系统设计文档

1 文档修订记录

修改人员	修改日期	修改内容	版本号
薛人玮	2021.03.10	初始化文档	v1.1
薛人玮	2021.03.15	更新服务端架构设计说明	v1.2
薛人玮	2021.03.19	更新前端架构设计说明，api部分	v1.3
薛人玮	2021.03.25	更新6.1，6.2，增加“用户”数据持久化层	v2.1
薛人玮	2021.04.02	更新5.3.1，5.3.2，5.3.3	v2.2
薛人玮	2021.04.11	更新5.3.4，5.3.5，5.3.6，5.3.7	v2.3
薛人玮	2021.04.18	检查，完善	v2.4

2 文档总体介绍

2.1 目的

本文档详细完成对COIN知识图谱项目的设计，达到指导详细设计和开发的目的，采用若干架构视图描述系统的不同方面，以便表示构造系统所需要的重要架构决策。

2.2 范围

本文档的读者是OurCOIN团队内部的开发和管理人员，参考了RUP的《软件架构文档模板》，用于指导下一循环的代码开发和测试工作。

2.3 参考

《需求规格说明书》，OurCOIN Team；

《软件架构文档模板》，Rational Software Corporation，2002

The Object Management Group（OMG），The Unified Modeling Languages Specification v1.4，2003

2.4 词汇表

词汇名称	词汇含义	备注
COIN	知识图谱定义及可视化系统	即 A system for COnstructing and vIsualizing kNowledge graph

3 项目总体描述

3.1 需求概述

- 构建一个系统平台
- 在线编辑基本图元
- 可视化图谱构建
- 覆盖基本图谱元素
- 初步具备一定知识推理能力

3.2 运行环境

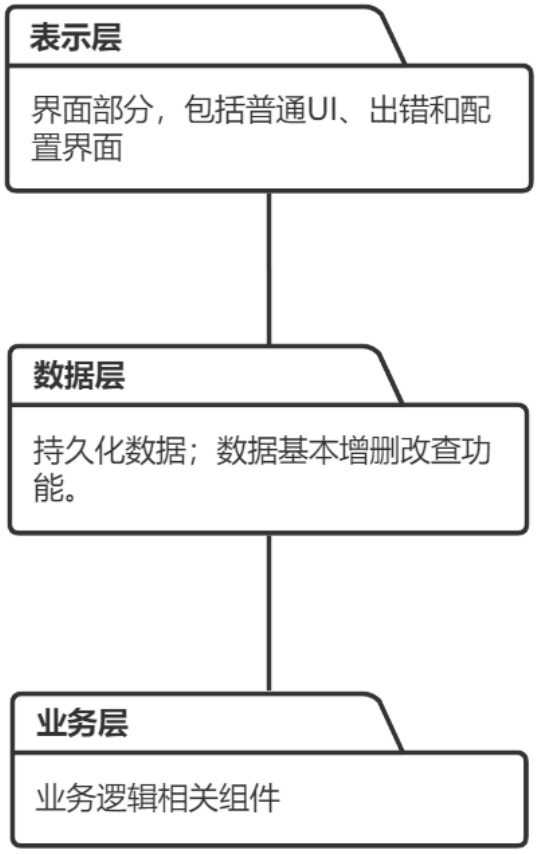
前端运行在主流浏览器上，包括 Chrome、Firefox、Edge 等；后端运行在阿里云服务器上。

3.3 条件与限制

- CON1：采用 Java 语言及其它相关的 Web 开发
- CON2：系统使用的是基于 Web 的数据库应用系统
- CON3：项目需要完整的单元测试、集成测试、系统级测试
- CON4：项目后期会增加需求及开放式功能
- CON5：将个人工程行为尽可能地记录在 Gitlab 上项目设计文档
- CON6：每次迭代产品均必须完成部署（使用 Jenkins 实现一键部署）

