**前端标准**

1.目模板 vue cli 3.0

2.UI界面 elementUI+

3.路由管理 vue-router

4.网络请求 axios

5.状态管理 Vuex

组件规范：

1.为组件样式设置作用域

2.将每个组件单独分成文件

3.由HTML大小写不敏感，模板中组件名采用横线连接

4.指令缩写 (用 : 表示 v-bind:、用 @ 表示 v-on: 和用 # 表 示 v-slot:) 应该要么都用要么都不用

5. 组件模板应该只包含简单的表达式，复杂的表达式则应该 重构为计算属性或方法

技术栈： V u e 使用Vue全家桶进行开发，脚手架版本@vue/cli 4.1.2+，vue版本3.x，

统一开发工具 VSCode 1.35

使用ElementUI+作为UI框架

公用组件和工具函数通过单元测试 对工具函数进行测试，且添加公用配置文件时不能修 改他人提交的配置内容

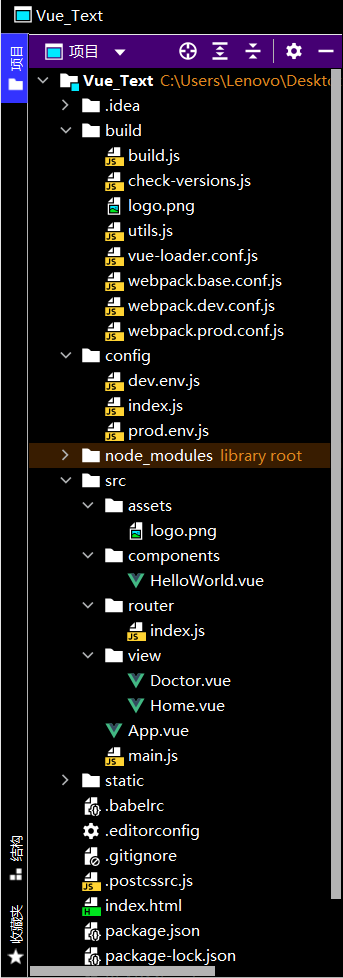
统一路由编写规范 根据路由去中心化概念，向router中提交路由模块文 件夹，路由设置与页面组件名称保持一致

