企业级编辑器类飞书文档架构设计与实践（Vanilla、React18、Nestjs）项目介绍

需求背景

项目概述

本项目旨在打造一个完整与飞书文档类似、功能齐全的企业级编辑器。此编辑器集成了所有基础编辑功能，并支持协同编辑，使用 **tiptap** 作为核心编辑框架，并基于 **yjs** 实现实时**协同**功能。

该项目采用全栈架构设计，旨在提供丰富的编辑体验，支持**多端（网页端、桌面端）**和多功能模块（**UI 套件、SDK** 等）。其架构分为基础 SDK、React 适配 SDK、UI 套件、服务端、网页应用和桌面应用

业务需求

背景需求：在企业级环境中，文档编辑和协同办公的需求不断增加，市场中已有的工具如飞书文档、Notion 等应用广泛，但仍存在企业用户在功能扩展、数据安全及个性化需求上的痛点。本项目旨在带领同学们深入研究类飞书文档产品，从而填补这些需求空白，为企业提供一个可扩展、易集成、且支持私有化部署的编辑器解决方案。

核心业务需求：

- 提供完善的文本**编辑**功能（富文本编辑、图片/文件嵌入、格式化工具等）
  - 工具条，toolbar
  - 建议菜单，suggestionMenu
- 实时**协同**编辑，确保多用户无缝协作
- 支持**评论**和**批注**，满足团队沟通需求
- 文档引用关系可视化，文档引用**关系图谱**
- 可扩展的**插件**系统，支持自定义功能
- **私有化部署**与数据安全，符合企业内部信息安全要求

技术需求

- **编辑器核心基础 SDK**：基于 tiptap 的富文本编辑器，提供基础文本编辑、格式化、插入等功能
- **协同编辑模块**：使用 yjs 进行实时同步，支持多用户协同编辑功能
- **UI 套件**：基于 tailwind 的 shadcn UI 套件，提供一致的视觉体验
- **服务端**：负责协同编辑数据同步、用户权限管理和存储功能基础功能，负责其他核心业务数据持久化
- **网页和桌面应用**：多端支持，提供一致的使用体验

竞品分析

竞品分析概述

目前市面上文档类项目非常多，我们列举主要分析一些具有代表性的产品，包括：

- 飞书文档
- Notion
- Appflowy
- wolai
- 钉钉文档

竞品功能对比

- **飞书文档**：集成了团队协同、权限管理等强大功能，但在自定义扩展性方面较弱，并且无法支持页面导出等
- **Notion**：具有灵活的数据库和文档管理功能，但数据私有化部署支持不够完善，AI 功能费用较高
- **Appflowy**：开源文档平台，支持自定义扩展，适合中小型团队，但缺乏企业级应用的扩展性和稳定性
- **wolai**：适合个人和小团队使用，轻量且灵活，但缺乏协同编辑功能，纯抄的 Notion，没什么创新，已经被钉钉收购
- **钉钉文档**：以团队协作为核心，功能较强大，但界面设计不够友好且自定义空间有限

项目价值

满足企业级定制需求

本项目通过提供高度可扩展的插件系统和可私有化部署的能力，为企业用户解决现有市场工具无法满足的个性化需求。企业可以根据自身业务场景定制功能，例如添加专属模板、内嵌业务流程工具等，提升办公效率与信息化水平。

增强数据安全性

通过支持私有化部署，本项目为企业用户提供数据安全保障。相比云端协同办公工具，私有化部署能够避免数据泄露风险，满足企业对敏感信息的合规管理需求，尤其适合金融、医疗和政府等对数据安全要求极高的行业。

提升协同办公效率

通过基于 yjs 实现的实时协同功能，用户可以在多终端、多用户环境下进行高效的文档编辑和协作。同时，评论、批注、引用关系图谱等功能为团队协作提供了全方位的支持，大幅降低沟通成本。

赋能企业数字化转型

本项目不仅是一个编辑器工具，更是企业数字化转型的重要基础模块。通过构建基于全栈架构的企业级编辑器，企业可以将其无缝集成至自身的数字化办公平台，形成内部知识库、协作系统等核心应用，为数字化转型提供支撑。

