



# 知识图谱可视化定义系统& 食谱领域智能问答系统

CO\$IN小组

181250003 蔡尚达

181250015 陈彦泽

181250083 林希澄

181250104 苗轶轩



2021/6/20

# CONTENTS

## 1. 项目介绍

- A. 核心特点亮点介绍
- B. 项目代码结构介绍
- C. 软件工程过程介绍

## 2. 项目演示

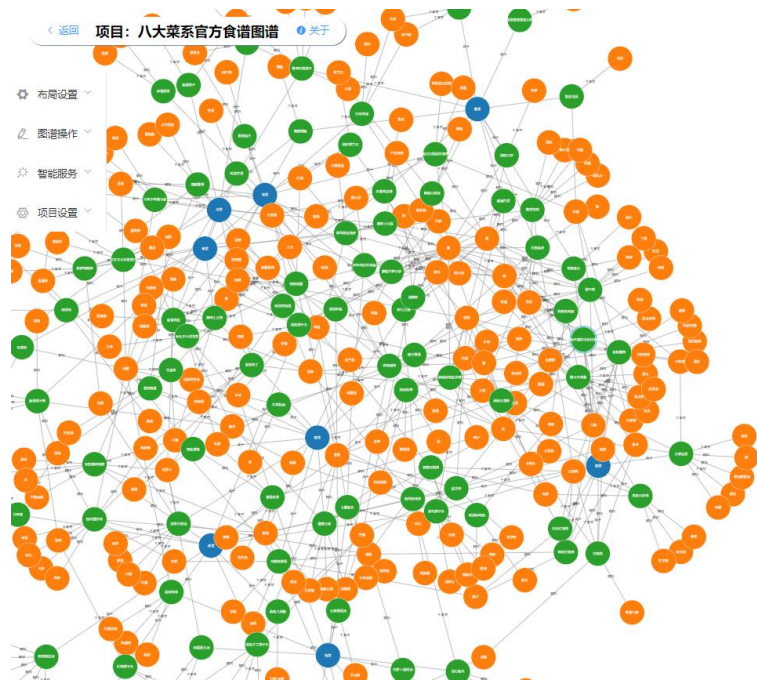
## 3. Q&A



# /01 项目介绍

## Introduction

# 项目整体介绍



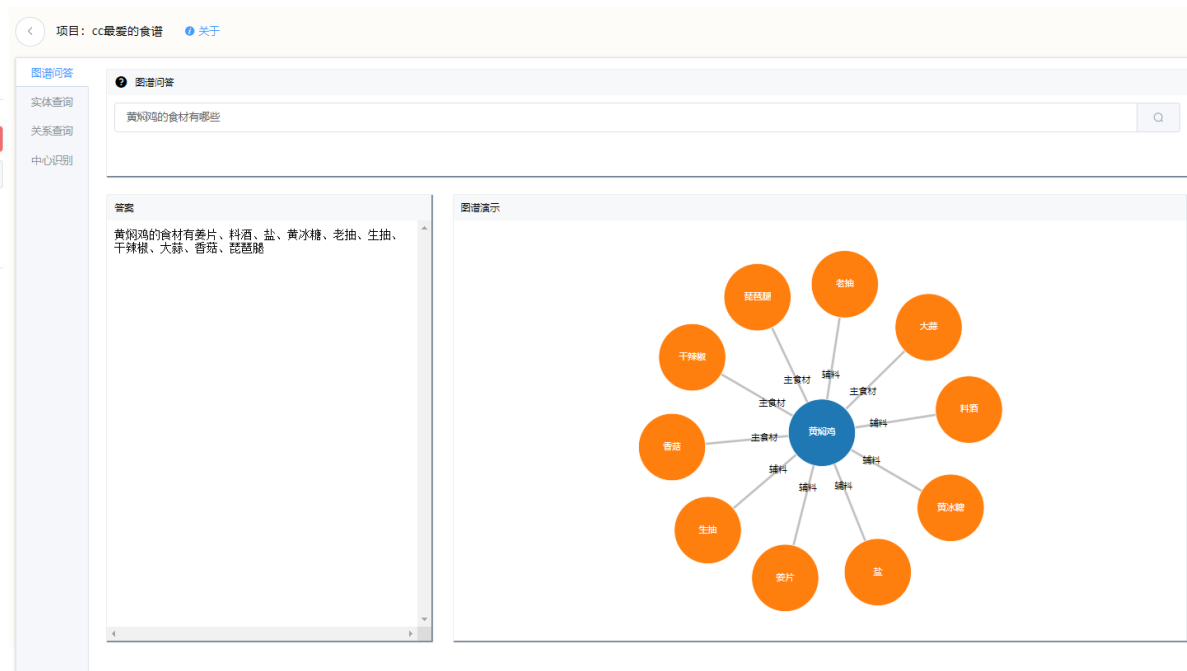
## 迭代一、二

- 知识图谱定义
- 布局相关



数据集

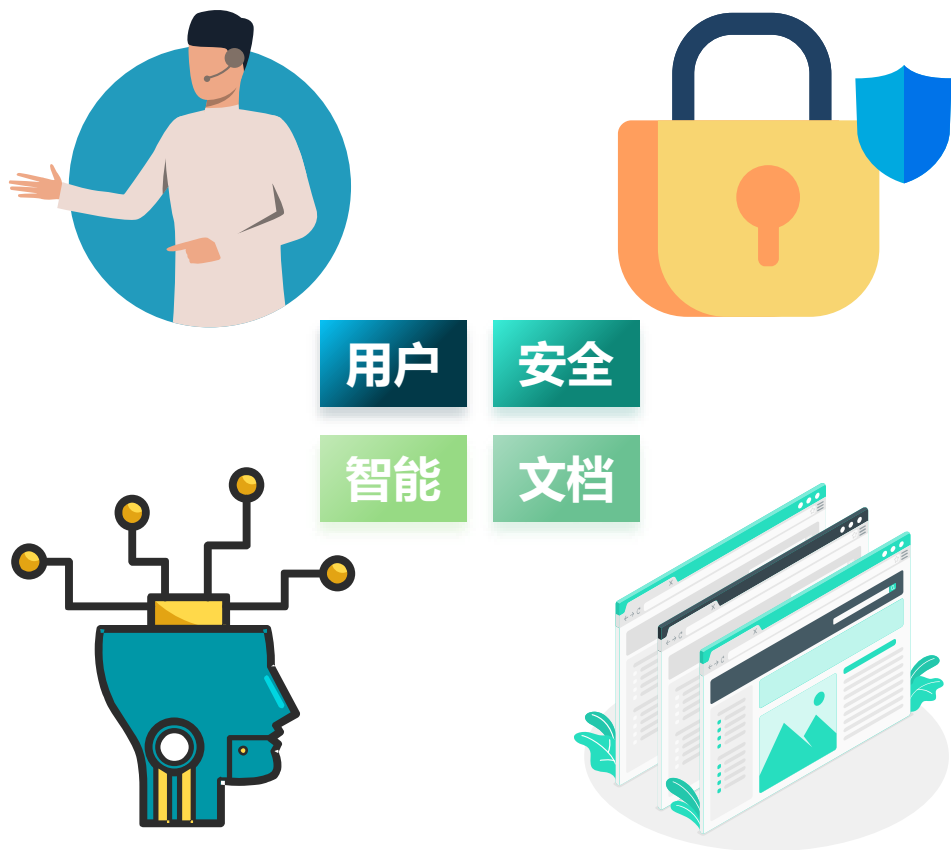
智能应用



## 迭代三

- 智能问答 KBQA ( Knowledge-based Question and Answering )
- 图谱助手 Helper ( 语义识别 )
- 节点查询
- 关系查询
- 中心识别

