

菜单自动翻译ios app

1. 项目概述与目标

项目的主要目的

开发一款iOS应用，帮助国际旅行者在不熟悉当地语言的情况下，通过拍照上传菜单，自动翻译并格式化成双语菜单，方便点餐。同时，提供真实的菜品图片，用户可上传和查看他人上传的菜品照片，提升点餐的准确性和体验。

目标用户群体

- **国际旅行者**：在国外旅行时需要翻译和理解本地菜单的用户。
- **餐厅顾客**：希望更好地了解菜单内容，选择合适菜品的顾客。
- **美食爱好者**：喜欢探索和分享不同菜品图片的用户。

预期成果

- **用户数量**：在MVP阶段目标获取5000活跃用户。
 - **收入目标**：通过广告、餐厅合作或高级功能订阅实现初步盈利。
 - **市场份额**：在主要旅游城市占据应用市场的10%份额。
-

2. 核心功能定义

MVP阶段必须具备的功能

1. 用户注册与登录

- 支持Google登录、Apple登录。
- 密码重置功能。

2. 菜单上传与翻译

- 用户拍照上传餐厅菜单。
- 自动OCR识别文字并翻译成双语（原文 + 目标语言）。

3. 电子版格式化菜单

- 将翻译后的菜单以电子版格式展示，便于点菜。

4. 菜品图片展示与上传

- 展示前人上传的真实菜品图片。
- 允许用户上传自己拍摄的菜品照片。

5. 餐厅与菜单管理

- 存储餐厅信息及其对应菜单。
- 允许用户搜索和浏览不同餐厅的菜单。

6. 用户反馈与评分

- 用户可以对翻译质量、菜品图片进行评分和反馈。

参考产品

- **OpenRice** 和 **Meituan**：虽然它们提供菜单和图片，但在翻译和用户上传菜品图片方面存在不足，可以作为改进的参考。
-

3. 技术栈与所需包

前端

- 语言与框架：SwiftUI
- 主要库与工具：
 - **Alamofire**：用于网络请求。
 - **Kingfisher**：用于图片下载和缓存。
 - **Combine**：响应式编程框架。
 - **SwiftLint**：代码规范和质量检查。

后端

- 语言与框架：Python (使用FastAPI)
- 主要库与工具：

- **FastAPI**：用于构建高性能API。
- **SQLAlchemy**：数据库ORM。
- **Pydantic**：数据验证。
- **Celery**：处理异步任务（如OCR和翻译）。
- **Redis**：作为Celery的消息中间件。

数据库与服务

- **数据库**：Supabase (PostgreSQL)
- **OCR与翻译服务**：
 - **AWS Textract** 或 **Google Cloud Vision**：用于OCR。
 - **Google Translate API** 或 **DeepL API**：用于翻译。

其他工具

- **版本控制**：Git + GitHub/GitLab
- **CI/CD**：GitHub Actions 或 GitLab CI
- **容器化**：Docker（用于开发和部署环境一致性）
- **监控与日志**：Sentry（错误监控）、Prometheus & Grafana（性能监控）

4. 项目文件夹结构

前端 (iOS - SwiftUI)

```

/YourApp
|
├── /YourApp
|   ├── AppDelegate.swift
|   ├── SceneDelegate.swift
|   ├── ContentView.swift
|   ├── Assets.xcassets
|   ├── Info.plist
|   └── /Views

```

```
| | | └─ LoginView.swift
| | | └─ MenuUploadView.swift
| | | └─ MenuDisplayView.swift
| | | └─ ...
| | └─ /ViewModels
| | | └─ AuthViewModel.swift
| | | └─ MenuViewModel.swift
| | | └─ ...
| | └─ /Models
| | | └─ User.swift
| | | └─ Menu.swift
| | | └─ ...
| | └─ /Services
| | | └─ APIService.swift
| | | └─ ImageService.swift
| | | └─ ...
| | └─ /Resources
| | | └─ Colors.swift
| | | └─ Fonts.swift
| | | └─ ...
| | └─ /Helpers
| |   └─ Extensions.swift
|
└─ /Tests
    └─ YourAppTests.swift
    └─ ...
└─ /YourApp.xcworkspace
```

后端 (Python - FastAPI)

```
/backend
|
└─ /app
    └─ main.py
```

```

├── /api
│   ├── __init__.py
│   ├── auth.py
│   ├── menus.py
│   └── ...
├── /models
│   ├── __init__.py
│   ├── user.py
│   ├── menu.py
│   └── ...
├── /schemas
│   ├── __init__.py
│   ├── user.py
│   ├── menu.py
│   └── ...
├── /services
│   ├── __init__.py
│   ├── ocr_service.py
│   ├── translation_service.py
│   └── ...
├── /core
│   ├── __init__.py
│   ├── config.py
│   └── security.py
├── /db
│   ├── __init__.py
│   ├── base.py
│   ├── session.py
│   └── ...
├── /workers
│   ├── __init__.py
│   ├── tasks.py
│   └── ...
├── /tests
│   └── test_auth.py

```

```
|   |─ test_menus.py
|   └─ ...
|
|─ Dockerfile
|─ docker-compose.yml
|─ requirements.txt
└─ README.md
```

公共

```
/docs
|─ API_Documentation.md
|─ Database_Schema.md
└─ ...

/scripts
|─ setup.sh
|─ deploy.sh
└─ ...

/configs
|─ .env
|─ config.yaml
└─ ...
```

5. 数据库设计

基于您的需求，我们将使用Supabase（PostgreSQL）来设计数据库。以下是推荐的表结构：

表：Users

- **id:** UUID (Primary Key)
- **name:** VARCHAR

- **email:** VARCHAR (Unique)
- **password_hash:** VARCHAR (如果支持密码登录)
- **created_at:** TIMESTAMP
- **updated_at:** TIMESTAMP
- **social_provider:** VARCHAR (如 Google, Apple)
- **social_id:** VARCHAR (社交登录的唯一标识)

表：Restaurants

- **id:** UUID (Primary Key)
- **name:** VARCHAR
- **address:** VARCHAR
- **phone_number:** VARCHAR
- **latitude:** DECIMAL
- **longitude:** DECIMAL
- **created_at:** TIMESTAMP
- **updated_at:** TIMESTAMP

表：Menus

- **id:** UUID (Primary Key)
- **restaurant_id:** UUID (Foreign Key → Restaurants.id)
- **original_image_url:** VARCHAR (菜单原图的存储路径)
- **translated_image_url:** VARCHAR (翻译后菜单图的存储路径)
- **formatted_menu_json:** JSONB (格式化后的菜单数据)
- **created_at:** TIMESTAMP
- **updated_at:** TIMESTAMP