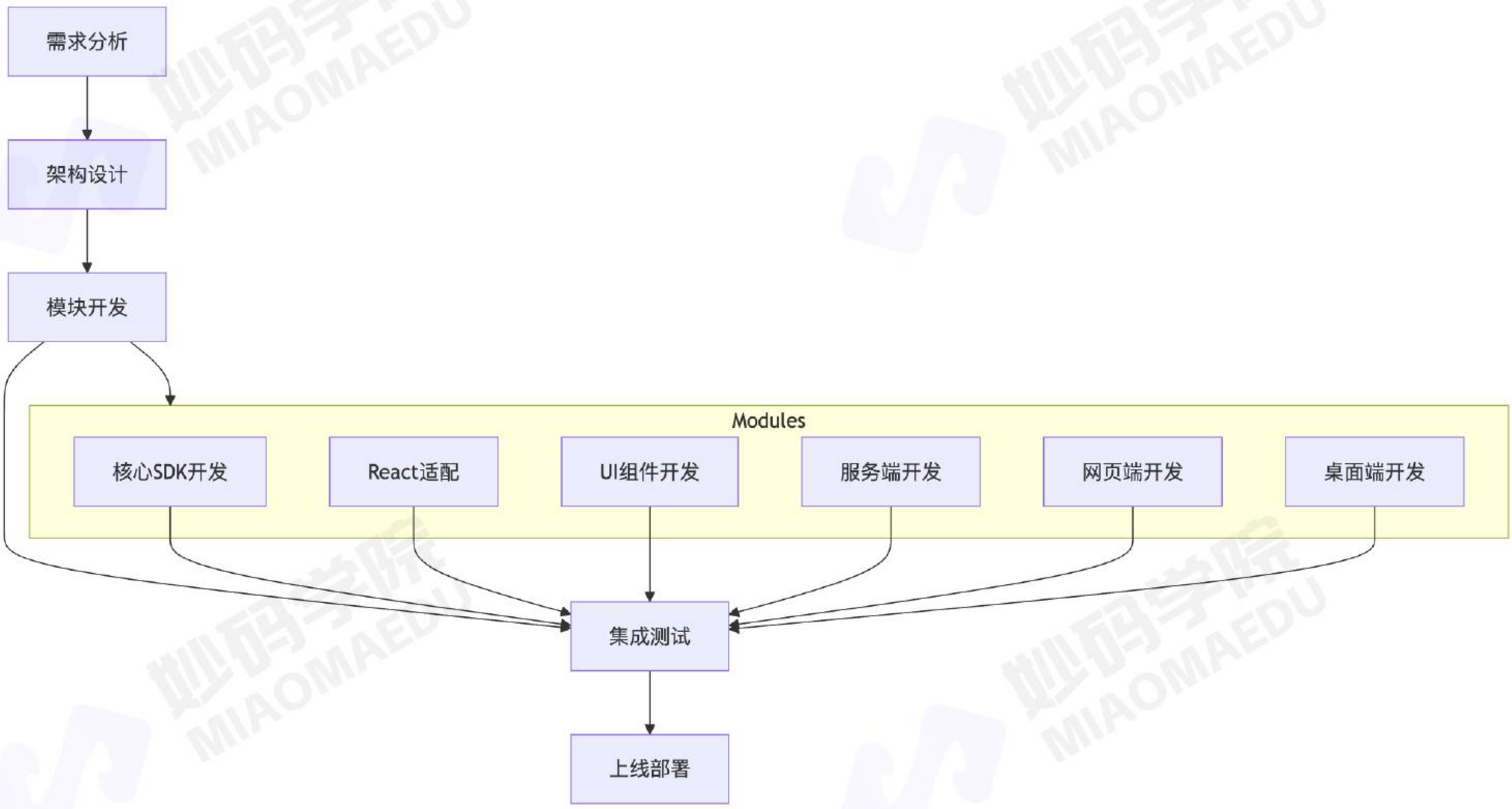
市场竞争力与商业潜力

本项目基于竞品的痛点分析进行了功能优化，例如提供更友好的用户界面、更灵活的插件扩展功能、更稳定的协同机制，以及完善的私有化部署方案。其综合能力将使其成为企业用户的重要选择，具备良好的市场竞争力与商业化潜力。