# 产品化!



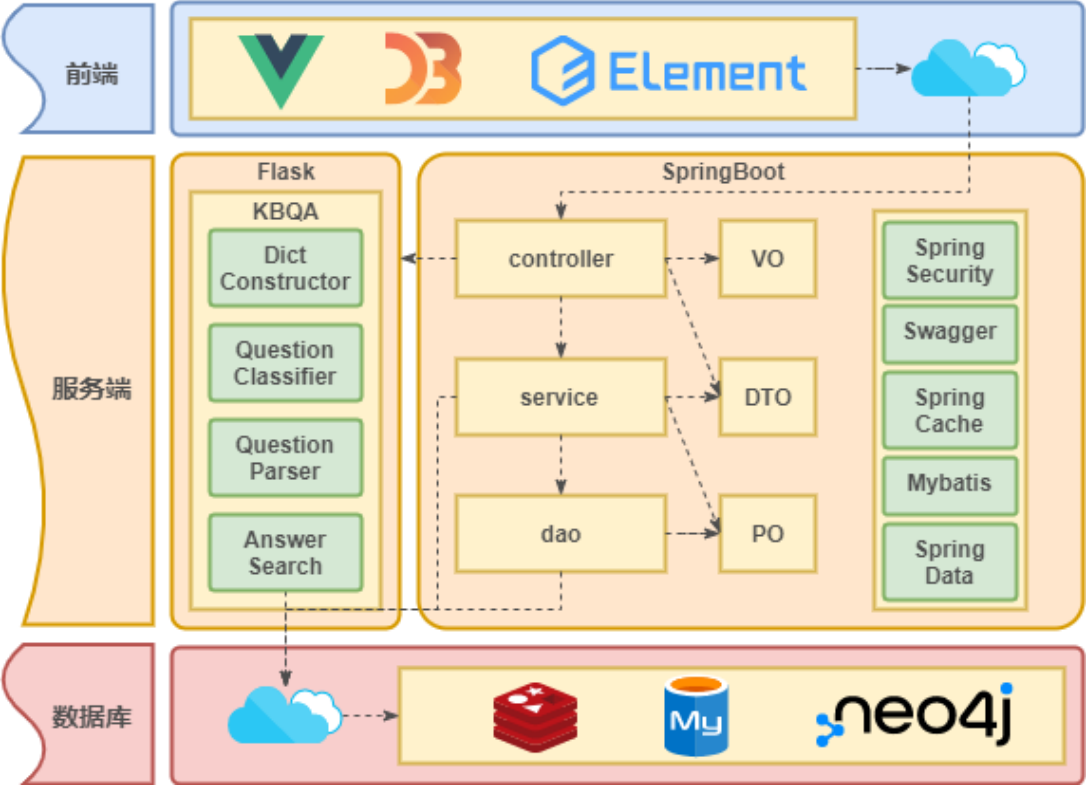
**01. 用户模块**  
用户登录、注册，项目私有、公开

**02. 安全模块**  
对称加密、非对称加密、授权

**03. 智能问答**  
用自然语言使用项目

**04. 文档化**  
提供用户手册和系统设计手册

# 系统设计



## 前端

Vue.js  
D3  
Echart  
ElementUI  
Router  
Vuex

## 服务端

Flask  
Spring Boot  
Spring Security  
Spring Cache  
Swagger

## 数据库

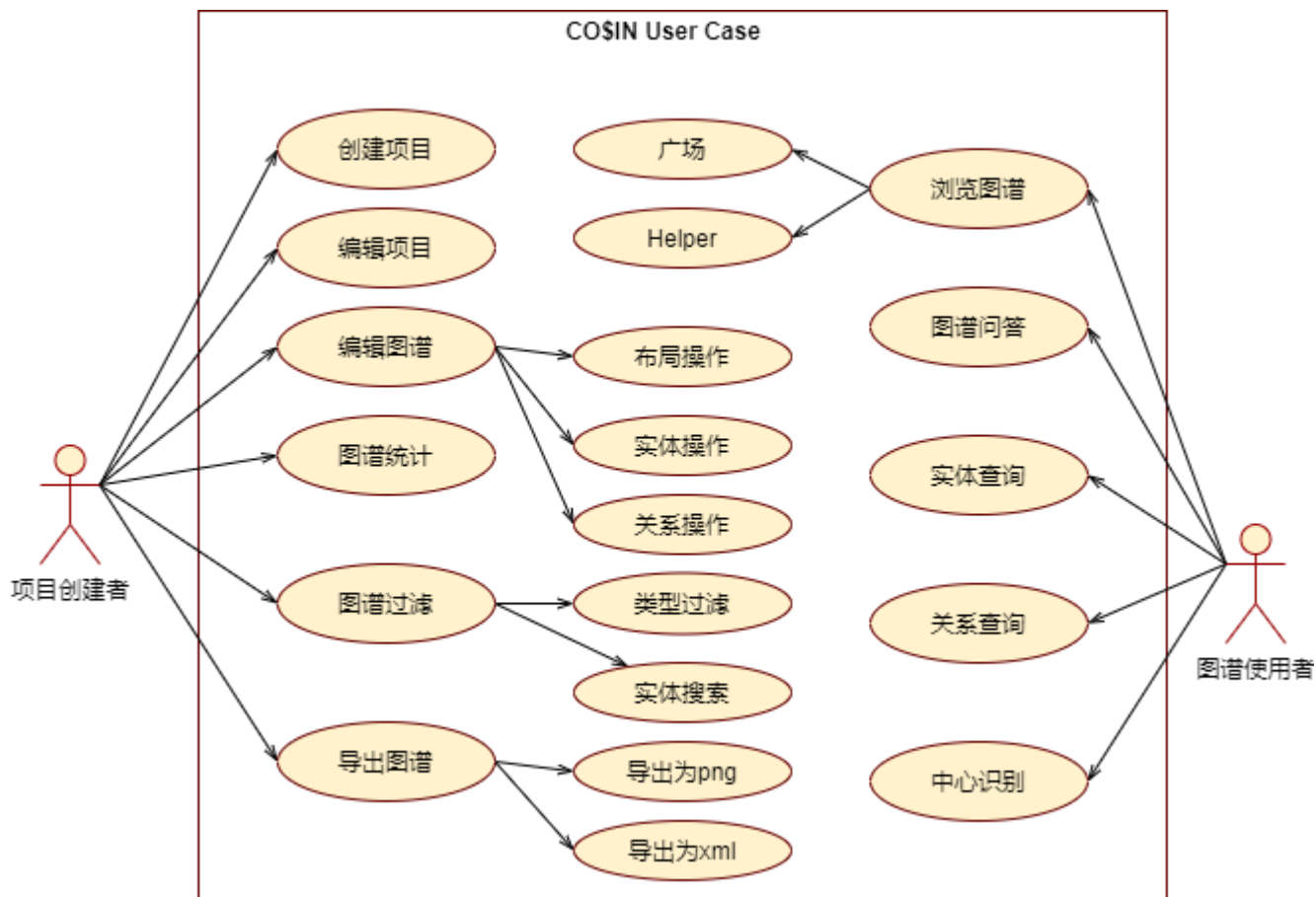
Redis  
Neo4j  
MySQL

# 用例

我的项目 广场 Helper 文档 系统设计



## co\$in 文档



### 1 我是项目创建者

- 1.1 创建知识图谱
- 1.2 编辑项目
- 1.3 编辑图谱
  - 1.3.1 图谱内容
    - 1.3.1.1 实体操作
    - 1.3.1.2 关系操作
  - 1.3.2 布局内容
- 1.4 图谱统计
- 1.5 图谱过滤
  - 1.5.1 图谱搜索
  - 1.5.2 类型过滤
- 1.6 导出图谱
  - 1.6.1 保存为png
  - 1.6.2 保存为xml

### 2 我是项目使用者

- 2.1 浏览图谱
  - 2.1.1 广场
  - 2.1.2 Helper
- 2.2 图谱问答
- 2.3 实体查询
- 2.4 关系查询
- 2.5 中心识别