表：Dishes

- **id:** UUID (Primary Key)

- **menu_id**: UUID (Foreign Key → Menus.id)
- **name_original**: VARCHAR
- **name_translated**: VARCHAR
- **description_original**: TEXT
- **description_translated**: TEXT
- **price**: DECIMAL
- **created_at**: TIMESTAMP
- **updated_at**: TIMESTAMP

表 : DishImages

- **id**: UUID (Primary Key)
- **dish_id**: UUID (Foreign Key → Dishes.id)
- **image_url**: VARCHAR
- **uploaded_by**: UUID (Foreign Key → Users.id)
- **created_at**: TIMESTAMP

表 : Translations

- **id**: UUID (Primary Key)
- **menu_id**: UUID (Foreign Key → Menus.id)
- **language**: VARCHAR
- **translated_text**: TEXT
- **created_at**: TIMESTAMP

表 : UserFeedback

- **id**: UUID (Primary Key)
- **user_id**: UUID (Foreign Key → Users.id)
- **menu_id**: UUID (Foreign Key → Menus.id)
- **rating**: INTEGER (1-5)

- **comments:** TEXT
- **created_at:** TIMESTAMP

关系图

- **Users ↔ DishImages:** 一对多
- **Restaurants ↔ Menus:** 一对多
- **Menus ↔ Dishes:** 一对多
- **Dishes ↔ DishImages:** 一对多
- **Menus ↔ Translations:** 一对多
- **Users ↔ UserFeedback:** 一对多
- **Menus ↔ UserFeedback:** 一对多

建议

- **索引：**在经常查询的字段上建立索引，如 `Restaurants.name`、`Menus.restaurant_id`、`Dishes.menu_id` 等，以提高查询效率。
- **外键约束：**确保数据的完整性和一致性，设置适当的外键约束。
- **存储优化：**对于图片存储，建议使用外部存储服务（如 AWS S3 或 Google Cloud Storage），并在数据库中存储其URL。

6. 登录页面组件

功能需求

1. 登录选项

- **Google 登录：**利用 OAuth 2.0 实现。
- **Apple 登录：**利用 Sign in with Apple 功能。
- **邮箱密码登录**（如果需要）。

2. 密码重置

- 用户可以通过邮箱请求密码重置链接。

3. 用户注册

- 新用户可以通过社交登录或邮箱注册。

4. 错误处理与提示

- 登录失败、网络错误等提示信息。

设计风格与布局

- **简约灵动**：使用简洁的布局，清晰的按钮和输入框。
- **视觉层次**：通过颜色和排版区分不同的部分，如登录选项、注册选项等。
- **响应式设计**：适配不同iOS设备尺寸。

组件结构 (SwiftUI)

```
struct LoginView: View {
    @State private var email: String = ""
    @State private var password: String = ""
    @State private var showingPasswordReset: Bool = false
    @State private var errorMessage: String?

    var body: some View {
        VStack(spacing: 20) {
            Text("欢迎使用双语菜单")
                .font(.largeTitle)
                .fontWeight(.bold)

            // Google 登录按钮
            Button(action: {
                // Google 登录逻辑
            }) {
                HStack {
                    Image("google_icon")
                    Text("使用 Google 登录")
                        .fontWeight(.medium)
                }
            }
        }
    }
}
```

```

        .frame(maxWidth: .infinity)
        .padding()
        .background(Color.white)
        .cornerRadius(8)
        .shadow(radius: 2)
    }

    // Apple 登录按钮
    Button(action: {
        // Apple 登录逻辑
    }) {
        HStack {
            Image("apple_icon")
            Text("使用 Apple 登录")
                .fontWeight(.medium)
        }
        .frame(maxWidth: .infinity)
        .padding()
        .background(Color.black)
        .foregroundColor(.white)
        .cornerRadius(8)
        .shadow(radius: 2)
    }

    // 或者使用邮箱登录
    HStack {
        Rectangle()
            .frame(height: 1)
        Text("或者使用邮箱")
        Rectangle()
            .frame(height: 1)
    }.padding(.vertical, 10)

    // 邮箱输入
    TextField("邮箱", text: $email)
        .padding()

```

```

        .background(Color(.secondarySystemBackground
d))
        .cornerRadius(8)
        .keyboardType(.emailAddress)
        .autocapitalization(.none)

// 密码输入
SecureField("密码", text: $password)
        .padding()
        .background(Color(.secondarySystemBackground
d))
        .cornerRadius(8)

// 登录按钮
Button(action: {
    // 邮箱登录逻辑
}) {
    Text("登录")
        .fontWeight(.bold)
        .foregroundColor(.white)
        .frame(maxWidth: .infinity)
        .padding()
        .background(Color.blue)
        .cornerRadius(8)
}

// 密码重置
Button(action: {
    showingPasswordReset.toggle()
}) {
    Text("忘记密码?")
        .foregroundColor(.blue)
}
.sheet(isPresented: $showingPasswordReset) {
    PasswordResetView()
}

```

```

        // 错误信息显示
        if let error = errorMessage {
            Text(error)
                .foregroundColor(.red)
                .padding()
        }

        Spacer()
    }
    .padding()
}
}

```

用户体验优化

- **加载指示器**：在登录过程中显示加载指示器，提升用户体验。
- **表单验证**：即时验证邮箱格式和密码强度，减少用户错误。
- **辅助功能**：支持VoiceOver等iOS辅助功能，提高可访问性。

7. 色彩调色板

设计理念

简约且灵动，采用柔和且对比度适中的颜色，提升用户的视觉体验，同时确保内容的可读性。

建议的色彩调色板

Aa 用途	≡ 颜色代码	≡ 颜色预览	📎 Files
<u>主色调</u>	#4A90E2		https://via.placeholder.com/50/4A90E2/FFFFFF?text=+
<u>辅助色</u>	#50E3C2		https://via.placeholder.com/50/50E3C2/FFFFFF?text=+

Aa 用途	≡ 颜色代码	≡ 颜色预览	📎 Files
			<u>text=+</u>
背景色	#F5F5F5		<u>https://via.placeholder.com/50/F5F5F5/FFFFFF?text=+</u>
标题和文字色	#333333		<u>https://via.placeholder.com/50/333333/FFFFFF?text=+</u>
次要文字色	#666666		<u>https://via.placeholder.com/50/666666/FFFFFF?text=+</u>
错误提示色	#FF3B30		<u>https://via.placeholder.com/50/FF3B30/FFFFFF?text=+</u>
成功提示色	#4CD964		<u>https://via.placeholder.com/50/4CD964/FFFFFF?text=+</u>
按钮颜色	#007AFF		<u>https://via.placeholder.com/50/007AFF/FFFFFF?text=+</u>