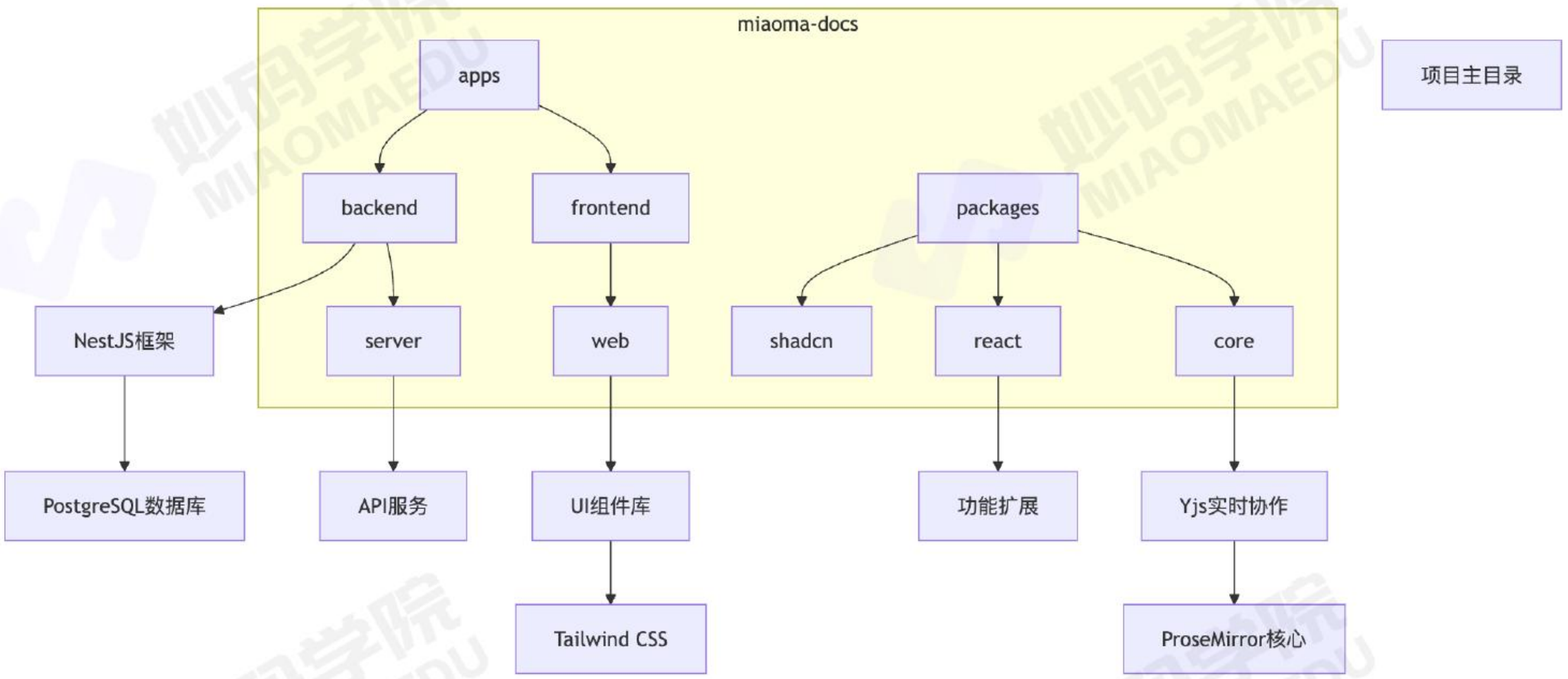
项目架构设计

图解

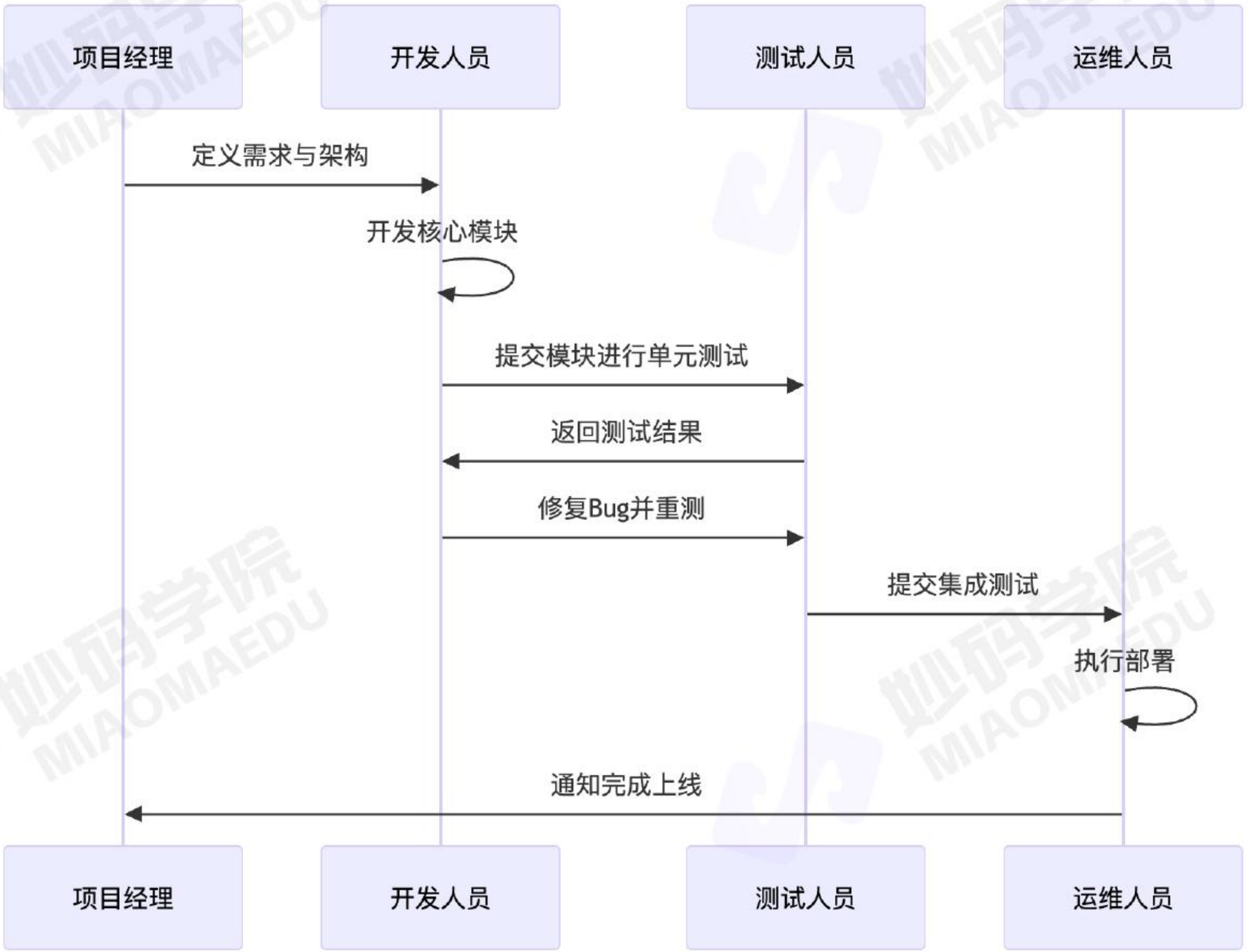
项目架构流程图



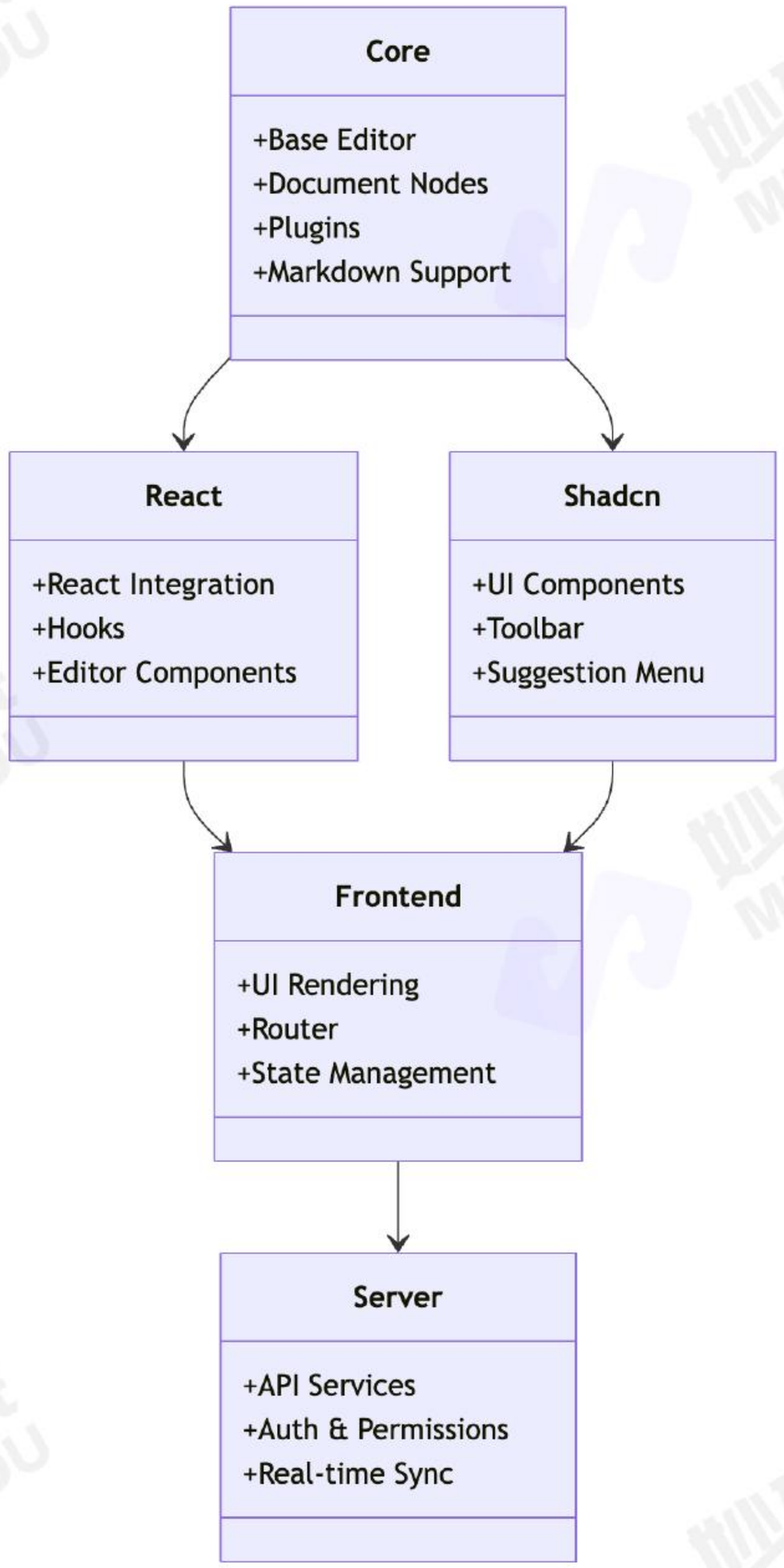
项目架构图



核心功能开发流程



项目模块详细分层架构图



该项目为全栈类型项目，包含编辑器基础 SDK、编辑器 SDK React 适配、编辑器 UI 套件、编辑器 web 网页端、编辑器桌面应用、编辑器 server：

- packages/core 编辑器核心基础 SDK
- packages/react 编辑器 SDK React 适配
- packages/shadcn 编辑器基于 tailwind 的 shadcn ui 套件
- apps/backend/server 编辑器服务端
- apps/frontend/web 编辑器应用网页端
- apps/frontend/app 编辑器应用桌面端

项目工程化设计

本项目名为 **miaoma-docs**，是一个全栈类型的企业级文档编辑器架构设计。

该项目采用模块化的文件结构，包含前端、后端、核心 SDK、UI 套件和基础配置等内容。

根目录结构

```
miaoma-docs
├─ .cspell                // 拼写检查配置目录
├─ .gitignore             // Git 忽略文件配置
├─ .husky                 // Git 钩子配置
├─ .prettierignore        // Prettier 忽略文件配置
├─ .prettierrc            // Prettier 格式化配置
├─ LICENSE                // 项目许可协议
├─ commitlint.config.js   // Git 提交信息规范配置
├─ cspell.json            // 拼写检查工具 cspell 配置文件