### 3 附录

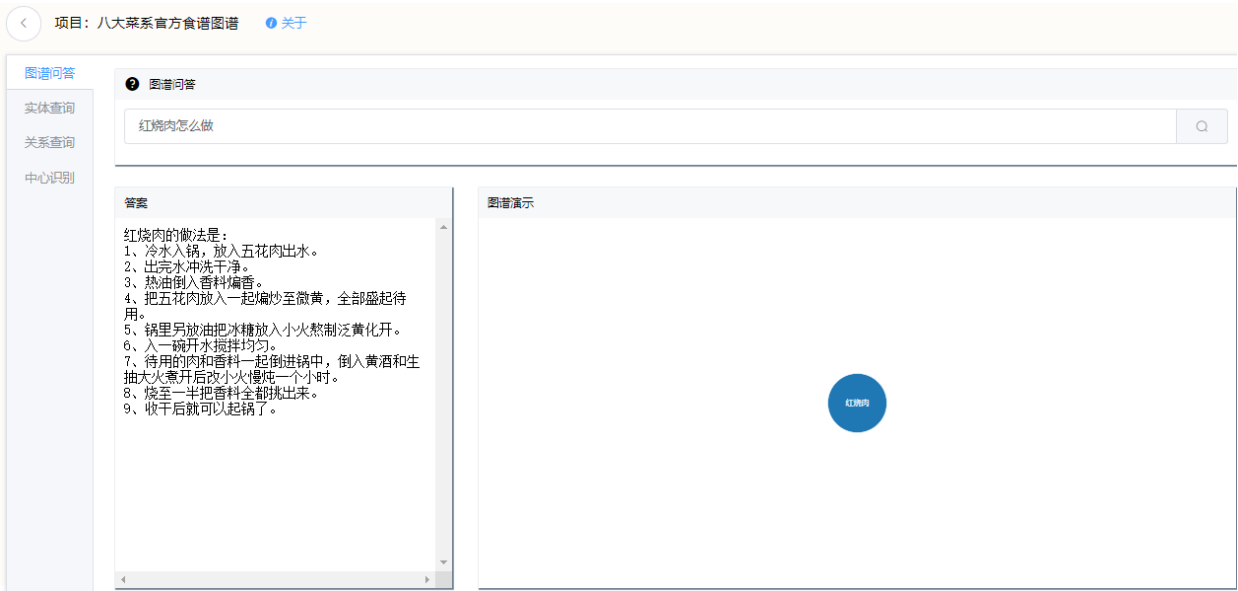
- 3.1 可被智能应用使用的实体类型
- 3.2 可被智能应用使用的关系类型

# /A 核心亮点特点介绍



# 出发点

- 信息丰富，知识匮乏
  - 百度、谷歌、美食天下，有着极为丰富的数据，但是检索起来困难，按层搜索
  - 提供一个信息检索的工具
- 迭代一、二提供了一个丰富完善的可视化知识图谱定义工具，怎么利用起来？
  - 作为用户自己构建数据集的工具
- 用户可依赖的确定性
  - 用户无需知道太多的知识，通过自然语言进行问答

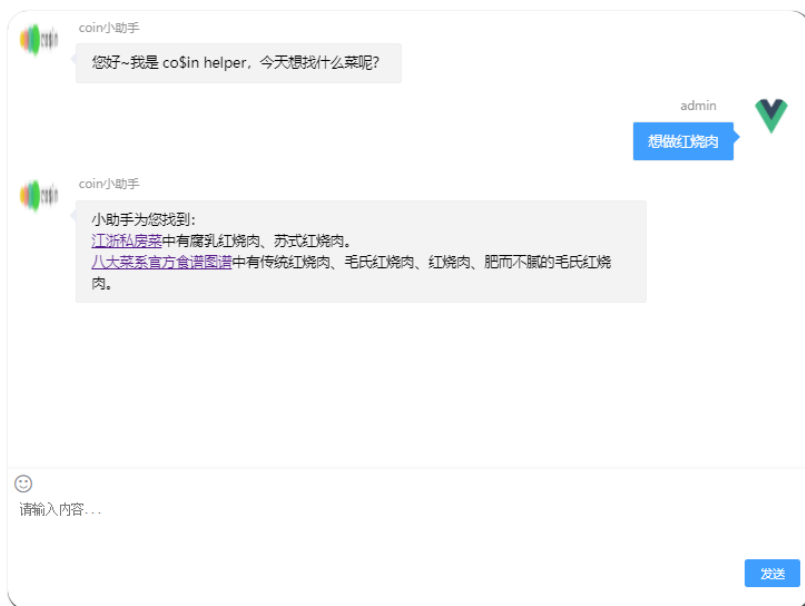


# 产品的用户体验地图

## • 故事一：

- 小希是和大部分人一样，睡觉前都会有滑滑手机的习惯，有一天他在b站上看了某个美食博主的烹饪视频，觉得看起来很美味很满意，决定第二天要做给女朋友吃，但是他忘记收藏也忘记了博主的名字什么的，倒头就睡了。结果隔天早上起来就发现再也找不到昨天那个视频，只能说下次一定。
- 从此以后，他每次看到喜欢的视频，都会第一时间点赞投币收藏，但久而久之，收藏夹越来越满，每次做菜前都要在收藏夹里翻上半天，再拉动进度条，十分麻烦。
- 有一天，小希遇到了CO\$IN系统，这是一个基于知识图谱的菜谱问答系统，在这里，他只需要记住做的是什么菜，甚至不需要记下全称，就可以在小助手的帮助下知道菜谱的做法、食材等信息...

我的项目 广场 Helper 文档 系统设计



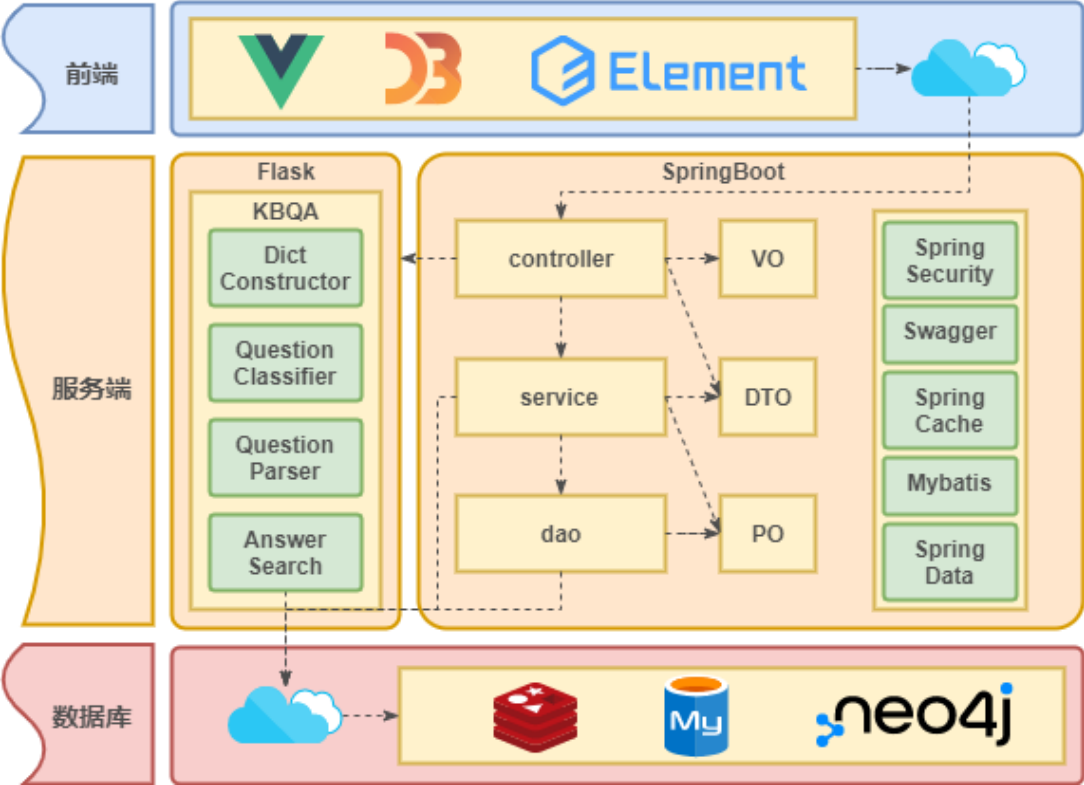
# 产品的用户体验地图

- 故事二：
  - 小希借助着CO\$IN系统，很快就成为了美食大厨，甚至对许多菜品进行了改良，在家门口开了一家“cc饭馆”，深受大家的喜欢。为了与更多人分享，小希在CO\$IN系统中使用了可视化工具制作了“cc最爱的食谱”这一知识图谱，包括了黄焖鸡、卤肉饭等他最得意的菜谱。
  - 有一天，阿菜来cc饭馆吃饭。在这里阿菜可以使用“cc最爱的食谱”的问答系统对菜谱进行了解。饭后，他对cc的厨艺十分满意，也想学习卤肉饭的做法，cc向他推荐了CO\$IN系统，在这里阿菜利用智能问答，可以随时学习食谱中的菜谱做法、所需食材等信息。