4 系统架构

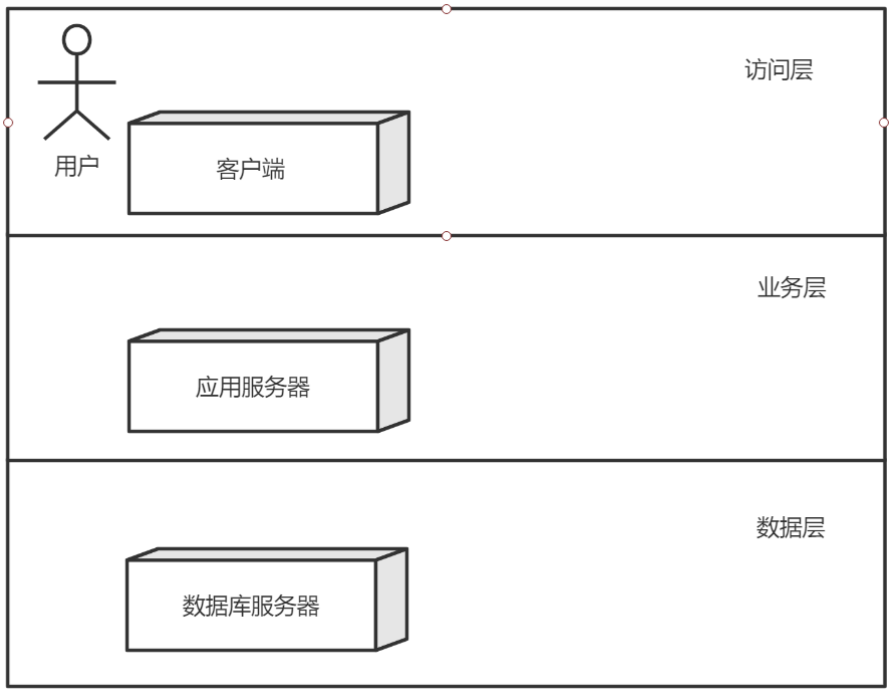
4.1 分层架构



系统划分为以下3个逻辑层次。

1. 表示层：用于前台界面展示和配置的层次。
2. 业务层：包含业务控制和逻辑的层次。
3. 数据层：定义和存储系统中相关数据的层次。

4.2 物理层次

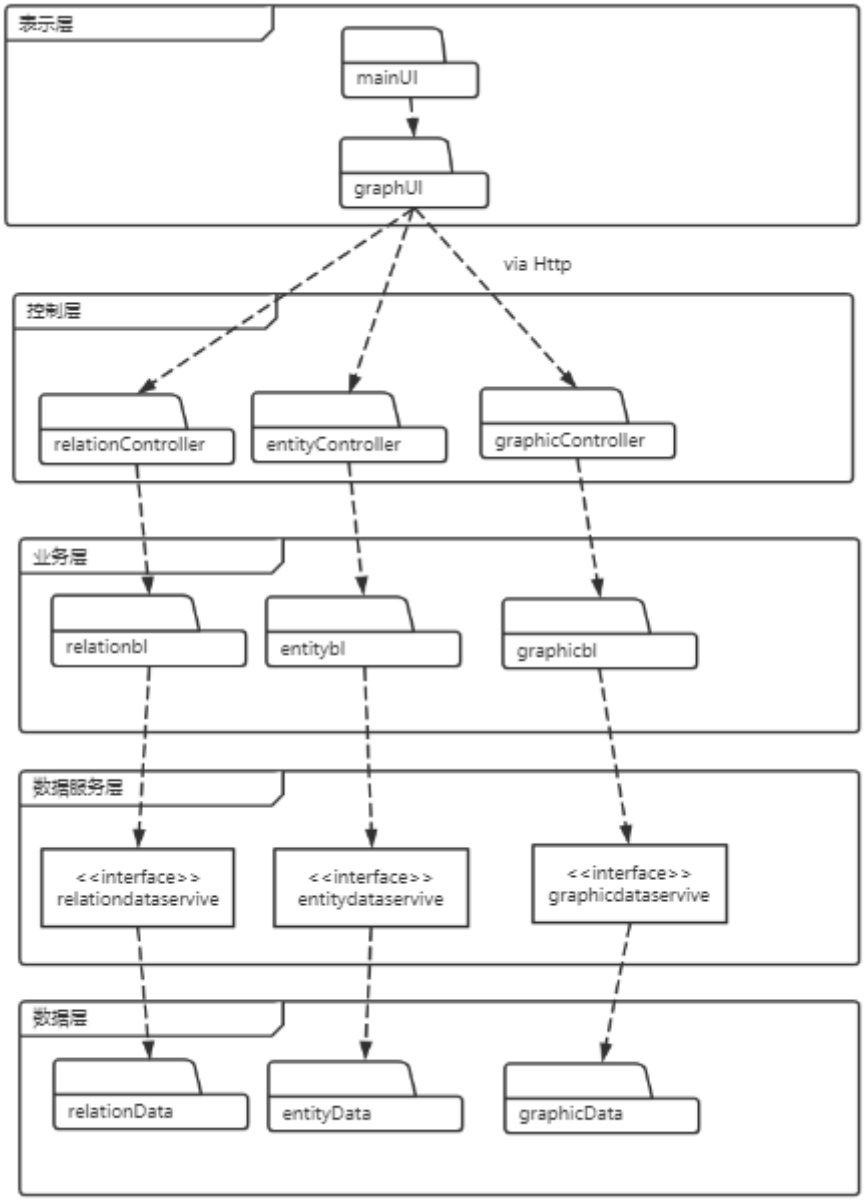


系统部署在以下三个物理层：

1. 访问层：用于用户访问系统的层次。

2. 业务层：部署业务控制和逻辑的层次。
3. 数据层：部署和存储系统中相关数据的层次。

4.3 架构设计



5 系统设计描述主体

5.1 客户端架构分解

技术栈：Vue.js + Vue-router + ElementUI + axios

主要实现本系统的前端页面展示。在基于 vue-cli 3 的模板基础上加入了 vue-router 等配套设施，由 vue-router 来控制 views 中各页面的跳转。开发时采用组件化策略，降低与页面之间的耦合，并使用 ElementUI 组件库进行美化，views 中各页面则复用已开发好的组件。 使用 axios 与后端进行数据通信。