├─ eslint.config.js       // ESLint 配置文件
├─ package.json           // 项目根依赖配置
├─ pnpm-lock.yaml         // pnpm 锁文件，记录依赖的版本
├─ pnpm-workspace.yaml    // pnpm 工作区配置文件
├─ tsconfig.client.json   // 客户端 TypeScript 配置
├─ tsconfig.json          // TypeScript 根配置
├─ tsconfig.server.json   // 服务端 TypeScript 配置
└─ turbo.json             // Turborepo 配置文件，用于项目编译和构建
```

项目结构

项目分为多个子模块，分别用于前端、后端、核心 SDK、React 适配和 UI 套件等功能。以下是各主要模块的详细说明：

- **apps**：项目的应用层，包含前端和后端应用。
- **packages**：项目的功能模块层，包含核心 SDK、React 适配和 UI 套件。

apps 子目录

`apps` 目录下包含项目的前端和后端应用模块：

backend（服务端）

```
apps/backend/server
```

此目录为项目的服务端应用，提供以下主要功能：

- **用户认证与权限管理**：为不同角色的用户提供权限管理。
- **数据同步与协同**：基于 yjs 实现实时协同编辑，确保用户间数据一致。
- **API 接口**：为前端提供数据接口支持，包括文档数据、用户信息等。

frontend（前端）

```
apps/frontend/web
├─ README.md
├─ index.html              // 项目入口 HTML 文件
├─ package.json            // 前端依赖配置文件
├─ src                     // 前端源码目录