# / B 项目代码结构介绍

# 总体架构



## 前端

Vue.js  
D3  
Echart  
ElementUI  
Router  
Vuex

## 服务端

Flask  
Spring Boot  
Spring Security  
Spring Cache  
Swagger

## 数据库

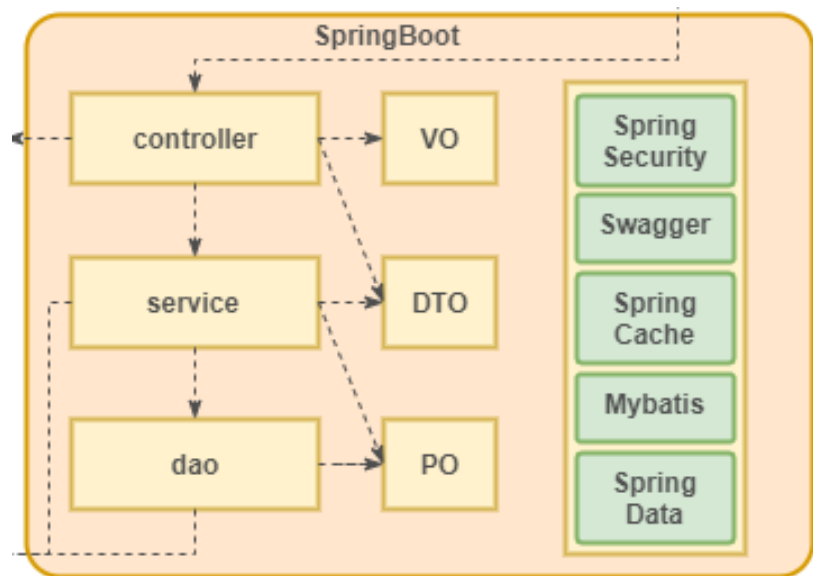
Redis  
Neo4j  
MySQL

# 数据库

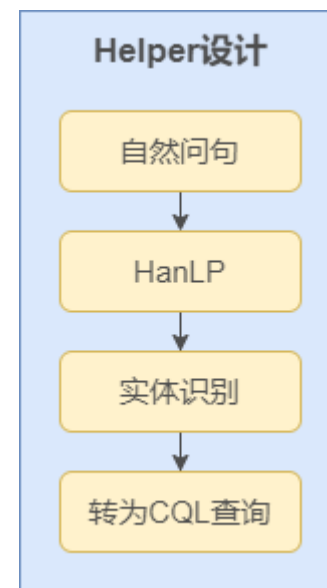
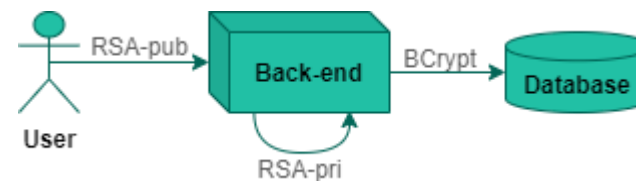


数据库	存储内容	目的
MySQL	布局信息、用户信息、项目信息等结构化数据	关系型数据库，存储结构化数据方便
Neo4j	存储图数据	图数据库，天然符合图谱的数据结构存储，提供了内置的算法库
Redis	缓存、图ID生成	内存KV数据库，减少磁盘IO，读取快

# 服务端 — 知识图谱服务



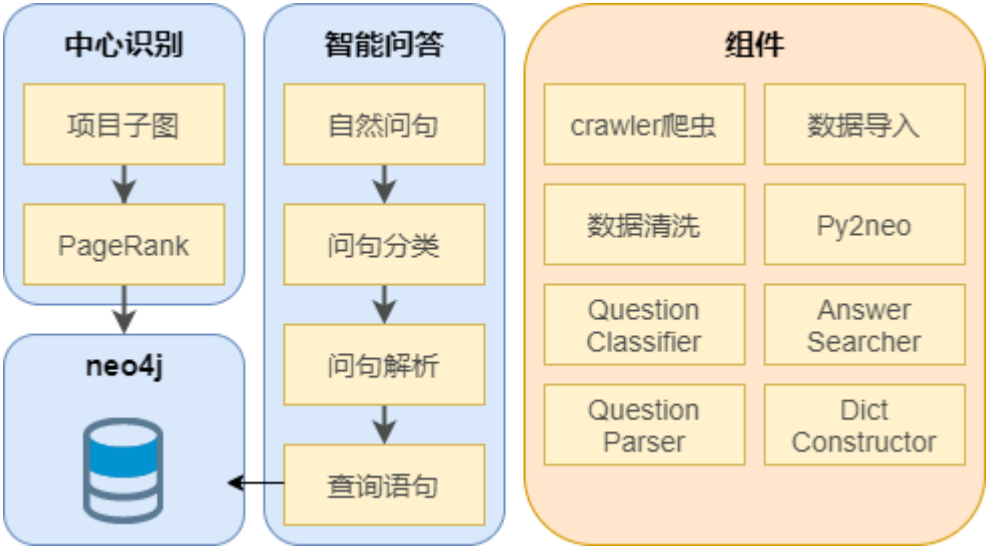
- 用户管理模块
  - 授权、认证
  - 对称加密与非对称加密
  - Spring Security
- Helper模块
  - 全局索引, HanLP
- 项目管理模块
  - 读多写少, Spring Cache
- 图谱管理模块
  - Spring Data Neo4j
- 接口文档
  - Swagger



# 服务端 — 智能应用服务

问题类型	中文含义	问题举例
food_query ✓	查询菜谱属性	烤肠怎么做
food_constraint ✓	查询符合约束的菜	微辣的炒菜有哪些
desc_recipe ✓	描述菜谱	红烧肉
desc_cuisine ✓	描述菜系	闽菜
desc_material ✓	描述食材	五花肉
ingredients ✓	食材查询	红烧肉用什么做
ingredients_prime ✓	主食材查询	红烧肉的主食材是什么
ingredients_sub ✓	辅料查询	红烧肉的辅料是什么
ingredients_num ✓	查询所需食材的量	红烧肉要用多少五花肉
ingredients_include ✓	某个菜是否用了某个东西	红烧肉要用五花肉吗？
food_belong ✓	查询菜的菜系	红烧肉属于什么菜系？
cuisine_food ✓	查询菜系有哪些菜	闽菜有哪些美食？
cuisine_query ✓	查询菜系	有哪些菜系的菜
unknown ✓	不知道	我帅吗

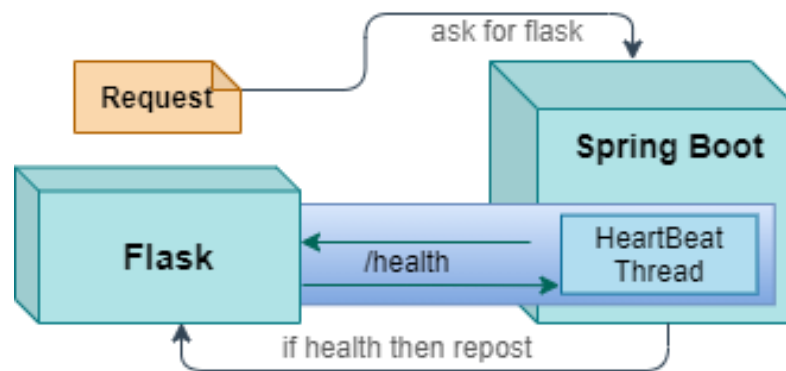
- KBQA
  - Rule-based
  - 基于用户通过可视化图谱定义系统构建的数据
- 中心识别
  - 基于子图，PageRank
- 数据爬取、预处理
  - 美食天下