具体应用

- **主色调**：用于主要按钮、链接和高亮元素。
- **辅助色**：用于次要按钮、图标和强调元素。
- **背景色**：应用的主要背景色，确保内容的可读性。
- **标题和文字色**：用于主要文本，如标题和重要信息。
- **次要文字色**：用于次要文本，如说明和辅助信息。
- **错误提示色**：用于错误消息和警告。
- **成功提示色**：用于成功消息和确认。
- **按钮颜色**：统一的按钮颜色，确保用户操作的一致性。

示例代码 (SwiftUI)

```
import SwiftUI
```

```

struct Colors {
    static let primary = Color(hex: "#4A90E2")
    static let secondary = Color(hex: "#50E3C2")
    static let background = Color(hex: "#F5F5F5")
    static let textPrimary = Color(hex: "#333333")
    static let textSecondary = Color(hex: "#666666")
    static let error = Color(hex: "#FF3B30")
    static let success = Color(hex: "#4CD964")
    static let button = Color(hex: "#007AFF")
}

extension Color {
    init(hex: String) {
        let hex = hex.trimmingCharacters(in: CharacterSet.alphanumerics.inverted)
        var int: UInt64 = 0
        Scanner(string: hex).scanHexInt64(&int)
        let a, r, g, b: UInt64
        switch hex.count {
            case 3: // RGB (12-bit)
                (a, r, g, b) = (255, (int >> 8) * 17, (int >> 4 &
0xF) * 17, (int & 0xF) * 17)
            case 6: // RGB (24-bit)
                (a, r, g, b) = (255, int >> 16, int >> 8 & 0xFF,
int & 0xFF)
            case 8: // ARGB (32-bit)
                (a, r, g, b) = (int >> 24, int >> 16 & 0xFF, int
>> 8 & 0xFF, int & 0xFF)
            default:
                (a, r, g, b) = (255, 0, 0, 0)
        }
        self.init(
            .sRGB,
            red: Double(r) / 255,
            green: Double(g) / 255,
            blue: Double(b) / 255,

```

```
        opacity: Double(a) / 255
    )
}
}
```

8. 文案撰写

设计理念

文案应贴近用户，传达产品的真实性和实用性，让用户感受到产品能够真正解决他们的痛点。

主要页面文案示例

登录页面

- 欢迎标题： 欢迎使用双语菜单
- Google 登录按钮： 使用 Google 登录
- Apple 登录按钮： 使用 Apple 登录
- 邮箱登录按钮： 登录
- 邮箱占位符： 请输入您的邮箱
- 密码占位符： 请输入您的密码
- 忘记密码链接： 忘记密码？
- 错误提示： 登录失败，请检查您的邮箱和密码。

注册页面

- 欢迎标题： 创建您的账号
- Google 注册按钮： 使用 Google 注册
- Apple 注册按钮： 使用 Apple 注册
- 邮箱注册按钮： 注册
- 邮箱占位符： 请输入您的邮箱

- 密码占位符： 创建一个密码
- 确认密码占位符： 确认您的密码
- 错误提示： 注册失败，请检查信息并重试。

菜单上传页面

- 标题： 上传菜单
- 说明： 拍照上传餐厅菜单，我们将为您自动翻译并生成双语菜单。
- 上传按钮： 选择照片
- 翻译按钮： 开始翻译
- 成功提示： 菜单上传并翻译成功！

菜单展示页面

- 标题： 双语菜单
- 说明： 以下是餐厅的双语菜单，您可以直接点菜或查看真实的菜品图片。
- 点菜按钮： 立即点菜
- 上传菜品图片按钮： 上传菜品照片
- 错误提示： 无法加载菜单，请稍后再试。

菜品详情页面

- 菜品名称： 原文名称 / 翻译名称
- 价格： \$价格
- 描述： 原文描述 / 翻译描述
- 真实图片标题： 真实用户上传的菜品图片
- 上传按钮： 上传您的菜品照片
- 成功提示： 感谢您的上传！

密码重置页面

- 标题： 重置密码

- **说明：** 请输入您的邮箱，我们将发送密码重置链接。
- **邮箱占位符：** 请输入您的邮箱
- **发送按钮：** 发送
- **成功提示：** 密码重置链接已发送到您的邮箱。
- **错误提示：** 发送失败，请检查您的邮箱地址。

通用按钮与提示

- **确认按钮：** 确认
- **取消按钮：** 取消
- **加载提示：** 正在加载...
- **无数据提示：** 暂无数据

用户引导与帮助文案



- **首次使用引导：** 欢迎使用双语菜单！通过拍照上传菜单，轻松获取翻译和点菜建议，享受无忧的美食体验。
- **上传菜单提示：** 确保菜单清晰可见，避免反光或模糊，以提高翻译准确性。
- **上传菜品图片提示：** 请上传真实的菜品照片，帮助其他用户更好地了解菜品。

营销文案

- **App Store 描述：**

双语菜单 - 让国际旅行更加轻松！只需拍照上传菜单，自动翻译并生成双语电子菜单，助您轻松点餐。不仅如此，您还可以查看和上传真实的菜品图片，确保每一餐都符合您的口味。无论您身处何地，双语菜单都是您的美食好帮手！

- **社交媒体推广文案：**

 旅行中遇到不懂的菜单？别担心！使用双语菜单应用，拍照上传菜单，轻松翻译，真实菜品图片助您精准点餐。现在就下载，开启无障碍的美食之旅！
 #双语菜单 #旅行必备 #美食无国界

9. MenuPal API 文档

目录

1. 用户认证
 - 登录
 - 注册
 - 密码重置
 - 登出
 2. 用户管理
 3. 餐厅管理
 4. 菜单管理
 5. 菜品管理
 6. 菜品图片管理
 7. 上传管理
 8. 用户反馈
 9. 搜索功能
 10. 通用错误响应
-

用户认证

1. 登录

1.1 使用 Google 登录

- 端点: `/auth/login/google`
- 方法: `POST`
- 描述: 通过Google OAuth 2.0进行用户登录。

- 请求参数:

- **Body:**

```
json
复制代码
{
  "token": "string" // Google OAuth 2.0 访问令牌
}
```

- 响应:

- **成功 (200 OK):**

```
json
复制代码
{
  "access_token": "string",
  "refresh_token": "string",
  "user": {
    "id": "uuid",
    "name": "string",
    "email": "string",
    "created_at": "timestamp"
  }
}
```

- **失败 (400 Bad Request):**

```
json
复制代码
{
  "error": "Invalid Google token."
```

```
}
```

1.2 使用 Apple 登录

- 端点: `/auth/login/apple`
- 方法: `POST`
- 描述: 通过Apple ID进行用户登录。
- 请求参数:

- **Body:**

```
json
复制代码
{
  "identity_token": "string" // Apple ID 身份令牌
}
```

- 响应:

- **成功 (200 OK):**

```
json
复制代码
{
  "access_token": "string",
  "refresh_token": "string",
  "user": {
    "id": "uuid",
    "name": "string",
    "email": "string",
    "created_at": "timestamp"
  }
}
```

```
}
```

- **失败 (400 Bad Request):**

```
json
复制代码
{
  "error": "Invalid Apple token."
}
```

1.3 使用邮箱密码登录

- **端点:** `/auth/login`
- **方法:** `POST`
- **描述:** 通过邮箱和密码进行用户登录。
- **请求参数:**

- **Body:**

```
json
复制代码
{
  "email": "string",
  "password": "string"
}
```

- **响应:**

- **成功 (200 OK):**

```
json
复制代码
{
```

```
"access_token": "string",
"refresh_token": "string",
"user": {
  "id": "uuid",
  "name": "string",
  "email": "string",
  "created_at": "timestamp"
}
```

- 失败 (401 Unauthorized):

```
json
复制代码
{
  "error": "Invalid email or password."
}
```

2. 注册

2.1 使用邮箱注册

- 端点: /auth/register
- 方法: POST
- 描述: 通过邮箱和密码注册新用户。
- 请求参数:

- Body:

```
json
复制代码
{
  "name": "string",
```

```
"email": "string",
"password": "string"
}
```

- 响应:

- 成功 (`201 Created`):

```
json
复制代码
{
  "message": "User registered successfully.",
  "user": {
    "id": "uuid",
    "name": "string",
    "email": "string",
    "created_at": "timestamp"
  }
}
```

- 失败 (`400 Bad Request`):

```
json
复制代码
{
  "error": "Email already exists."
}
```

3. 密码重置

3.1 请求密码重置

- 端点: `/auth/reset-password`

- **方法:** `POST`
- **描述:** 通过邮箱请求密码重置链接。
- **请求参数:**

- **Body:**

```
json
复制代码
{
  "email": "string"
}
```

- **响应:**

- **成功 (`200 OK`):**

```
json
复制代码
{
  "message": "Password reset link sent to your email."
}
```

- **失败 (`404 Not Found`):**

```
json
复制代码
{
  "error": "Email not found."
}
```

3.2 执行密码重置

- **端点:** `/auth/reset-password/confirm`
- **方法:** `POST`
- **描述:** 使用重置链接中的token重置密码。
- **请求参数:**

- **Body:**

```
json
复制代码
{
  "token": "string",
  "new_password": "string"
}
```

- **响应:**

- **成功 (`200 OK`):**

```
json
复制代码
{
  "message": "Password has been reset successfully."
}
```

- **失败 (`400 Bad Request`):**

```
json
复制代码
{
  "error": "Invalid or expired token."
}
```

4. 登出

- 端点: `/auth/logout`
- 方法: `POST`
- 描述: 用户登出，失效当前会话。
- 请求参数:
 - **Headers:**
 - `Authorization`: `Bearer <access_token>`
- 响应:
 - **成功 (`200 OK`):**

```
json
复制代码
{
  "message": "Logged out successfully."
}
```

- **失败 (`401 Unauthorized`):**

```
json
复制代码
{
  "error": "Invalid or missing token."
}
```

用户管理

获取用户信息

- 端点: `/users/{user_id}`

- **方法:** GET
- **描述:** 获取指定用户的详细信息。
- **请求参数:**
 - **路径参数:**
 - `user_id` (UUID): 用户的唯一标识符。
 - **Headers:**
 - `Authorization`: Bearer <access_token>
- **响应:**
 - **成功 (200 OK):**

```
json
复制代码
{
  "id": "uuid",
  "name": "string",
  "email": "string",
  "created_at": "timestamp",
  "updated_at": "timestamp"
}
```

- **失败 (404 Not Found):**

```
json
复制代码
{
  "error": "User not found."
}
```

更新用户信息

- **端点:** `/users/{user_id}`
- **方法:** `PUT`
- **描述:** 更新指定用户的详细信息。
- **请求参数:**

- **路径参数:**

- `user_id` (UUID): 用户的唯一标识符。

- **Headers:**

- `Authorization`: `Bearer <access_token>`

- **Body:**

```
json
复制代码
{
  "name": "string",
  "email": "string"
}
```

- **响应:**

- **成功 (`200 OK`):**

```
json
复制代码
{
  "message": "User updated successfully.",
  "user": {
    "id": "uuid",
    "name": "string",
    "email": "string",
    "updated_at": "timestamp"
  }
}
```

```
}
```

- 失败 (400 Bad Request):

```
json
复制代码
{
  "error": "Invalid input data."
}
```

餐厅管理

获取所有餐厅

- 端点: `/restaurants`
- 方法: `GET`
- 描述: 获取所有餐厅的列表。
- 请求参数:
 - 查询参数 (可选):
 - `name`: `string` - 按名称搜索餐厅。
 - `location`: `string` - 按地理位置搜索餐厅。
 - `page`: `integer` - 分页页码。
 - `limit`: `integer` - 每页条目数。
- 响应:
 - 成功 (200 OK):

```
json
复制代码
{
```

```
"restaurants": [  
  {  
    "id": "uuid",  
    "name": "string",  
    "address": "string",  
    "phone_number": "string",  
    "latitude": "decimal",  
    "longitude": "decimal",  
    "created_at": "timestamp",  
    "updated_at": "timestamp"  
  },  
  ...  
],  
"pagination": {  
  "current_page": "integer",  
  "total_pages": "integer",  
  "total_items": "integer"  
}  
}
```

创建新餐厅

- 端点: `/restaurants`
- 方法: `POST`
- 描述: 创建一个新的餐厅记录。
- 请求参数:
 - Headers:
 - `Authorization`: `Bearer <access_token>`
 - Body:

json
复制代码

```
{
  "name": "string",
  "address": "string",
  "phone_number": "string",
  "latitude": "decimal",
  "longitude": "decimal",
  "website": "string" // 可选
}
```