│   ├─ components          // 自定义组件目录
│   ├─ pages               // 页面组件
│   ├─ router              // 路由配置
│   ├─ utils               // 工具函数
│   ├─ views               // 视图层
│   └─ main.tsx            // 项目入口文件
├─ tailwind.config.ts      // Tailwind CSS 配置
└─ vite.config.ts          // Vite 打包配置
```

此目录为项目的网页端前端应用，包含核心的编辑器组件、页面、路由等实现。

主要功能：

- **文档编辑与展示**：基于 React 实现的前端展示和编辑功能。
- **实时协同**：与服务端通信，实现多人实时编辑。
- **UI 布局**：基于 Tailwind CSS 的样式设计，确保视觉一致性。

packages 子目录

`packages` 目录包含核心 SDK、React 适配和 UI 组件库等模块：

core（核心 SDK）

```
packages/core
├── src
│   ├── api                 // 提供各类数据操作的 API 模块
│   ├── blocks              // 定义编辑器支持的各种块元素
│   ├── editor              // 编辑器核心逻辑实现
│   ├── extensions          // 编辑器扩展功能
│   ├── i18n                // 国际化配置
│   ├── schema              // 定义文档结构的 schema
│   └── util                // 工具函数集合
└── tsup.config.ts          // 构建配置
```

