COIN 项目设计文档

1 文档修改历史

修改人员	日期	修改原因	版本号
郭增嘉	2021.03.08	完成引言、逻辑视角	V1.0
郭增嘉	2021.03.10	添加图示与部分接口	V1.2
郭增嘉	2021.03.20	添加Jenkins部署pipeline脚本	v1.3

2 引言

2.1 编制目的

本文档提供COIN知识图谱可视化系统的软件架构的概要设计，采用若干架构层面识图描述系统达到指导详细设计与开发的目的，同时实现与测试人员及用户的沟通。
本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

2.2 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	无

2.3 参考资料

[1] CSEIII 01-COIN项目介绍文档

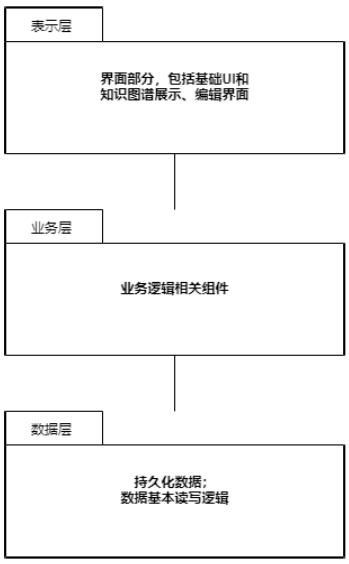
[2] COIN需求规格说明文档

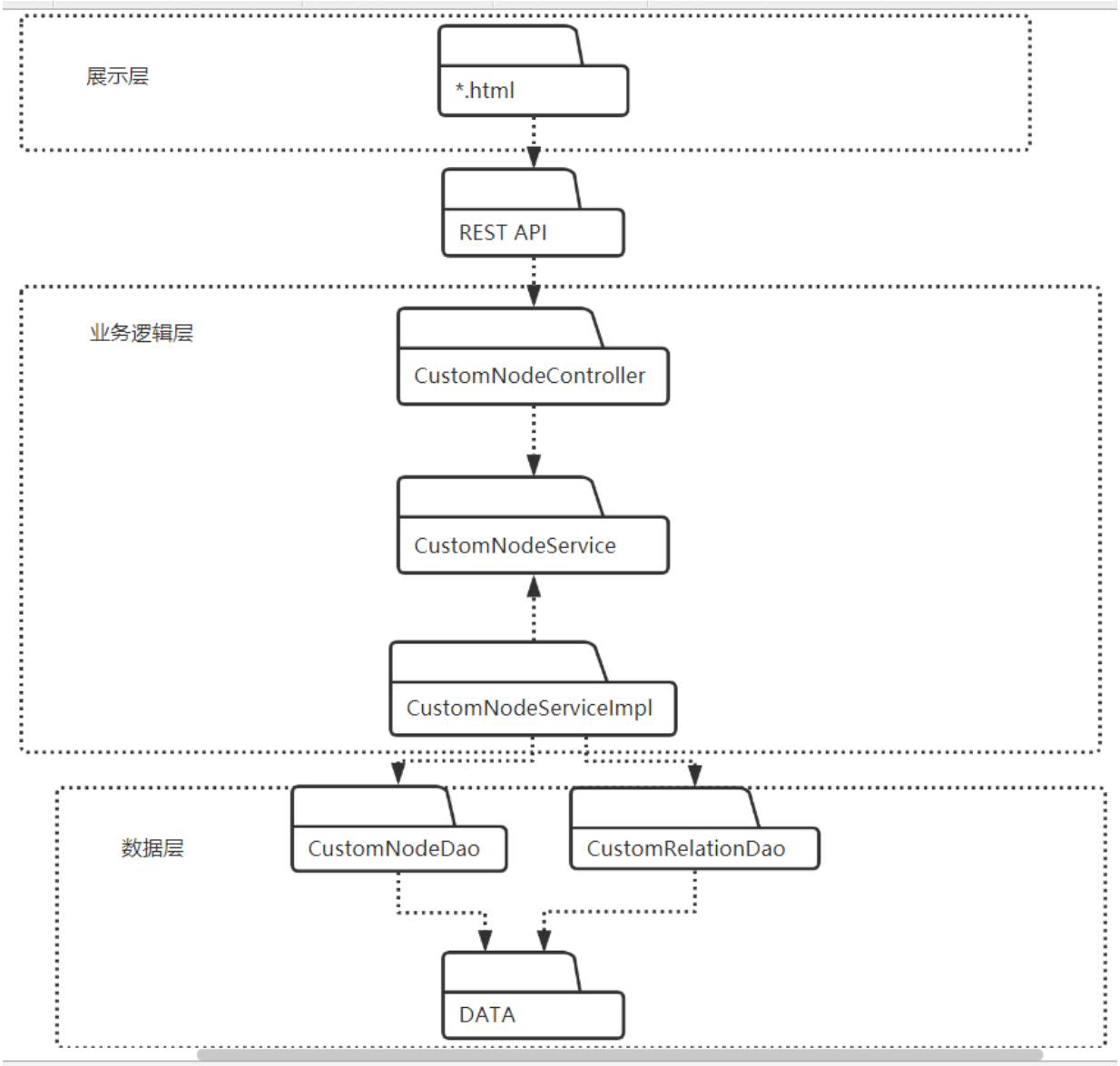
[3] 《软件工程与计算（卷三） 团队与软件开发实践》，骆斌、刘嘉、张瑾玉、黄蕾，机械工业出版社

3 逻辑视角

3.1 系统的分层架构

COIN系统中，选择了分层体系结构风格，将系统分为3层（展示层、业务逻辑层、数据层）能够更好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。