◦ 响应:

▪ 成功 (201 Created):

```
json
复制代码
{
  "message": "Restaurant created successfully.",
  "restaurant": {
    "id": "uuid",
    "name": "string",
    "address": "string",
    "phone_number": "string",
    "latitude": "decimal",
    "longitude": "decimal",
    "website": "string",
    "created_at": "timestamp",
    "updated_at": "timestamp"
  }
}
```

▪ 失败 (400 Bad Request):

```
json
复制代码
```



```
{
  "error": "Invalid input data."
}
```

获取单个餐厅信息

- 端点: `/restaurants/{restaurant_id}`
- 方法: `GET`
- 描述: 获取指定餐厅的详细信息。
- 请求参数:
 - 路径参数:
 - `restaurant_id` (UUID): 餐厅的唯一标识符。
- 响应:
 - 成功 (`200 OK`):

```
json
复制代码
{
  "id": "uuid",
  "name": "string",
  "address": "string",
  "phone_number": "string",
  "latitude": "decimal",
  "longitude": "decimal",
  "website": "string",
  "created_at": "timestamp",
  "updated_at": "timestamp"
}
```

- 失败 (`404 Not Found`):

```
json
复制代码
{
  "error": "Restaurant not found."
}
```

更新餐厅信息

- **端点:** `/restaurants/{restaurant_id}`
- **方法:** `PUT`
- **描述:** 更新指定餐厅的详细信息。
- **请求参数:**
 - **路径参数:**
 - `restaurant_id` (UUID): 餐厅的唯一标识符。
 - **Headers:**
 - `Authorization`: `Bearer <access_token>`
 - **Body:**

```
json
复制代码
{
  "name": "string",
  "address": "string",
  "phone_number": "string",
  "latitude": "decimal",
  "longitude": "decimal",
  "website": "string" // 可选
}
```

- **响应:**

▪ **成功 (200 OK):**

```
json
复制代码
{
  "message": "Restaurant updated successfully.",
  "restaurant": {
    "id": "uuid",
    "name": "string",
    "address": "string",
    "phone_number": "string",
    "latitude": "decimal",
    "longitude": "decimal",
    "website": "string",
    "updated_at": "timestamp"
  }
}
```

▪ **失败 (400 Bad Request):**

```
json
复制代码
{
  "error": "Invalid input data."
}
```

删除餐厅

- **端点:** `/restaurants/{restaurant_id}`
- **方法:** `DELETE`
- **描述:** 删除指定餐厅及其相关联的菜单和菜品。
- **请求参数:**

- **路径参数:**

- `restaurant_id` (UUID): 餐厅的唯一标识符。

- **Headers:**

- `Authorization`: `Bearer <access_token>`

- **响应:**

- **成功 (`200 OK`):**

```
json
复制代码
{
  "message": "Restaurant and associated menus and dishes deleted successfully."
}
```

- **失败 (`404 Not Found`):**

```
json
复制代码
{
  "error": "Restaurant not found."
}
```

菜单管理

获取餐厅的所有菜单

- **端点:** `/restaurants/{restaurant_id}/menus`
- **方法:** `GET`
- **描述:** 获取指定餐厅的所有菜单。
- **请求参数:**

- **路径参数:**

- `restaurant_id` (UUID): 餐厅的唯一标识符。

- **查询参数 (可选):**

- `language`: `string` - 按语言筛选菜单。
- `page`: `integer` - 分页页码。
- `limit`: `integer` - 每页条目数。

- **响应:**

- **成功 (200 OK):**

```
json
复制代码
{
  "menus": [
    {
      "id": "uuid",
      "restaurant_id": "uuid",
      "language": "string",
      "image_url": "string",
      "created_at": "timestamp",
      "updated_at": "timestamp"
    },
    ...
  ],
  "pagination": {
    "current_page": "integer",
    "total_pages": "integer",
    "total_items": "integer"
  }
}
```

创建新菜单

- **端点:** `/restaurants/{restaurant_id}/menus`
- **方法:** `POST`
- **描述:** 为指定餐厅创建一个新菜单。
- **请求参数:**
 - **路径参数:**
 - `restaurant_id` (UUID): 餐厅的唯一标识符。

- **Headers:**

- `Authorization`: `Bearer <access_token>`

- **Body:**

```
json
复制代码
{
  "language": "string",
  "image_url": "string" // 菜单图片的URL
}
```

- **响应:**

- **成功 (`201 Created`):**

```
json
复制代码
{
  "message": "Menu created successfully.",
  "menu": {
    "id": "uuid",
    "restaurant_id": "uuid",
    "language": "string",
    "image_url": "string",
    "created_at": "timestamp",
    "updated_at": "timestamp"
  }
}
```

```
}  
}
```

- **失败 (400 Bad Request):**

```
json  
复制代码  
{  
  "error": "Invalid input data."  
}
```

获取单个菜单信息

- **端点:** `/menus/{menu_id}`
- **方法:** `GET`
- **描述:** 获取指定菜单的详细信息。
- **请求参数:**
 - **路径参数:**
 - `menu_id` (UUID): 菜单的唯一标识符。
- **响应:**
 - **成功 (200 OK):**

```
json  
复制代码  
{  
  "id": "uuid",  
  "restaurant_id": "uuid",  
  "language": "string",  
  "image_url": "string",  
  "created_at": "timestamp",  
  "updated_at": "timestamp"
```

```
}
```

■ **失败 (404 Not Found):**

```
json
复制代码
{
  "error": "Menu not found."
}
```

更新菜单信息

- **端点:** `/menus/{menu_id}`
- **方法:** `PUT`
- **描述:** 更新指定菜单的详细信息。
- **请求参数:**
 - **路径参数:**
 - `menu_id` (UUID): 菜单的唯一标识符。
 - **Headers:**
 - `Authorization`: `Bearer <access_token>`
 - **Body:**

```
json
复制代码
{
  "language": "string",
  "image_url": "string" // 菜单图片的URL
}
```


- 响应:

- 成功 (200 OK):

```
json
复制代码
{
  "message": "Menu updated successfully.",
  "menu": {
    "id": "uuid",
    "restaurant_id": "uuid",
    "language": "string",
    "image_url": "string",
    "updated_at": "timestamp"
  }
}
```

- 失败 (400 Bad Request):

```
json
复制代码
{
  "error": "Invalid input data."
}
```

删除菜单

- 端点: `/menus/{menu_id}`
- 方法: `DELETE`
- 描述: 删除指定菜单及其相关联的菜品和图片。
- 请求参数:
 - 路径参数:

- `menu_id` (UUID): 菜单的唯一标识符。
- **Headers:**
 - `Authorization`: `Bearer <access_token>`
- **响应:**
 - **成功 (`200 OK`):**

```
json
复制代码
{
  "message": "Menu and associated dishes deleted successfully."
}
```

- **失败 (`404 Not Found`):**

```
json
复制代码
{
  "error": "Menu not found."
}
```

菜单翻译

- **端点:** `/menus/{menu_id}/translate`
- **方法:** `POST`
- **描述:** 为指定菜单创建翻译版本。
- **请求参数:**
 - **路径参数:**
 - `menu_id` (UUID): 菜单的唯一标识符。
 - **Headers:**

- **Authorization**: **Bearer** <access_token>

- **Body:**

```
json
复制代码
{
  "target_language": "string" // 目标翻译语言
}
```

- **响应:**

- **成功 (200 OK):**

```
json
复制代码
{
  "message": "Menu translated successfully.",
  "translated_menu": {
    "id": "uuid",
    "menu_id": "uuid",
    "language": "string",
    "translated_text": "string",
    "created_at": "timestamp"
  }
}
```

- **失败 (400 Bad Request):**

```
json
复制代码
{
  "error": "Translation failed."
```

```
}
```

菜品管理

获取菜单的所有菜品

- 端点: `/menus/{menu_id}/dishes`
- 方法: `GET`
- 描述: 获取指定菜单的所有菜品。
- 请求参数:
 - 路径参数:
 - `menu_id` (UUID): 菜单的唯一标识符。
 - 查询参数 (可选):
 - `category`: `string` - 按类别筛选菜品。
 - `page`: `integer` - 分页页码。
 - `limit`: `integer` - 每页条目数。
- 响应:
 - 成功 (`200 OK`):

```
json
复制代码
{
  "dishes": [
    {
      "id": "uuid",
      "menu_id": "uuid",
      "name_original": "string",
      "name_translated": "string",
      "description_original": "string",
      "description_translated": "string",
```

```

        "price": "decimal",
        "category": "string",
        "created_at": "timestamp",
        "updated_at": "timestamp"
    },
    ...
],
"pagination": {
    "current_page": "integer",
    "total_pages": "integer",
    "total_items": "integer"
}
}

```

创建新菜品

- 端点: `/menus/{menu_id}/dishes`
- 方法: `POST`
- 描述: 为指定菜单创建一个新菜品。
- 请求参数:
 - 路径参数:
 - `menu_id` (UUID): 菜单的唯一标识符。
 - Headers:
 - `Authorization`: `Bearer <access_token>`
 - Body:

```

json
复制代码
{
    "name_original": "string",
    "name_translated": "string",

```

```
"description_original": "string",
"description_translated": "string",
"price": "decimal",
"category": "string"
}
```

◦ 响应:

▪ 成功 (201 Created):

```
json
复制代码
{
  "message": "Dish created successfully.",
  "dish": {
    "id": "uuid",
    "menu_id": "uuid",
    "name_original": "string",
    "name_translated": "string",
    "description_original": "string",
    "description_translated": "string",
    "price": "decimal",
    "category": "string",
    "created_at": "timestamp",
    "updated_at": "timestamp"
  }
}
```

▪ 失败 (400 Bad Request):

```
json
复制代码
{
  "error": "Invalid input data."
}
```

```
}
```

获取单个菜品信息

- 端点: `/dishes/{dish_id}`
- 方法: `GET`
- 描述: 获取指定菜品的详细信息。
- 请求参数:
 - 路径参数:
 - `dish_id` (UUID): 菜品的唯一标识符。
- 响应:
 - 成功 (`200 OK`):

```
json
复制代码
{
  "id": "uuid",
  "menu_id": "uuid",
  "name_original": "string",
  "name_translated": "string",
  "description_original": "string",
  "description_translated": "string",
  "price": "decimal",
  "category": "string",
  "created_at": "timestamp",
  "updated_at": "timestamp"
}
```

- 失败 (`404 Not Found`):

```
json
复制代码
{
  "error": "Dish not found."
}
```

更新菜品信息

- 端点: `/dishes/{dish_id}`
- 方法: `PUT`
- 描述: 更新指定菜品的详细信息。
- 请求参数:
 - 路径参数:
 - `dish_id` (UUID): 菜品的唯一标识符。
 - Headers:
 - `Authorization`: `Bearer <access_token>`
 - Body:

```
json
复制代码
{
  "name_original": "string",
  "name_translated": "string",
  "description_original": "string",
  "description_translated": "string",
  "price": "decimal",
  "category": "string"
}
```

- 响应:

▪ **成功** (`200 OK`):

```
json
复制代码
{
  "message": "Dish updated successfully.",
  "dish": {
    "id": "uuid",
    "menu_id": "uuid",
    "name_original": "string",
    "name_translated": "string",
    "description_original": "string",
    "description_translated": "string",
    "price": "decimal",
    "category": "string",
    "updated_at": "timestamp"
  }
}
```

▪ **失败** (`400 Bad Request`):

```
json
复制代码
{
  "error": "Invalid input data."
}
```

删除菜品

- **端点:** `/dishes/{dish_id}`
- **方法:** `DELETE`
- **描述:** 删除指定菜品及其相关联的图片。

- 请求参数:
 - 路径参数:
 - `dish_id` (UUID): 菜品的唯一标识符。
 - Headers:
 - `Authorization`: `Bearer <access_token>`

- 响应:
 - 成功 (`200 OK`):

```
json
复制代码
{
  "message": "Dish and associated images deleted successfully."
}
```

- 失败 (`404 Not Found`):

```
json
复制代码
{
  "error": "Dish not found."
}
```

菜品图片管理

获取菜品的所有图片

- 端点: `/dishes/{dish_id}/images`
- 方法: `GET`
- 描述: 获取指定菜品的所有图片。

◦ 请求参数:

▪ 路径参数:

- `dish_id` (UUID): 菜品的唯一标识符。

▪ 查询参数 (可选):

- `uploaded_by`: `uuid` - 按上传者筛选图片。
- `page`: `integer` - 分页页码。
- `limit`: `integer` - 每页条目数。

◦ 响应:

▪ 成功 (200 OK):

```
json
复制代码
{
  "images": [
    {
      "id": "uuid",
      "dish_id": "uuid",
      "image_url": "string",
      "uploaded_by": "uuid",
      "uploaded_at": "timestamp",
      "image_size": "integer" // 以字节为单位
    },
    ...
  ],
  "pagination": {
    "current_page": "integer",
    "total_pages": "integer",
    "total_items": "integer"
  }
}
```