统一Vuex编写规范 对于项目通用资源，由store下公用配置文件管理， 对于单功能所需通用资源，向store提交资源模块文 件夹

统一组件编写规范 根据页面作为逻辑向view提交页面文件夹，再在页 面文件夹中书写页面组件以及可能使用的页面重用组 件

统一Vu e风格规范 参考官方网站中风格指南

统一网络请求模拟测试工具 使用Mock.js模拟网络请求



**后端标准**

统一开发工具：IDEA2021.2.1

JDK统一为1.8.0（群资料下载）

框架为SpringBoot+MyBatis

SpingBoot用2.5.4版本

测试统一JUnity

日志统一log4j

后端测试数据库用postman

统一用maven项目

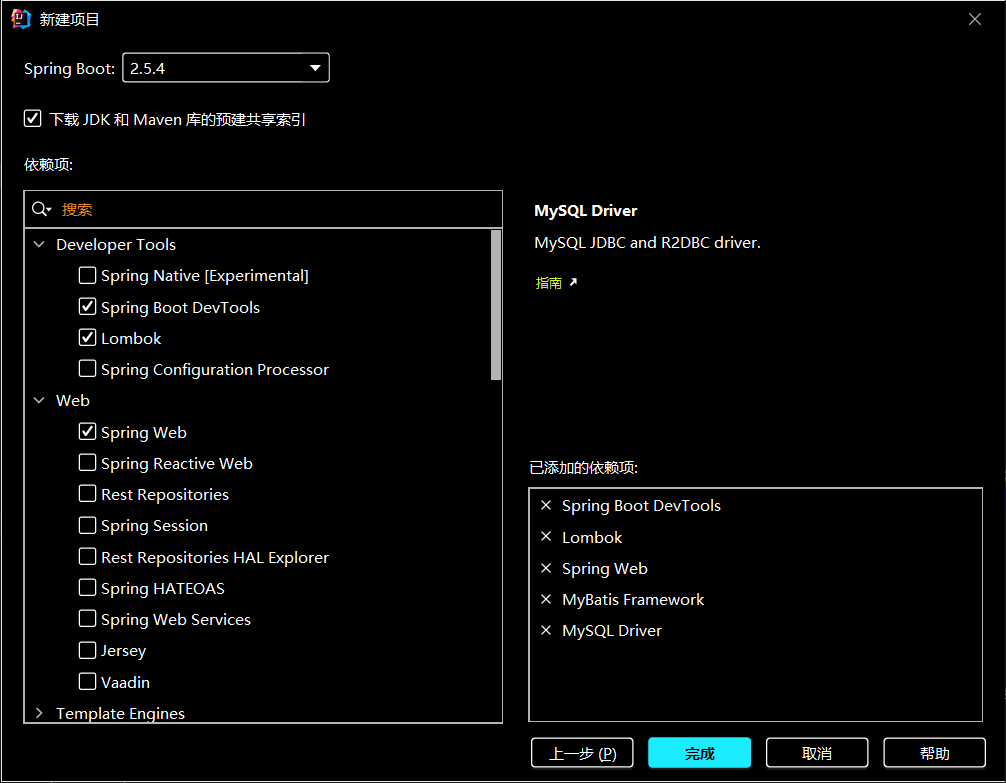
Maven库由项目经理统一搭配好，上传到git，各组员从git下载。

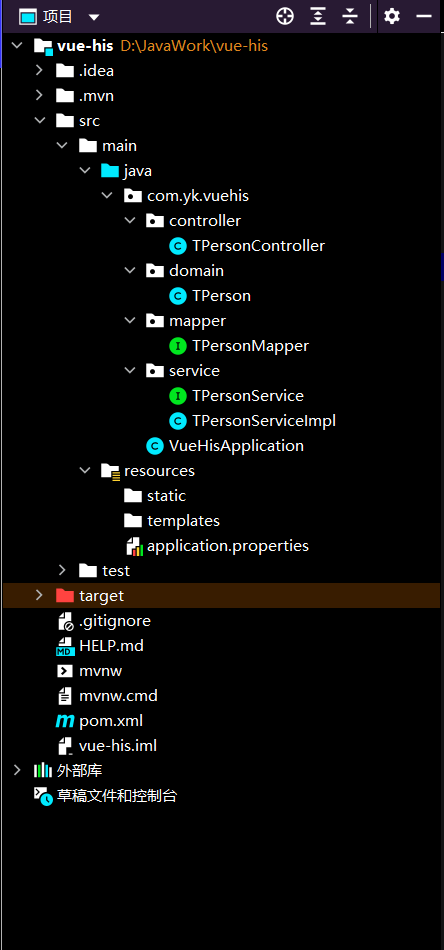
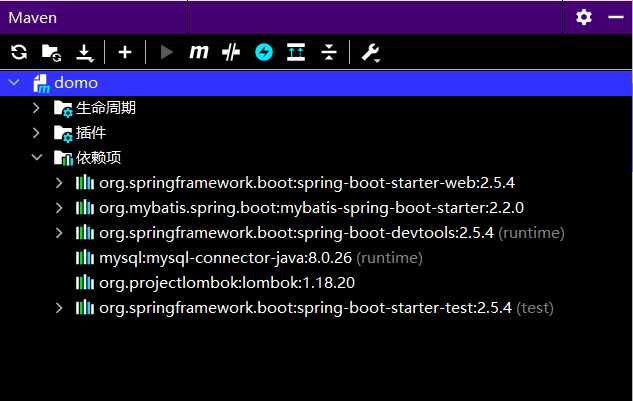
数据库用Mysql8.0.26（比较熟悉）

统一下载阿里规范插件

如下图：







**编码规范：**

1. 【强制】代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

反例：\_name / \_\_name / $name / name\_ / name$ / name\_\_

2. 【强制】代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。

说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式

也要避免采用。

正例：alibaba / taobao / youku / hangzhou 等国际通用的名称，可视同英文。

反例：DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3

3. 【强制】类名使用 UpperCamelCase 风格，但以下情形例外：DO / BO / DTO / VO / AO /

PO 等。

正例：MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion

反例：macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion

4. 【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从 驼峰形式。

正例： localValue / getHttpMessage() / inputUserId

5. 【强制】常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。

正例：MAX\_STOCK\_COUNT

反例：MAX\_COUNT

6. 【强制】抽象类命名使用 Abstract 或 Base 开头；异常类命名使用 Exception 结尾；测试类

命名以它要测试的类名开始，以 Test 结尾。

7. 【强制】类型与中括号紧挨相连来定义数组。

正例：定义整形数组 int[] arrayDemo;