5.2 服务端架构分解

技术栈：Spring + MySQL + Jenkins（Docker）

主要负责本系统的业务处理及数据处理相关。使用Spring作为本项目的Web后端框架，由MySQL负责数据的持久化以及设计的数据的增删改查功能。同时通过在阿里云服务器中配置Docker容器，在容器中配置Jenkins实现快速构建和部署。

5.3 接口定义

5.3.1 添加实体

5.3.1.1 接口信息

接口名称	接口路径	请求协议	请求方法	状态
addEntityApi	${api.entity}/addEntity	HTTP	post	正常启用

5.3.1.2 参数信息

请求参数

参数名	说明	是否必需	类型	备注
entity	实体信息	是	[Object]	name，catagory，x，y，symbol，symbolSize，color，labelFontSize，borderColor，borderWidth，eid，gid

5.3.2 删除实体

5.3.2.1 接口信息

接口名称	接口路径	请求协议	请求方法	状态
deleteEntityApi	${api.entity}/deleteEntity	HTTP	post	正常启用

5.3.2.2 参数信息

请求参数

参数名	说明	是否必需	类型	备注
entity	实体信息	是	[Object]	name，catagory，x，y，symbol，symbolSize，color，labelFontSize，borderColor，borderWidth，eid，gid

5.3.3 添加关系

5.3.3.1 接口信息

接口名称	接口路径	请求协议	请求方法	状态
addRelationApi	${api.relation}/addRelation	HTTP	post	正常启用

5.3.3.2 参数信息

请求参数

参数名	说明	是否必需	类型	备注
relation	关系信息	是	[Object]	id，source，target，value，gid，lineStyle

5.3.4 删除关系

5.3.4.1 接口信息

接口名称	接口路径	请求协议	请求方法	状态
deleteRelationApi	${api.relation}/deleteRelation	HTTP	post	正常启用

5.3.4.2 参数信息

请求参数

参数名	说明	是否必需	类型	备注
relation	关系信息	是	[Object]	id，source，target，value，gid，lineStyle

5.3.5 保存图谱

5.3.5.1 接口信息

接口名称	接口路径	请求协议	请求方法	状态
saveGraphApi	${api.graph}/saveGraph	HTTP	post	正常启用

5.3.5.2 参数信息

请求参数

参数名	说明	是否必需	类型	备注
graph	图谱信息	是	[Object]	gid，name，entities，relations，uid，directory

5.3.6 根据ID获得图谱

5.3.6.1 接口信息

接口名称	接口路径	请求协议	请求方法	状态
getGraphByIdApi	${api.graph}/getGraphById/${data}	HTTP	get	正常启用

5.3.6.2 参数信息

GET参数

参数名	说明	是否必需	类型	备注
gid	图谱id	是	[Int]	

返回参数

参数名	说明	是否必需	类型	备注
graph	图谱信息	是	[Object]	gid，name，entities，relations，uid，directory

5.3.7 根据ID获得实体

5.3.7.1 接口信息

接口名称	接口路径	请求协议	请求方法	状态
getEntityByEidApi	${api.entity}/getEntityByEid/${data}	HTTP	get	正常启用

5.3.7.2 参数信息

GET参数

参数名	说明	是否必需	类型	备注
eid	实体id	是	[Int]	

返回参数

参数名	说明	是否必需	类型	备注
entity	实体信息	是	[Object]	name，catagory，x，y，symbol，symbolSize，color，labelFontSize，borderColor，borderWidth，eid，gid

6 信息视角

6.1 数据持久化对象

系统的PO类就是对应的相关的实体类，在此只做简单的介绍。

类名	包含的属性
entityPO	实体id，图id，名字
graphicPO	图id，实体关系列表，名字
relationPO	关系id，图id，名字
userPO	用户id，用户名，密码，图谱

持久化用户对象userPO的定义如下所示：

```
package com.example.demo.po;


import java.util.ArrayList;

public class User {
    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void setGraphs(ArrayList<Integer> graphs) {
        this.graphs = graphs;
    }

    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }
```

```
    public ArrayList<Integer> getGraphs() {
        return graphs;
    }

    public Integer getUid() {
        return uid;
    }

    public void setUid(Integer uid) {
        this.uid = uid;
    }

    private String userName;

    private String password;

    private ArrayList<Integer> graphs=new ArrayList<>();

    private Integer uid;

}
```

6.2 数据库表

数据库中包含entity表、graphic表、relation表，user表

附录

索引