上传菜品图片

- 端点: `/dishes/{dish_id}/images`
- 方法: `POST`
- 描述: 上传一张新的菜品图片。
- 请求参数:
 - 路径参数:
 - `dish_id` (UUID): 菜品的唯一标识符。
 - Headers:
 - `Authorization`: `Bearer <access_token>`
 - Body:

```
json
复制代码
{
  "image_url": "string", // 图片存储后的URL
  "image_size": "integer" // 图片大小, 单位字节
}
```

- 响应:
 - 成功 (`201 Created`):

```
json
复制代码
{
  "message": "Dish image uploaded successfully.",
  "image": {
    "id": "uuid",
    "dish_id": "uuid",
    "image_url": "string",
    "uploaded_by": "uuid",
  }
}
```

```
    "uploaded_at": "timestamp",
    "image_size": "integer"
  }
}
```

- **失败 (400 Bad Request):**

```
json
复制代码
{
  "error": "Invalid image data."
}
```

获取单个菜品图片

- **端点:** `/images/{image_id}`
- **方法:** `GET`
- **描述:** 获取指定图片的详细信息。
- **请求参数:**
 - **路径参数:**
 - `image_id` (UUID): 图片的唯一标识符。
- **响应:**
 - **成功 (200 OK):**

```
json
复制代码
{
  "id": "uuid",
  "dish_id": "uuid",
  "image_url": "string",
  "uploaded_by": "uuid",
```

```
"uploaded_at": "timestamp",
"image_size": "integer"
}
```

■ 失败 (404 Not Found):

```
json
复制代码
{
  "error": "Image not found."
}
```

删除菜品图片

- 端点: `/images/{image_id}`
- 方法: `DELETE`
- 描述: 删除指定的菜品图片。
- 请求参数:
 - 路径参数:
 - `image_id` (UUID): 图片的唯一标识符。
 - Headers:
 - `Authorization`: `Bearer <access_token>`
- 响应:
 - 成功 (200 OK):

```
json
复制代码
{
  "message": "Dish image deleted successfully."
```

```
}
```

- 失败 (`404 Not Found`):

```
json
复制代码
{
  "error": "Image not found."
}
```

上传管理

上传菜单图片

- 端点: `/uploads/menus`
- 方法: `POST`
- 描述: 上传餐厅菜单的图片，进行OCR和翻译处理。
- 请求参数:
 - Headers:
 - `Authorization`: `Bearer <access_token>`
 - `Content-Type`: `multipart/form-data`
 - Body:
 - `restaurant_id` (UUID): 餐厅的唯一标识符。
 - `image` (file): 菜单的图片文件。
 - `language` (string): 菜单的原始语言。
- 响应:
 - 成功 (`201 Created`):

```
json
复制代码
{
  "message": "Menu uploaded and processed successfully.",
  "menu": {
    "id": "uuid",
    "restaurant_id": "uuid",
    "language": "string",
    "image_url": "string",
    "created_at": "timestamp",
    "updated_at": "timestamp"
  },
  "dishes": [
    {
      "id": "uuid",
      "menu_id": "uuid",
      "name_original": "string",
      "name_translated": "string",
      "description_original": "string",
      "description_translated": "string",
      "price": "decimal",
      "category": "string",
      "created_at": "timestamp",
      "updated_at": "timestamp"
    },
    ...
  ]
}
```

■ 失败 (400 Bad Request):

```
json
复制代码
```



```
{
  "error": "Invalid image format or missing data."
}
```

上传菜品图片

- 端点: `/uploads/dishes`
- 方法: `POST`
- 描述: 上传菜品的图片，关联到指定菜品。
- 请求参数:
 - Headers:
 - `Authorization`: `Bearer <access_token>`
 - `Content-Type`: `multipart/form-data`
 - Body:
 - `dish_id` (UUID): 菜品的唯一标识符。
 - `image` (file): 菜品的图片文件。
- 响应:
 - 成功 (`201 Created`):

```
json
复制代码
{
  "message": "Dish image uploaded successfully.",
  "image": {
    "id": "uuid",
    "dish_id": "uuid",
    "image_url": "string",
    "uploaded_by": "uuid",
    "uploaded_at": "timestamp",
    "image_size": "integer"
  }
}
```

```
}  
}
```

- 失败 (400 Bad Request):

```
json  
复制代码  
{  
  "error": "Invalid image format or missing data."  
}
```

用户反馈

提交用户反馈

- 端点: /feedback
- 方法: POST
- 描述: 用户提交对菜单、菜品或应用的反馈和评分。
- 请求参数:

- Headers:

- Authorization : Bearer <access_token>

- Body:

```
json  
复制代码  
{  
  "menu_id": "uuid", // 可选  
  "dish_id": "uuid", // 可选  
  "rating": "integer", // 1-5  
  "comments": "string" // 可选
```

```
}
```

- 响应:

- 成功 (201 Created):

```
json
复制代码
{
  "message": "Feedback submitted successfully.",
  "feedback": {
    "id": "uuid",
    "user_id": "uuid",
    "menu_id": "uuid",
    "dish_id": "uuid",
    "rating": "integer",
    "comments": "string",
    "created_at": "timestamp"
  }
}
```

- 失败 (400 Bad Request):

```
json
复制代码
{
  "error": "Invalid input data."
}
```

获取用户反馈

- 端点: /feedback
- 方法: GET

◦ **描述:** 获取所有用户反馈，支持筛选和分页。

◦ **请求参数:**

▪ **查询参数 (可选):**

- `menu_id`: `uuid` - 按菜单筛选反馈。
- `dish_id`: `uuid` - 按菜品筛选反馈。
- `user_id`: `uuid` - 按用户筛选反馈。
- `rating`: `integer` - 按评分筛选反馈。
- `page`: `integer` - 分页页码。
- `limit`: `integer` - 每页条目数。

◦ **响应:**

▪ **成功 (200 OK):**

```
json
复制代码
{
  "feedbacks": [
    {
      "id": "uuid",
      "user_id": "uuid",
      "menu_id": "uuid",
      "dish_id": "uuid",
      "rating": "integer",
      "comments": "string",
      "created_at": "timestamp"
    },
    ...
  ],
  "pagination": {
    "current_page": "integer",
    "total_pages": "integer",
    "total_items": "integer"
  }
}
```

```
}
```