核心 SDK 包含编辑器的基本功能与逻辑，主要作用是为不同的内容块（如文本块、代码块、图像块等）提供插入、删除、更新等操作支持。还包含扩展功能（如颜色选择、快捷键、侧边菜单）和国际化支持。

react（React 适配）

```
packages/react
```

```
├─ src
│   ├─ blocks                // React 组件封装的内容块
│   ├─ components            // 可复用的 UI 组件
│   ├─ editor                // 编辑器的 React 封装
│   ├─ hooks                 // 自定义钩子函数
│   ├─ schema                // 定义文档结构的 React schema
│   └─ util                  // 实用工具函数
└─ tsup.config.ts             // 构建配置
```

此模块封装了编辑器的 React 适配，使编辑器能够更好地与 React 项目进行集成。包含对文档内容块的 React 组件封装、React schema 定义以及各种自定义钩子函数。

shadcn（UI 套件）

```
packages/shadcn
├─ src
│   ├─ components            // 基于 Tailwind 的 UI 组件
│   ├─ menu                  // 菜单组件
│   ├─ panel                 // 面板组件
│   ├─ sideMenu              // 侧边栏菜单组件
│   ├─ suggestionMenu        // 建议菜单组件
│   ├─ tableHandle           // 表格操作组件
│   └─ toolbar               // 工具栏组件
└─ tailwind.config.js         // Tailwind CSS 配置
```

`shadcn` 目录包含一系列基于 Tailwind CSS 的 UI 组件，如按钮、标签、输入框、选择器等，这些组件用于实现一致的编辑器界面风格，且便于复用。

代码质量控制与自动化工具

- **.husky**：用于设置 Git 钩子，确保在提交代码之前执行自动化检查。
- **.cspell**：拼写检查工具，保证代码中无拼写错误。
- **commitlint.config.js**：用于规范 Git 提交信息。

- **eslint.config.js**：代码风格检查配置。
- **.prettierrc** 和 **.prettierignore**：用于代码格式化，确保代码风格一致。

课程目标

💡
- 初中级：
  a. **企业级文档项目需求分析与方案评审**：能够理解企业级文档编辑项目的核心需求，包括协同编辑、富文本支持、可扩展性等；掌握项目需求评审的流程，理解从需求到解决方案制定的基本思路
  b. **项目架构设计与模块化分析**：理解企业级文档编辑器的架构设计原则，掌握基于 monorepo 的全栈项目架构的组织方式
  c. **编辑器核心功能分析与拓展**：理解基于 `Tiptap` 和 `Prosemirror` 实现的编辑器核心功能，能够设计分析不同类型的文档节点（如文本、列表、表格、图像等）和基础扩展（如格式化工具、历史记录、拖拽等）的作用及实现原理
- 高级：
  a. **高级架构设计与模块化思想**：具备从零开始设计企业级文档编辑器架构的能力，理解如何将核心编辑功能、协同编辑模块和 UI 组件库整合在项目架构中，并掌握性能与扩展性最佳实践
  b. **文档节点与扩展实现方案**：能够深入理解 `Prosemirror` 和 `Tiptap` 的节点扩展机制，分析自定义节点（如代码块、复杂表格、多媒体嵌入等）的实现原理，掌握协同光标、emoji 支持、Markdown 导入导出等高级功能的架构设计
  c. **协同编辑实现与同步机制分析**：理解基于 `yjs` 实现协同编辑的工作原理，掌握实时数据同步的流程和核心技术，分析如何在多人协同编辑中实现低延迟、高一致性，并深入了解 `y-prosemirror` 的协同适配
  d. **工程化与质量控制思想**：理解使用 ESLint、CSpell、Prettier、CommitLint 等工具进行代码质量控制的工程化理念，掌握制定工程化规范的流程，理解如何通过代码风格一致性、提交信息规范化和拼写检查来提升项目的可维护性和团队协作效率
  e. **UI 组件库设计思路**：理解基于 `Tailwind CSS` 和 `Radix UI` 构建企业级 UI 组件库的设计思想，掌握工具栏、菜单、表单等复杂 UI 组件的实现思路，理解 `@miaoma-doc/shadcn` 模块在提升编辑器界面一致性和可扩展性中的作用