反例：在 main 参数中，使用 String args[]来定义。

8. 【强制】POJO 类中布尔类型的变量，都不要加 is 前缀，否则部分框架解析会引起序列化错误。

反例：定义为基本数据类型 Boolean isDeleted；的属性，它的方法也是 isDeleted()，框架在反向解析的时候，“误以为”对应的属性名称是 deleted，导致属性获取不到，进而抛出异常。

9. 【强制】包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用

单数形式，但是类名如果有复数含义，类名可以使用复数形式。

正例：应用工具类包名为 com.alibaba.ai.util、类名为 MessageUtils（此规则参考 spring

的框架结构）

10. 【强制】杜绝完全不规范的缩写，避免望文不知义。

反例：AbstractClass“缩写”命名成 AbsClass；condition“缩写”命名成 condi，此类随

意缩写严重降低了代码的可阅读性。

11. 【推荐】为了达到代码自解释的目标，任何自定义编程元素在命名时，使用尽量完整的单词 组合来表达其意。

正例：从远程仓库拉取代码的类命名为 PullCodeFromRemoteRepository。

反例：变量 int a; 的随意命名方式。

12. 【推荐】如果模块、接口、类、方法使用了设计模式，在命名时体现出具体模式。

说明：将设计模式体现在名字中，有利于阅读者快速理解架构设计理念。

正例：public class OrderFactory;

public class LoginProxy;

public class ResourceObserver;

13. 【推荐】接口类中的方法和属性不要加任何修饰符号（public 也不要加），保持代码的简洁

性，并加上有效的 Javadoc 注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是 与接口方法相关，并且是整个应用的基础常量。

正例：接口方法签名 void f();

接口基础常量 String COMPANY = "alibaba";

反例：接口方法定义 public abstract void f();

说明：JDK8 中接口允许有默认实现，那么这个 default 方法，是对所有实现类都有价值的默认实现。

14. 接口和实现类的命名有两套规则：

1）【强制】对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部的实现类用 Impl 的后缀与接口区别。

正例：CacheServiceImpl 实现 CacheService 接口。

2）【推荐】如果是形容能力的接口名称，取对应的形容词为接口名（通常是–able 的形式）。

正例：AbstractTranslator 实现 Translatable。

15. 【参考】枚举类名建议带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。

说明：枚举其实就是特殊的常量类，且构造方法被默认强制是私有。

正例：枚举名字为 ProcessStatusEnum 的成员名称：SUCCESS / UNKNOWN\_REASON。

16. 【参考】各层命名规约：

A) Service/DAO 层方法命名规约

1） 获取单个对象的方法用 get 作前缀。

2） 获取多个对象的方法用 list 作前缀。

3） 获取统计值的方法用 count 作前缀。

4） 插入的方法用 save/insert 作前缀。

5） 删除的方法用 remove/delete 作前缀。

6） 修改的方法用 update 作前缀。

**常量定义 ：**

1. 【强制】不允许任何魔法值（即未经预先定义的常量）直接出现在代码中。

反例：String key = "Id#taobao\_" + tradeId;

cache.put(key, value);

2. 【强制】long 或者 Long 初始赋值时，使用大写的 L，不能是小写的 l，小写容易跟数字 1 混

淆，造成误解。

说明：Long a = 2l; 写的是数字的 21，还是 Long 型的 2?