获取单个反馈

- 端点: `/feedback/{feedback_id}`
- 方法: `GET`
- 描述: 获取指定的用户反馈。
- 请求参数:
 - 路径参数:
 - `feedback_id` (UUID): 反馈的唯一标识符。
- 响应:
 - 成功 (`200 OK`):

```
json
复制代码
{
  "id": "uuid",
  "user_id": "uuid",
  "menu_id": "uuid",
  "dish_id": "uuid",
  "rating": "integer",
  "comments": "string",
  "created_at": "timestamp"
}
```

- 失败 (`404 Not Found`):

```
json
复制代码
{
  "error": "Feedback not found."
}
```

```
}
```

更新用户反馈

- 端点: `/feedback/{feedback_id}`
- 方法: `PUT`
- 描述: 更新指定的用户反馈。
- 请求参数:
 - 路径参数:
 - `feedback_id` (UUID): 反馈的唯一标识符。
 - Headers:
 - `Authorization`: `Bearer <access_token>`
 - Body:

```
json
复制代码
{
  "rating": "integer", // 可选
  "comments": "string" // 可选
}
```

- 响应:
 - 成功 (`200 OK`):

```
json
复制代码
{
  "message": "Feedback updated successfully.",
  "feedback": {
    "id": "uuid",
```

```
    "user_id": "uuid",
    "menu_id": "uuid",
    "dish_id": "uuid",
    "rating": "integer",
    "comments": "string",
    "updated_at": "timestamp"
  }
}
```

- **失败 (400 Bad Request):**

```
json
复制代码
{
  "error": "Invalid input data."
}
```

删除用户反馈

- **端点:** `/feedback/{feedback_id}`
- **方法:** `DELETE`
- **描述:** 删除指定的用户反馈。
- **请求参数:**
 - **路径参数:**
 - `feedback_id` (UUID): 反馈的唯一标识符。
 - **Headers:**
 - `Authorization`: `Bearer <access_token>`
- **响应:**
 - **成功 (200 OK):**

```
json
复制代码
{
  "message": "Feedback deleted successfully."
}
```

- 失败 (404 Not Found):

```
json
复制代码
{
  "error": "Feedback not found."
}
```

搜索功能

搜索餐厅和菜单

- 端点: `/search`
- 方法: `GET`
- 描述: 根据关键词搜索餐厅和菜单。
- 请求参数:
 - 查询参数:
 - `query`: `string` - 搜索关键词。
 - `type`: `string` - 搜索类型, 可选值: `restaurant`, `menu`, `dish`。
 - `page`: `integer` - 分页页码。
 - `limit`: `integer` - 每页条目数。
- 响应:

▪ 成功 (200 OK):

```
json
复制代码
{
  "results": {
    "restaurants": [
      {
        "id": "uuid",
        "name": "string",
        "address": "string",
        "phone_number": "string",
        "latitude": "decimal",
        "longitude": "decimal",
        "created_at": "timestamp",
        "updated_at": "timestamp"
      },
      ...
    ],
    "menus": [
      {
        "id": "uuid",
        "restaurant_id": "uuid",
        "language": "string",
        "image_url": "string",
        "created_at": "timestamp",
        "updated_at": "timestamp"
      },
      ...
    ],
    "dishes": [
      {
        "id": "uuid",
        "menu_id": "uuid",
        "name_original": "string",
        "name_translated": "string",
```

```
        "description_original": "string",
        "description_translated": "string",
        "price": "decimal",
        "category": "string",
        "created_at": "timestamp",
        "updated_at": "timestamp"
    },
    ...
]
},
"pagination": {
    "current_page": "integer",
    "total_pages": "integer",
    "total_items": "integer"
}
}
```

- 失败 (`400 Bad Request`):

```
json
复制代码
{
    "error": "Invalid search parameters."
}
```

通用错误响应

所有API端点在失败时应返回一致的错误格式，以便前端能够统一处理和展示错误信息。

- 结构:

```
json
复制代码
{
  "error": "string" // 错误描述
}
```

◦ 示例:

▪ 认证错误 (401 Unauthorized):

```
json
复制代码
{
  "error": "Invalid or missing token."
}
```

▪ 资源未找到 (404 Not Found):

```
json
复制代码
{
  "error": "Resource not found."
}
```

▪ 服务器错误 (500 Internal Server Error):

```
json
复制代码
{
  "error": "An unexpected error occurred. Please try again later."
}
```

```
}
```

附录

数据模型简述

用户 (User)

- 字段:

- `id` (UUID)
- `name` (String)
- `email` (String, Unique)
- `password_hash` (String)
- `created_at` (Timestamp)
- `updated_at` (Timestamp)

餐厅 (Restaurant)

- 字段:

- `id` (UUID)
- `name` (String)
- `address` (String)
- `phone_number` (String)
- `latitude` (Decimal)
- `longitude` (Decimal)
- `website` (String, Optional)
- `created_at` (Timestamp)
- `updated_at` (Timestamp)

菜单 (Menu)

- 字段:

- `id` (UUID)
- `restaurant_id` (UUID, Foreign Key)
- `language` (String)
- `image_url` (String)
- `created_at` (Timestamp)
- `updated_at` (Timestamp)

菜品 (Dish)

- 字段:

- `id` (UUID)
- `menu_id` (UUID, Foreign Key)
- `name_original` (String)
- `name_translated` (String)
- `description_original` (String)
- `description_translated` (String)
- `price` (Decimal)
- `category` (String)
- `created_at` (Timestamp)
- `updated_at` (Timestamp)

菜品图片 (DishImage)

- 字段:

- `id` (UUID)
- `dish_id` (UUID, Foreign Key)
- `image_url` (String)

- `uploaded_by` (UUID, Foreign Key)
- `uploaded_at` (Timestamp)
- `image_size` (Integer)

用户反馈 (Feedback)

◦ 字段:

- `id` (UUID)
- `user_id` (UUID, Foreign Key)
- `menu_id` (UUID, Foreign Key, Optional)
- `dish_id` (UUID, Foreign Key, Optional)
- `rating` (Integer, 1-5)
- `comments` (String, Optional)
- `created_at` (Timestamp)

身份验证

所有需要身份验证的端点应在请求头中包含 `Authorization` 字段，格式为 `Bearer <access_token>`。后台应验证令牌的有效性，并确保用户具有执行该操作的权限。

分页

对于返回大量数据的端点，建议使用分页机制，通过 `page` 和 `limit` 查询参数来控制返回的数据量，提升性能和用户体验。

数据验证

所有API端点应进行严格的数据验证，确保接收到的参数符合预期格式和约束，防止数据异常和安全漏洞。