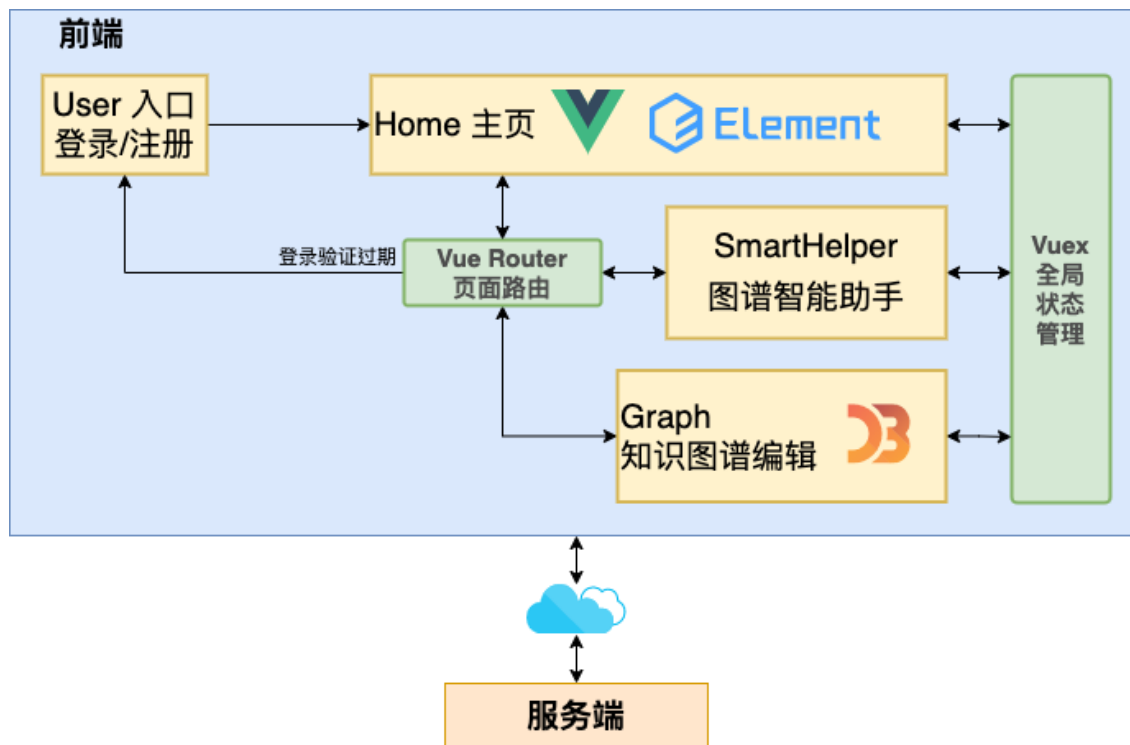
# 服务端 — 服务调用

- 异构微服务
  - 基于 Python 的 Flask 服务
  - 基于 Java 的 Spring Boot 服务
- 服务调用
  - 由 RestTemplate 进行统一的转发
- 服务发现
  - 使用心跳机制
  - Flask 提供 /health 接口
  - Spring Boot 进行定时监听，如果服务异常，对请求进行熔断



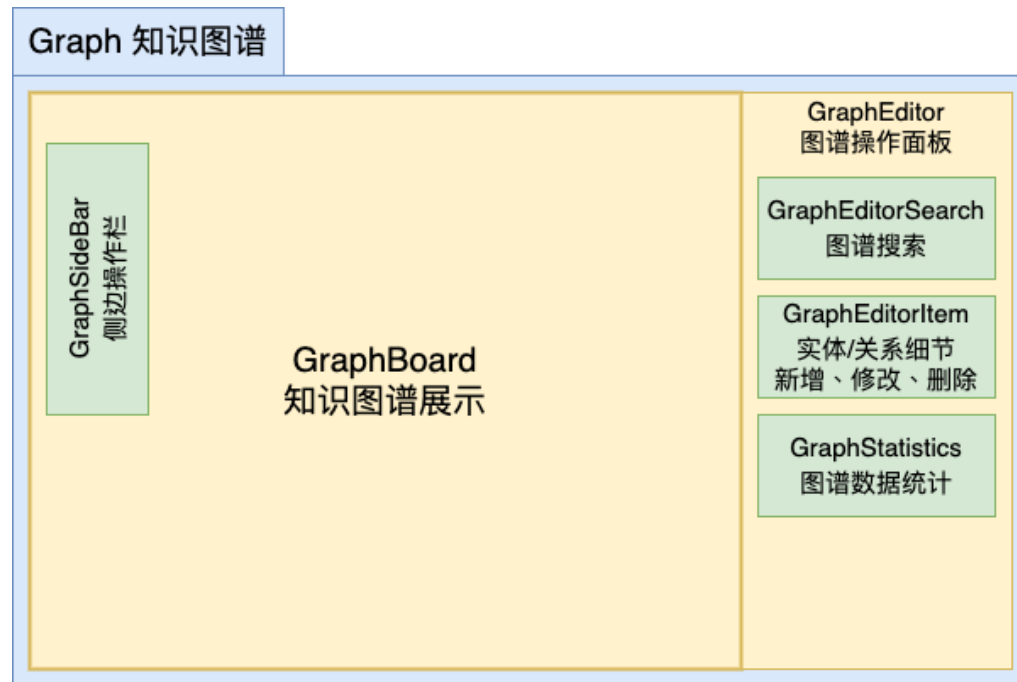
## 前端 — 整体架构

- 页面主要模块
    - 用户入口 User: 登录/注册
    - 主页 Home: 用户项目管理、其他模块入口
    - 知识图谱展示 Graph: 知识图谱的展示、编辑、导出
    - 图谱智能助手 SmartHelper: 图谱问答、实体/关系查询、中心识别
  - 页面核心中间件、三方库工具
    - 前端开发框架基础 Vue
    - 页面路由 Vue Router
    - 全局状态管理 Vuex
    - 页面样式组件库 ElementUI
    - 图谱可视化 D3
    - 数据统计可视化 EChart
    - 智能助手问答页面 JwChat
- 
- ```
graph LR; subgraph 前端; direction LR; A[User 入口  
登录/注册] --> B[Home]; B -- "登录验证过期" --> A; end;
```
- The diagram illustrates the front-end architecture. It features a light blue background with a title '前端' (Frontend) in the top left. Two yellow boxes are present: 'User 入口 登录/注册' (User Entry Login/Registration) on the left and 'Home' on the right. A horizontal arrow points from the User box to the Home box. A return arrow points from the Home box back to the User box, labeled '登录验证过期' (Login verification expired).



# 前端 — 图谱页面设计

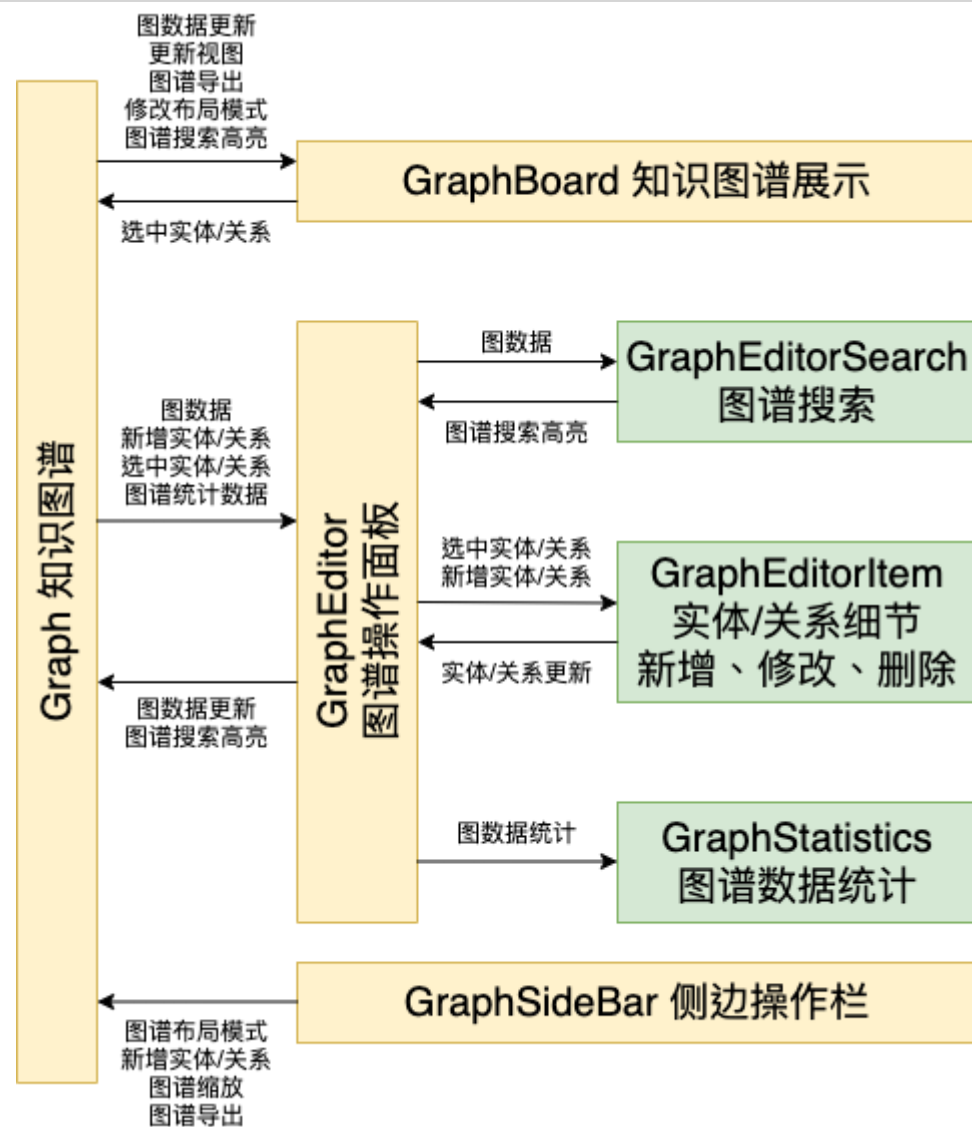
- 图谱展示核心 GraphBoard
- 图谱操作面板 GraphEditor
  - 图谱搜索 GraphEditorSearch
  - 实体/关系的新增、修改、删除、细节查看 GraphEditorItem
  - 图谱数据统计 GraphStatistics
- 侧边操作栏 GraphSideBar
  - 布局模式、布局保存/恢复 GraphLayout
  - 新增实体/关系、图谱缩放重置、图谱导出 GraphAction
  - 智能服务
  - 项目设置