3. 【推荐】不要使用一个常量类维护所有常量，按常量功能进行归类，分开维护。

说明：大而全的常量类，非得使用查找功能才能定位到修改的常量，不利于理解和维护。

正例：缓存相关常量放在类 CacheConsts 下；系统配置相关常量放在类 ConfigConsts 下。

4. 【推荐】常量的复用层次有五层：跨应用共享常量、应用内共享常量、子工程内共享常量、包

内共享常量、类内共享常量。

1） 跨应用共享常量：放置在二方库中，通常是 client.jar 中的 constant 目录下。

2） 应用内共享常量：放置在一方库中，通常是子模块中的 constant 目录下。

反例：易懂变量也要统一定义成应用内共享常量，两位攻城师在两个类中分别定义了表示

“是”的变量：

类 A 中：public static final String YES = "yes";

类 B 中：public static final String YES = "y";

A.YES.equals(B.YES)，预期是 true，但实际返回为 false，导致线上问题。

3） 子工程内部共享常量：即在当前子工程的 constant 目录下。

4） 包内共享常量：即在当前包下单独的 constant 目录下。

5） 类内共享常量：直接在类内部 private static final 定义。

5. 【推荐】如果变量值仅在一个固定范围内变化用 enum 类型来定义。

说明：如果存在名称之外的延伸属性使用 enum 类型，下面正例中的数字就是延伸信息，表示 一年中的第几个季节。

正例：

public enum SeasonEnum {

SPRING(1), SUMMER(2), AUTUMN(3), WINTER(4);

int seq;

SeasonEnum(int seq){

this.seq = seq;

}

}

**代码格式 ：**

1. 【强制】大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；如果

是非空代码块则：

1） 左大括号前不换行。

2） 左大括号后换行。

3） 右大括号前换行。

4） 右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。

2. 【强制】 左小括号和字符之间不出现空格；同样，右小括号和字符之间也不出现空格。详见

第 5 条下方正例提示。

反例：if (空格 a == b 空格)

3.

【强制】if/for/while/switch/do 等保留字与括号之间都必须加空格。

4.

【强制】任何二目、三目运算符的左右两边都需要加一个空格。

说明：运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号等。

5. 【强制】采用 4 个空格缩进，禁止使用 tab 字符。

说明：如果使用 tab 缩进，必须设置 1 个 tab 为 4 个空格。IDEA 设置 tab 为 4 个空格时，

请勿勾选 Use tab character；而在 eclipse 中，必须勾选 insert spaces for tabs。

正例： （涉及 1-5 点）

public static void main(String[] args) {

// 缩进 4 个空格 String say = "hello";

// 运算符的左右必须有一个空格

int flag = 0;

// 关键词 if 与括号之间必须有一个空格，括号内的 f 与左括号，0 与右括号不需要空格

if (flag == 0) {

System.out.println(say);

}

// 左大括号前加空格且不换行；左大括号后换行

if (flag == 1) {

System.out.println("world");

// 右大括号前换行，右大括号后有 else，不用换行

} else {

System.out.println("ok");

// 在右大括号后直接结束，则必须换行

}

}

6. 【强制】注释的双斜线与注释内容之间有且仅有一个空格。

正例：

// 这是示例注释，请注意在双斜线之后有一个空格

String ygb = new String();

7.

【强制】单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：

1） 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。

2） 运算符与下文一起换行。

3） 方法调用的点符号与下文一起换行。

4） 方法调用时，多个参数，需要换行时，在逗号后进行。

5） 在括号前不要换行，见反例。

正例：

StringBuffer sb = new StringBuffer();

// 超过 120 个字符的情况下，换行缩进 4 个空格，点号和方法名称一起换行

sb.append("zi").append("xin")...

.append("huang")...

.append("huang")...

.append("huang");

反例：

StringBuffer sb = new StringBuffer();

// 超过 120 个字符的情况下，不要在括号前换行

sb.append("zi").append("xin")...append

("huang");

// 参数很多的方法调用可能超过 120 个字符，不要在逗号前换行

method(args1, args2, args3, ...

, argsX);

8. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。

正例：下例中实参的"a",后边必须要有一个空格。

method("a", "b", "c");

9. 【强制】IDE 的 text file encoding 设置为 UTF-8; IDE 中文件的换行符使用 Unix 格式，

不要使用 Windows 格式。

10. 【推荐】没有必要增加若干空格来使某一行的字符与上一行对应位置的字符对齐。

正例：

int a = 3;

long b = 4L;

float c = 5F;

StringBuffer sb = new StringBuffer();

说明：增加 sb 这个变量，如果需要对齐，则给 a、b、c 都要增加几个空格，在变量比较多的

情况下，是非常累赘的事情。

11. 【推荐】不同逻辑、不同语义、不同业务的代码之间插入一个空行分隔开来以提升可读性。

说明：没有必要插入多个空行进行隔开。