3.2 系统架构设计

- 系统架构中的对象
  - UI对象，负责处理系统数据的展现和用户的交互
  - CustomNodeController对象，控制器负责获取用户输入，并调用ServiceImpl接口的服务
  - CustomNodeService对象，业务逻辑服务的抽象接口，获取从数据端组装好的数据
  - CustomNodeServiceImpl对象，负责Service接口的实现
  - CustomNodeDao对象，操作neo4j数据库节点label的接口，获取数据
  - CustomRelationDao对象，操作neo4j数据库关系label的接口，获取数据
  - CustomNode对象，用于节点的数据封装
  - CustomRelation对象，用于关系的数据封装
- 系统对外接口设计

提供的服务（供接口）		
CustomNodeController.createNode	语法	CustomNode createNode(String name,int group);
	API调用格式	/api/customNode/create/{group}/{name}
	前置条件	用户请求添加一个节点
	后置条件	调用service，在Dao层完成添加节点的逻辑
CustomNodeController.deleteNodeById	语法	boolean deleteNodeById(long id)
	API调用格式	/api/customNode/delete/{id}
	前置条件	用户请求删除一个节点
	后置条件	调用service，在Dao层完成删除节点的逻辑
CustomNodeController.editNodeById	语法	boolean editNodeById(long id,String newName,int newGroup)
	API调用格式	/api/customNode/edit/{id}/{newGroup}/{newName}
	前置条件	用户请求编辑一个节点
	后置条件	调用service，在Dao层完成编辑节点的逻辑
CustomNodeController.retrieveCustomNodeById	语法	CustomNode retrieveCustomNodeById(long id)

	API调用格式	/api/customNode/retrieve/{id}
	前置条件	用户请求获取一个节点
	后置条件	调用service，在Dao层完成获取节点的逻辑
CustomNodeController.addRelation	语法	CustomRelation addRelation(long fromId, long toId,String type)
	API调用格式	/api/customNode/addRelation/{fromId}/{toId}/{type}
	前置条件	用户请求添加一个关系
	后置条件	调用service，在Dao层完成添加关系的逻辑
CustomNodeController.deleteRelation	语法	boolean deleteRelation(long fromId,long toId)
	API调用格式	/api/customNode/deleteRelation/{fromId}/{toId}
	前置条件	用户请求删除一个关系
	后置条件	调用service，在Dao层完成删除关系的逻辑
CustomNodeController.deleteAll	语法	boolean deleteAll()
	API调用格式	/api/customNode/deleteAll

	前置条件	用户请求删除所有节点与关系
	后置条件	调用service，在Dao层完成删除节点与关系的逻辑
CustomNodeController.editRelation	语法	boolean editRelation(long fromId,long toId,String type,int value)
	API调用格式	/api/customNode/editRelation/{fromId}/{toId}/{type}/{value}
	前置条件	用户请求编辑关系的类型与属性值
	后置条件	调用service，在Dao层完成编辑关系的类型与属性值的逻辑
CustomNodeController.retrieveRelation	语法	CustomRelation retrieveRelation(long fromId, long toId)
	API调用格式	/api/customNode/retrieveRelation/{fromId}/{toId}
	前置条件	用户请求获取关系
	后置条件	调用service，在Dao层完成获取关系的逻辑
CustomNodeController.retrieveAllRelation	语法	List <CustomRelation> retrieveAllRelation()
	API调用格式	/api/customNode/retrieveAllRelation
	前置条件	用户请求获取所有关系
	后置条件	调用service，在Dao层完成获取所有关系的逻辑

CustomNodeController.retrieveAllNode	语法	List <CustomNode> retrieveAllNode()
	API调用格式	/api/customNode/retrieveAllNode
	前置条件	用户请求获取所有节点
	后置条件	调用service，在Dao层完成获取所有节点的逻辑

4 Jenkins部署

前端项目Pipeline脚本

```
pipeline {
    agent any
    options{
        timestamps()
        timeout(10)
    }
    stages {
        stage('git') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '*/master']], extensions: [],
userRemoteConfigs: [[credentialsId: 'e6580780-e75d-4e95-a44d-ed8e783caf03', url:
'http://212.129.149.40/181250076_ac/frontend-coin.git']]])
                 }
        }
        stage('build') {
            steps {
                nodejs('nodejs'){


                }
                sh 'npm install'
                sh 'npm run build'
            }
        }
        stage('mv&del') {
            steps {
                sh 'rm -rf /tomcat/apache-tomcat-8.5.64/webapps/dist'
                sh 'mv /var/lib/jenkins/workspace/frontend_coin/dist /tomcat/apache-tomcat-
8.5.64/webapps'
            }
        }
        stage('clean'){
            steps{
                cleanWs()
            }

        }
    }

}
```

后端项目Pipeline脚本

```
pipeline {
    agent any
    options{
        timestamps()
```

```
        timeout(10)
    }
    stages {
        stage('git') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '*/master'], [name: '*/release']],
extensions: [], userRemoteConfigs: [[credentialsId: 'e6580780-e75d-4e95-a44d-ed8e783caf03', url:
'http://212.129.149.40/181250076_ac/backend-projectname.git']]])
            }
        }
        stage('build') {
            steps {
                sh 'mvn clean package -DskipTests'
            }
        }
        stage('publish code') {
            steps {
                deploy adapters: [tomcat8(credentialsId: 'bd1c8fe9-582f-49eb-bdce-e13e8705a567',
path: '', url: 'http://101.200.52.46:8080/')], contextPath: null, war: 'target/*.war'
            }
        }
    }
}
```