# 前端 — 图谱页面数据流

- 事件驱动模型

- 由 GraphBoard 保存核心图数据
- 其他模块引用图数据进行展示
- 其他模块向 GraphBoard 发起事件修改视图



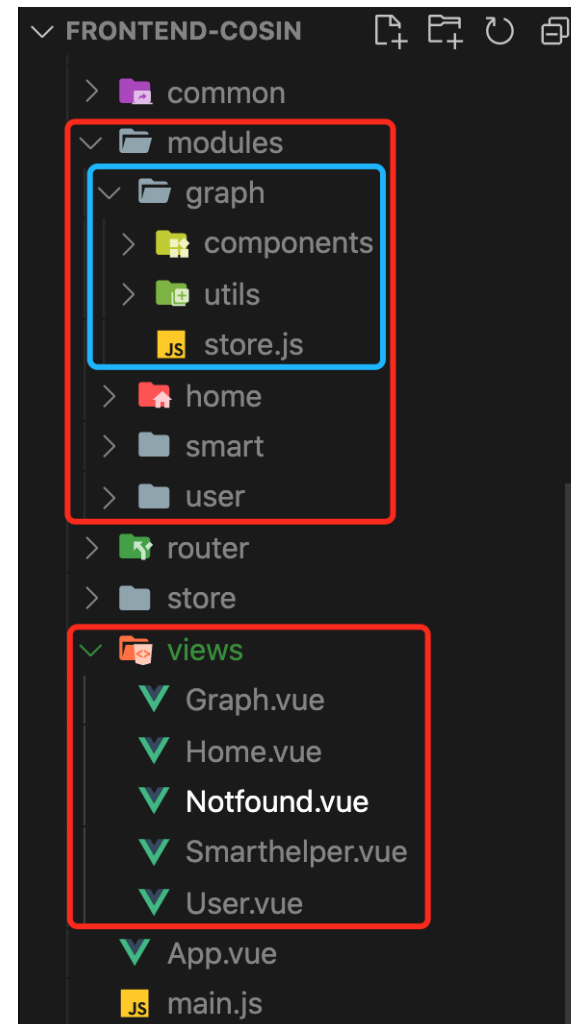
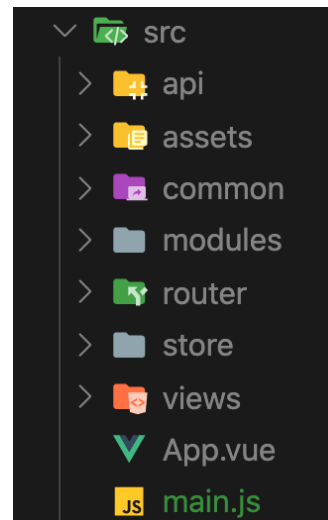
# 前端 — 详细设计模块

- 底层代码模块划分

- api 后端交互封装
- assets 静态资源(图片)
- common 全局共用模块
- views 主要视图
- modules 主要视图模块
- router 页面路由
- store 全局状态管理

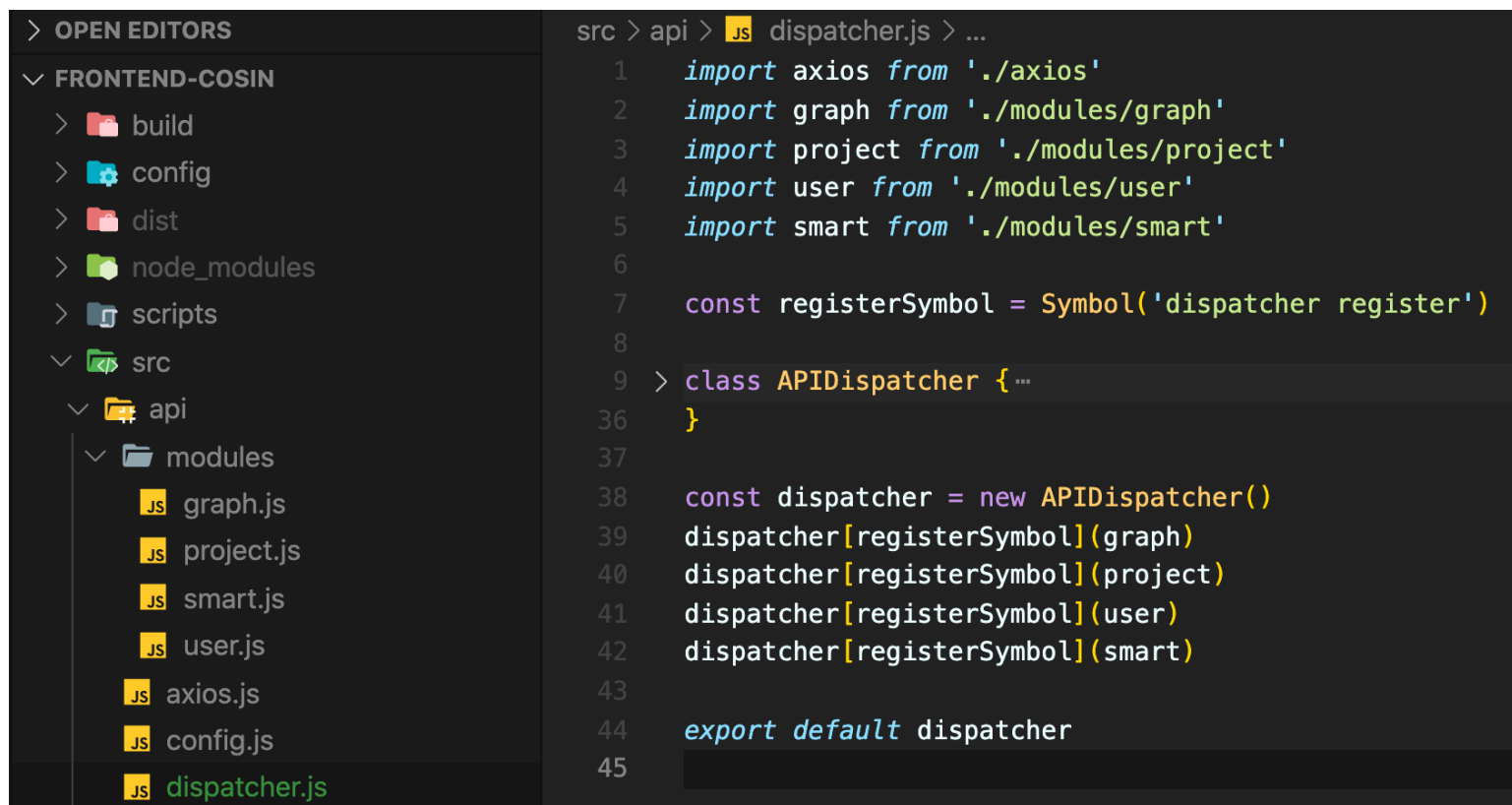
- 视图模块细节

- 每个主要视图在 modules/下拥有一个包、每个包包含下列结构
- components 子视图/组件
- utils 视图内部工具函数
- store 视图内共享全局数据