💡
- 初中级：

a. **富文本编辑器核心原理理解与实现**：掌握富文本编辑器的基本结构与核心原理，包括基于 `contenteditable` 实现可编辑 DOM 元素、AST 数据结构的设计与解析，以及 HTML 与 AST 间的转换逻辑

b. **掌握数据协议设计与样式管理**：理解富文本编辑器的数据协议约定，能够定义节点类型（文本、段落、提及等）和样式属性（如粗体、斜体等）；掌握自定义节点的设计与插入

c. **掌握基础功能实现与优化**：能够实现富文本编辑器的核心功能（如文本格式化、节点插入与删除、自定义样式应用）；掌握撤销与重做功能的实现，理解编辑历史管理机制

- 高级：

a. **掌握扩展功能实现与高级节点设计技巧**：深入理解基于 `ProseMirror` 和 `Tiptap` 的节点扩展机制，掌握自定义复杂节点（如代码块、表格、多媒体嵌入）的实现原理与方法；能够扩展高级功能如协同光标、Markdown 支持

b. **了解协同编辑与数据同步机制**：理解基于 `Yjs` 的协同编辑原理，掌握 `CRDT` 和 `y-prosemirror` 的同步机制；能够设计和实现实时协作功能（包括光标同步、框选、输入同步等）

c. **掌握基于 tiptap 的编辑器 SDK 封装思路**：掌握基于 monorepo 架构设计的富文本编辑器 SDK 封装思路，沉淀 core、react 以及 UI 套件包

💡

- 初中级：

a. **熟练掌握文档提及（Mention）功能的开发与实现**：触发字符（如“@”）快速实现文档引用功能，使用 `createReactInlineContentSpec` 定义提及功能的内容类型，并实现元信息存储与渲染。

b. **熟练掌握文档关系图谱的可视化开发**：使用节点和边的形式展示文档引用关系，动态更新图谱；结合 `@xyflow/react` 构建交互式图谱，并使用 `d3-force` 实现力导向布局；掌握自定义节点样式、边样式以及交互逻辑的开发技能。

c. **熟练使用 PostgreSQL 数据库的部署与集成**：通过 Docker 和 Docker Compose 部署 PostgreSQL 数据库环境，掌握数据库的容器化部署、数据持久化与调试技巧，并在 NestJS 项目中动态配置和集成 PostgreSQL 数据库。

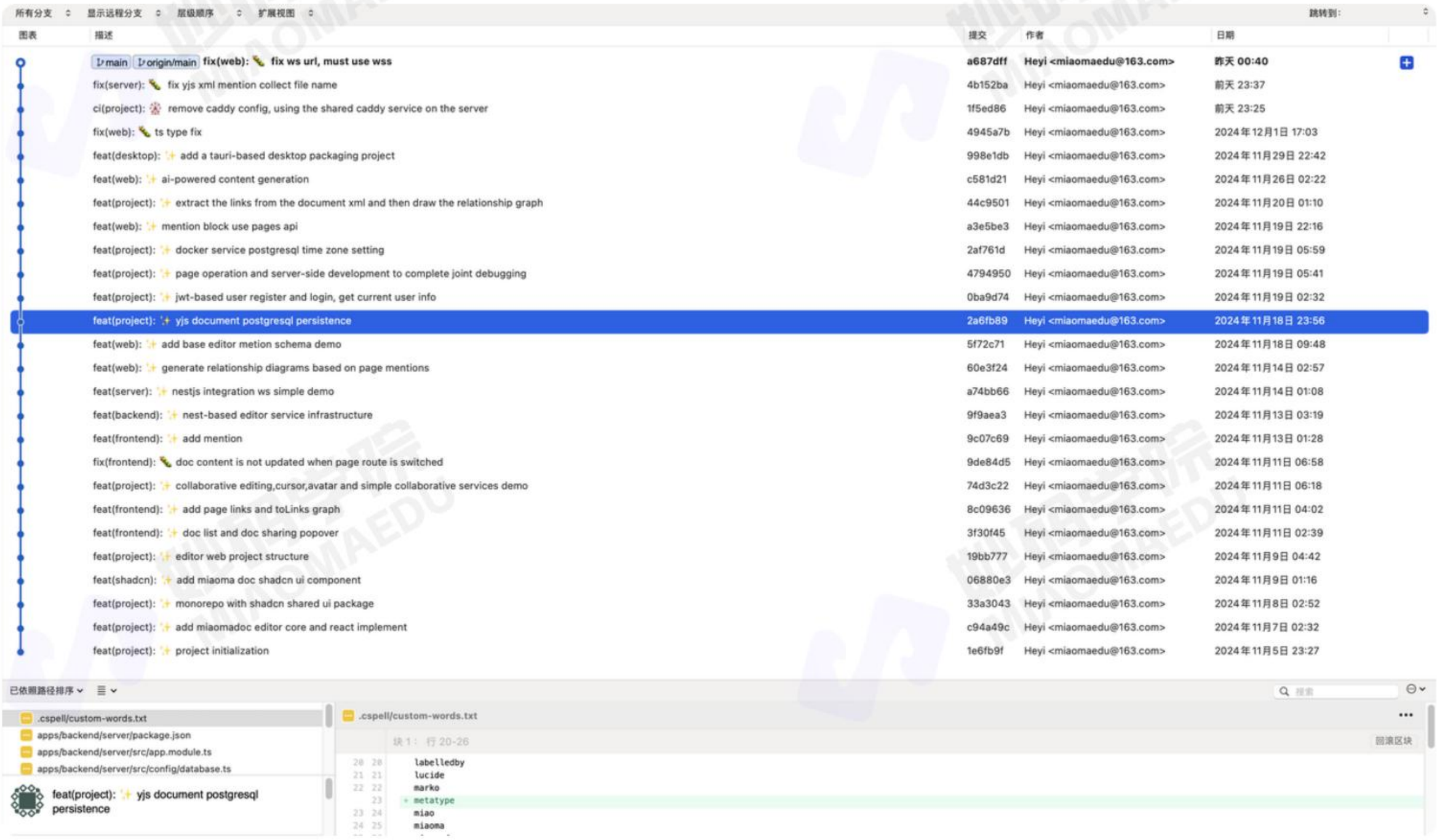
d. **熟练掌握 NestJS 服务的基础架构**：配置全局异常过滤器、WebSocket 适配器、全局路由前缀，集成 Swagger 提供 API 文档支持；使用模块化设计提升代码可维护性，并动态加载数据库配置以适应多环境需求。

