓存

- 数据同步：

- 云端备份：支持数据云端备份和恢复

- 多设备同步：多设备间数据实时同步

- 冲突解决：智能处理数据冲突

- 离线支持：离线模式下正常使用核心功能

- 数据导出：

- 格式支持：支持导出为Excel、CSV、JSON等格式

- 选择性导出：可选择导出特定时间段或类型的数据

- 数据统计：导出包含统计信息的报告

- 分享功能：支持通过邮件、社交平台分享数据



### 分析功能

- 数据分析：

- 使用模式分析：分析用户的食材使用模式

- 浪费分析：统计和分析食材浪费情况

- 成本分析：计算和分析食材采购成本

- 营养分析：分析营养摄入情况