# 前端 — 后端交互封装

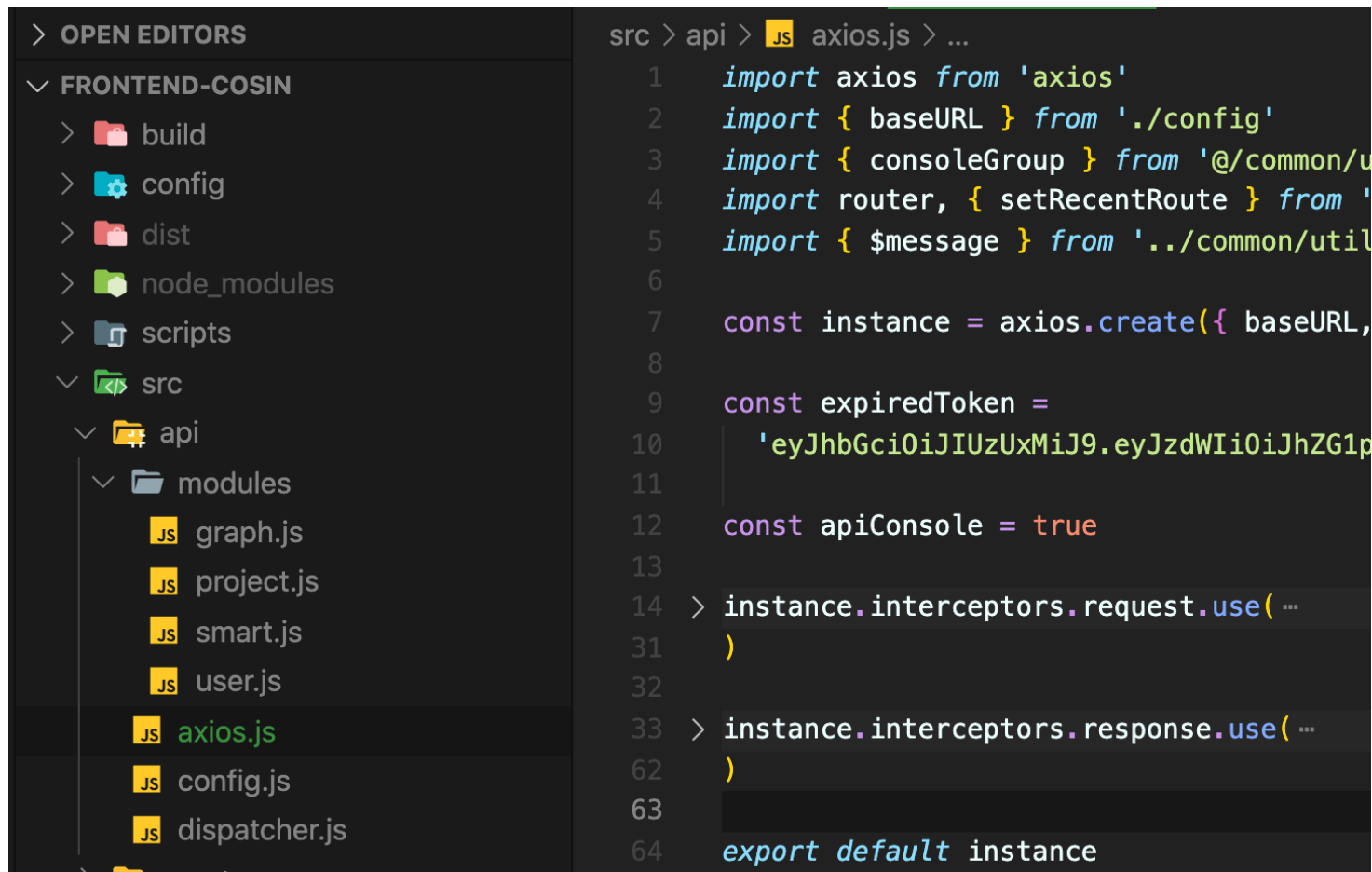
- APIDispatcher 后端 API 封装成内部 DSL
- 对后端接口进行划分并分别向 APIDispatcher 进行注册



```
src > api > JS dispatcher.js > ...
1  import axios from './axios'
2  import graph from './modules/graph'
3  import project from './modules/project'
4  import user from './modules/user'
5  import smart from './modules/smart'
6
7  const registerSymbol = Symbol('dispatcher register')
8
9  > class APIDispatcher { ...
36 }
37
38 const dispatcher = new APIDispatcher()
39 dispatcher[registerSymbol](graph)
40 dispatcher[registerSymbol](project)
41 dispatcher[registerSymbol](user)
42 dispatcher[registerSymbol](smart)
43
44 export default dispatcher
45
```

# 前端 — 后端交互封装

- 透过 axios 接口对后端请求/响应进行拦截



The image shows a VS Code editor interface. On the left, the 'EXPLORER' sidebar displays the project structure. The 'FRONTEND-COSIN' folder is expanded, showing subfolders like 'build', 'config', 'dist', 'node\_modules', 'scripts', and 'src'. The 'src' folder is further expanded to show 'api', which contains a 'modules' subfolder. Inside 'modules', several JavaScript files are listed: 'graph.js', 'project.js', 'smart.js', 'user.js', 'axios.js' (highlighted in green), 'config.js', and 'dispatcher.js'. On the right, the editor window shows the code for 'src > api > axios.js'. The code imports 'axios' and other modules, creates an axios instance with a base URL and a token, and sets up request and response interceptors. The file path 'src > api > axios.js' is visible in the top breadcrumb.

```
src > api > JS axios.js > ...
1  import axios from 'axios'
2  import { baseUrl } from './config'
3  import { consoleGroup } from '@common/u
4  import router, { setRecentRoute } from '
5  import { $message } from '../common/util
6
7  const instance = axios.create({ baseUrl,
8
9  const expiredToken =
10     'eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1p
11
12  const apiConsole = true
13
14  > instance.interceptors.request.use( ...
31  )
32
33  > instance.interceptors.response.use( ...
62  )
63
64  export default instance
```

# 前端 — 后端交互封装

---

- 实际调用形式：使用统一 APIDispatcher 进行转发

```
1  import api from '@api/dispatcher'
2
3  const res = await api.getGraphByProjectId(projectId)
4  |
```



# /C 软件过程

# Swagger

- 接口文档

swagger

Select a specdefault

Swagger2演示1.0.0

[ Base URL: 121.4.125.198:8001/ ]

<http://121.4.125.198:8001/v2/api-docs>

API文档

reset-controllerReset Controller

用户控制User Controller

知识图谱智能应用App Controller

知识图谱节点操作Graph Controller

知识图谱项目操作Project Controller

Models

GET/graph/{projectId}根据id查找知识图谱

POST/graph/cascadeDeleteNode级联删除实体，会删除NodeVO及其相关关系，需要正确id

Parameters

Try it out

| Name                          | Description                                                                                                                 |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| nodeIdVO * required<br>(body) | nodeIdVO<br>Example Value   Model<br><pre>{  "nodeId": 1,  "projectId": 0}</pre> Parameter content type<br>application/json |


Responses


Response content type \*/\*


| Code | Description                                                                     |
|------|---------------------------------------------------------------------------------|
| 200  | OK<br>Example Value   Model<br><pre>{  "msg": "string",  "success": true}</pre> |
| 201  | Created                                                                         |
| 401  | Unauthorized                                                                    |
| 403  | Forbidden                                                                       |
| 404  | Not Found                                                                       |


# Jenkins


- Front-end
- Back-end
- KBQA
- 自动部署release分支


 **Jenkins**




 1


 1


 chenyanze


 注销


Dashboard


 新建任务


 用户列表


 构建历史


 项目关系

 检查文件指纹

 系统管理

 我的视图

 Lockable Resources

 新建视图

构建队列

队列中没有构建任务










构建执行状态

1 空闲

2 空闲

所有

+

| S                                                                                   | W                                                                                   | 名称 ↓              | 上次成功        | 上次失败            | 上次持续时间   |                                                                                     |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------|-------------|-----------------|----------|-------------------------------------------------------------------------------------|
|  |  | backend-pipeline  | 19 小时 - #74 | 17 天 - #41      | 1 分 58 秒 |  |
|  |  | frontend-pipeline | 16 小时 - #62 | 1 天 21 小时 - #48 | 1 分 16 秒 |  |
|  |  | kbqa-flask        | 19 小时 - #25 | 15 天 - #3       | 3 分 13 秒 |  |

图标: 小 中 大

图例

Atom feed 全部

Atom feed 失败

Atom feed 最新的构建

添加说明

# 语雀

---

## 语雀文档内容

目录

+

⋮

[api] 用户权限认证相关

功能点实现 & bug 发现与修复

[Frontend] 代码规范

- 部分 api 文档
- 功能实现&bug发现与修复
- 代码规范

目录

+ : <

[api] 用户权限认证相关

功能点实现 & bug 发... + :

[Frontend] 代码规范

## [Frontend] 代码规范

- 缩进: 2 个 space(js、.vue 文件统一)
- .vue 文件组件钩子顺序
  - name
  - components
  - data
  - computed
    - mapState、mapGetters 置顶
  - watch
  - methods
    - mapMutations、mapActions 置顶
  - 生命周期钩子(按时间顺序)
- 项目结构
  - /src 源代码目录
    - /api
      - /modules 具体路由注册(声明式接口定义)
      - axios.js Axios请求库封装
      - config.js 请求参数设置
      - dispatcher.js API自动注册实例(声明式接口定义封装)
    - /assets 静态资源