e. **了解 WebSocket Gateway 的实现**：使用 `@WebSocketGateway` 构建实时通信功能，通过 `@SubscribeMessage` 处理客户端消息并实现逻辑；使用 WebSocket 原生适配器替代默认适配器优化通信性能。

- 高级：
  a. **掌握复杂交互图谱的高性能实现**：深入理解 d3-force 的物理仿真模型，动态调整力导向布局参数；集成 @xyflow/react 与 d3-force 构建复杂交互逻辑；优化图谱性能以支持高并发动态更新。
  b. **掌握 NestJS 高级功能与全栈整合**：自定义 WebSocket 适配器以满足不同需求，深入使用 NestJS 模块化机制扩展服务，实现复杂服务之间的跨模块通信与协作。
  c. **了解复杂文档协作与同步服务架构设计**：构建双服务架构（持久化服务 + 实时协作服务），实现文档的持久化存储与实时同步；集成 Yjs 与 PostgreSQL 适配器，实现实时文档协作，并设计数据流以保障服务间的数据一致性。
  d. **掌握高性能实时文档编辑与冲突解决策略**：优化同步架构以处理网络延迟与断线重连，设计冲突解决策略保障文档数据一致性，使用差异同步算法提升同步效率并降低系统负载。

💡
- **初中级**：
  a. **熟练掌握协同服务的基础开发**：使用 **yjs** 与 **y-websocket** 实现实时协同功能，支持文档与图形的多用户编辑。
  b. **熟练实现协同数据的持久化存储**：通过自封装的 **y-postgresql** 持久化协作数据，确保数据一致性与简单版本管理。
  c. **掌握 XML 数据解析与关系图生成**：使用 **xml-js** 工具解析 XML 数据并生成基础的关系图，支持简单交互与静态布局。
  d. **集成基础 AI 功能**：了解 AI 开发流程，通过 **Dify AI** 接口实现智能补全、基础问题解答等功能，支持初步的智能助手开发。
- **高级**：
  a. **掌握复杂协同服务的高性能优化**：优化 **yjs** 算法以支持高并发场景，解决实时协作中的性能瓶颈；设计复杂的协作逻辑以支持多场景并行处理。
  b. **深入理解复杂关系图谱的动态交互**：通过 **xml-js** 生成复杂关系图数据，结合 D3.js 或其他工具实现动态布局调整与交互逻辑优化。
  c. **集成多场景智能助手**：基于 **Dify AI** 开发智能助手，支持知识图谱、智能建议和复杂任务的动态交互；优化模块设计以支持灵活扩展与适配需求。
  d. **掌握容器化与分布式部署技术**：使用 **Docker** 和 **Docker Compose** 进行复杂服务的容器化部署；优化服务架构以支持高可用性与弹性扩展，保障协同服务与 AI 应用的稳定运行。

代码 commit 截图



演示地址

https://docs.miaomaedu.com

妙码协同文档 | 妙码学院

“企业级项目”指采用企业常用技术栈及开发流程的教学模拟项目，不等同于商业生产环境使用的项目