冲刺安排

> 前言

大家先从`gitee`上clone项目，在自己电脑上跑起来。

6.21 星期天

PS:要是大家不能看懂`vue`，页面则按`XXX.html`，`XXX.css`，`XXX.js`放在一个文件夹内发我

`移动端加入我们界面`：邹翰林

`移动端需求希望界面`：资煌

`需求希望-后台`：邓畅伟

`移动端评分界面`：王颖

`评分-后台`：邓畅伟

`移动端team界面`：唐良秀

`移动端后勤页面`：何亚男

`移动端mi界面`：肖雨馨

`移动端作息时间界面`：胡雨婕

`后台接口测试，登录http:localhost:9898/swagger-ui.html测试以前写的`：陈宣任

`博客总结`：尹筝弈，陈星捷

6.22 星期一

`写完移动端剩余界面`：肖雨馨、胡雨婕、资煌、王颖

`美化界面，修改设计移动端展示的图片以及布局`：唐良秀、何亚男、邹翰林

`界面交互测试`：陈宣任

`博客总结`：尹筝弈，陈星捷

6.23 星期二

PS：今天是修修改改的一天！经过前面两天的冲刺，必然会出现一些大大小小的问题，今天的任务就是修复这些问题，完成这些问题后，`项目进入中点`。

`提前把问题总结6.23 08:00前`：陈宣任

`问题修复`：肖雨馨、胡雨婕、资煌、王颖、唐良秀、何亚男、邹翰林、邓畅伟

`博客总结`：尹筝弈，陈星捷

6.24 星期三

公众号自动回复-后台：邓畅伟

公众号菜单-后台：邓畅伟

公众号自动回复-管理界面：肖雨馨、胡雨婕、资煌

公众号菜单-管理界面：王颖、唐良秀、何亚男、邹翰林

接口测试：陈宣任

博客总结：尹筝弈，陈星捷

6.25 星期四

后台登录界面：胡雨婕、资煌、王颖、唐良秀、何亚男、邹翰林

vue js router：邓畅伟 、肖雨馨

博客总结：尹筝弈，陈星捷

测试：陈宣任

6.26 星期五

bug修复：邓畅伟

界面美化：肖雨馨、胡雨婕、资煌、王颖、唐良秀、何亚男、邹翰林

端口测试：陈宣任

JMeter压力测试：陈宣任

博客总结：尹筝弈，陈星捷

6.27 星期六

项目总结 + 博客撰写

团队代码规范

一、命名风格

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。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 / UID 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的名称开始，以 Test 结尾。
7. 【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.core.util、类名为 MessageUtils（此规则参考 spring 的框架结构）
8. 【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。
9. 【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：在 JDK 中，表达原子更新的类名为：AtomicReferenceFieldUpdater。
    - 反例：变量 int a 的随意命名方式。

二、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：
    - 左大括号前不换行。
    - 左大括号后换行。
    - 右大括号前换行。
    - 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】if/for/while/switch/do 等保留字与括号之间都必须加空格。
3. 【强制】采用 4 个空格缩进，禁止使用 tab 字符。

三、注释规约

1. 【强制】类、类属性、类方法的注释必须使用 Javadoc 规范，使用/ *内容* /格式，不得使用 // xxx 方式。
    - 说明：在 IDE 编辑窗口中，Javadoc 方式会提示相关注释，生成 Javadoc 可以正确输出相应注释；在 IDE 中，工程调用方法时，不进入方法即可悬浮提示方法、参数、返回值的意义，提高阅读效率。
2. 【强制】所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、异常说明外，还必须指出该方法做什么事情，实现什么功能。
    - 说明：对子类的实现要求，或者调用注意事项，请一并说明。
3. 【强制】所有的类都必须添加创建者和创建日期。
4. 【强制】方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释 使用/* */注释，注意与代码对齐。
5. 【强制】所有的枚举类型字段必须要有注释，说明每个数据项的用途。
6. 【推荐】与其“半吊子”英文来注释，不如用中文注释把问题说清楚。专有名词与关键字保持英文原文即可。
    - 反例：“TCP连接超时”解释成“传输控制协议连接超时”，理解反而费脑筋。

7. 【推荐】代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑等的修改。
    - 说明：代码与注释更新不同步，就像路网与导航软件更新不同步一样，如果导航软件严重滞后，就失去了导航的意义。
8. 【参考】谨慎注释掉代码。在上方详细说明，而不是简单地注释掉。如果无用，则删除。
    - 说明：代码被注释掉有两种可能性：
        - 后续会恢复此段代码逻辑。
        - 永久不用。
    - 前者如果没有备注信息，难以知晓注释动机。后者建议直接删掉（代码仓库保存了历史代码）。

团队成员贡献

分支介绍

1. 主分支：`master`
2. 开发分支：`dev`

贡献流程

1. 项目成员「开发者」`clone` 项目，在本地建立自己功能分支

```
git clone 项目 git 地址                          # 克隆项目
git checkout -b dev origin/dev                  # 以远程的dev分支创建本地dev分支
git checkout -b feature-[name-desc] dev         # 从dev分支建立自己的功能分支
```

2. 在自己的分支上进行开发：`git add`，`git commit` 等，注意此时不要 push 到远程分支（origin）。
3. 功能完成后可以直接合并本地的 `dev` 分支后 `push` 到远程仓库，合并的时候很大几率发生冲突，此时需要 merge，merge的时候确保不影响项目其他成员，如果多个人都操作了同一个类，最好当面确认后在进行修改。等合并完成确认无误后，删除本地开发分支

```
git checkout dev                                # 切换到dev分支
git pull origin develop                         # 确保本地 developer 分支为最新的
git merge feature-[name-desc]                   # 合并功能分支
git push                                        # 提交到远程仓库
git branch -d feature-[name-desc]               # 删除本地分支，谨慎操作!!!
```