📄 .prettierrc.json 110 Bytes 📋

```
1 {
2   "singleQuote": true,
3   "tabWidth": 2,
4   "useTabs": false,
5   "endOfLine": "lf",
6   "semi": false
7 }
```

# 功能点实现 & bug 发现与修复

## 功能点 / 特性

### 前端

- Graph 图谱页面
  - GraphBoard 操作页面效果
    - ☒ 实体高亮
    - ☒ 关系端点直接点击实体
    - ☒ 实体/关系选中取消
    - ☐ 关系方向
    - ☐ 选中后，环绕实体工具列
    - ☒ 图谱缩放、平移体验优化
  - GraphEditor 实体/关系内容操作
    - ☒ 实体/关系属性展示
    - ☒ 实体/关系添加、更新、删除
    - ☒ 扩展实体属性
    - ☐ 自定义实体属性
    - ☒ 自定义颜色/分组颜色切换
    - ☒ 实体字体大小调节

- ▼ 功能点 / 特性
  - 前端
  - 后端
- ▼ bug 发现 & 修复
  - 前端
  - 后端

## 迭代三

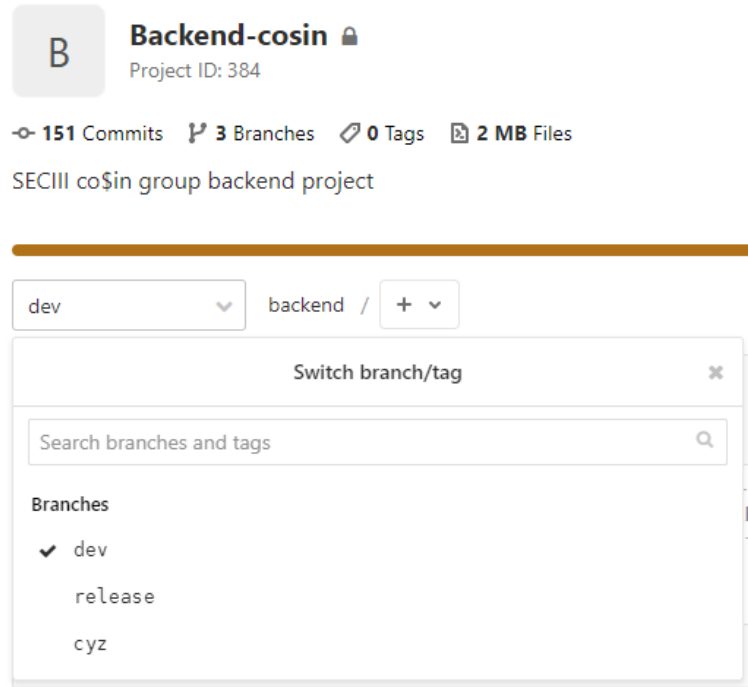
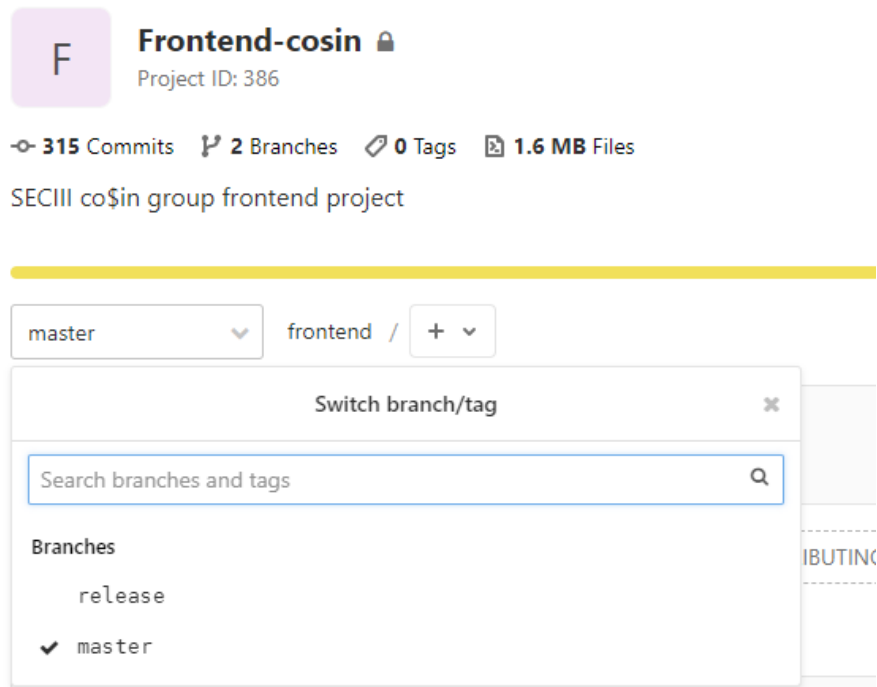
- ☒ 问答接口返回答案
- ☒ 问答接口返回图
- ☒ 项目私有/公开接口，并获得公开项目list
- ☒ 实体查询
- ☒ 关系查询
- ☒ 页数接口
- ☒ 个人项目也要分页
- ☒ 热门查询
- ☒ 全局助手

## 更新日志

- 2021/3/31
  - 为Node添加xAxis、yAxis
  - Project添加fixed
- 2021/4/1
  - 修改接口 `/graph/deleteNode`，添加异常信息
  - 添加接口 `/graph/cascadeDeleteNode`，级联删除节点
  - 添加接口 `/graph/updateNodesGraphic`，批量更新节点位置信息
- 2021/4/6

# GitLab

- 自动部署release分支
- dev / master 分支开发



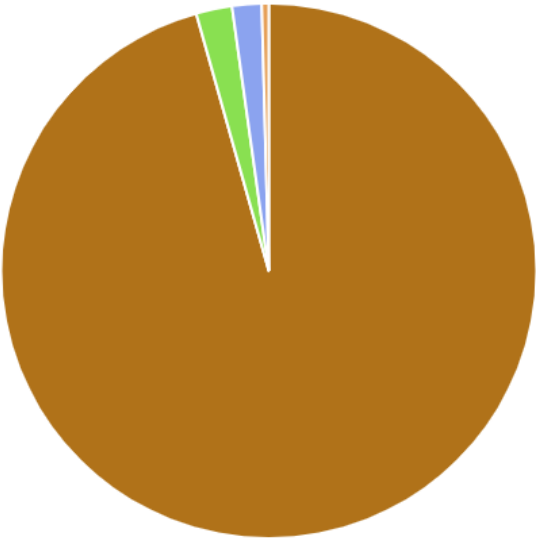
# GitLab

- 后端仓库代码饼图
- 前端仓库活跃图

## Programming languages used in this repository

Measured in bytes of code. Excludes generated and vendored code.

|        |         |
|--------|---------|
| Java   | 95.63 % |
| Shell  | 2.17 %  |
| TSQL   | 1.78 %  |
| Groovy | 0.42 %  |



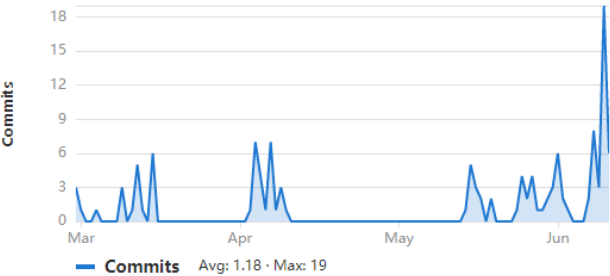
## Commits to master

Excluding merge commits. Limited to 6,000 commits.



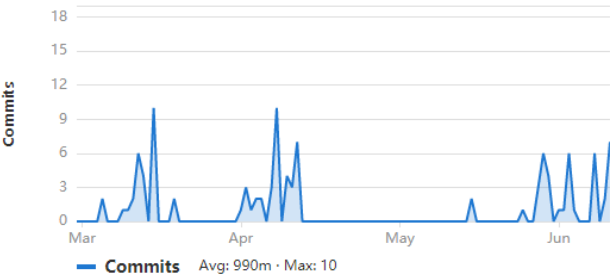
## cclin

124 commits (181250083@smail.nju.edu.cn)



## superfree

104 commits (superfreeeee@gmail.com)





# **/02 项目演示**

## **Project Exhibition**

# /03 Q&A



# THANKS

