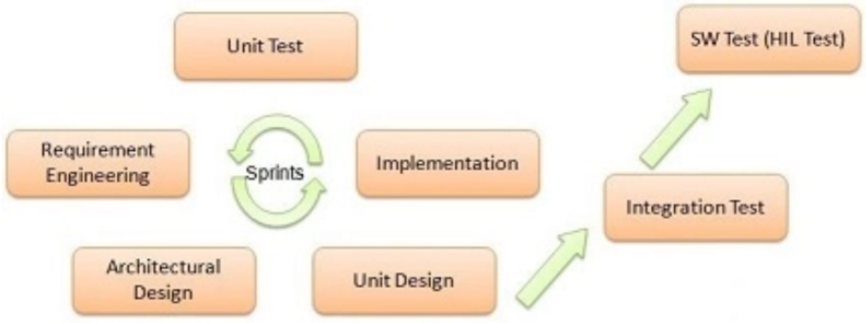
2020.09.11

项目总体

1. 协作流程



   a. 每日站立会议（或频繁）
   b. 总结会议（sprint结束时）
2. 总体技术栈的确定
   a. 前端
   b. 后端
3. 项目规范
   a. 总体规范
      i. git flow控制版本、分支
         - master分支：产品代码。只能从其他分支合并，**不允许**直接修改。

           该分支会触发CI/CD。
         - develop分支：开发分支，用于充当开发的基线。主要从其他分支合并。

           该分支会触发CI/CD。

           如果有需要，可以在经过讨论后，将尚未完成或未稳定的feature分支合并到本分支，并在提交信息中注明WIP(Working In Progress)。如：

           `wip(user-info): 增加用户个人档案`
         - feature分支：用于开发新功能。功能开发完成后，将分支内容合并到develop分支，并删除该分支。

           该分支命名为 `feature/[功能名]` 。
         - release分支：用于发布新版本。版本发布完成后，将分支内容合并到master和develop分支，并删除该分支。

           该分支会触发CI/CD。

           该分支命名为 `release/[版本号]` 。版本号格式规定如下： `x.y.[z]` ，其中x为当前迭代数，y为该迭代已发布的功能数 + 1，z为该版本的修复补丁数，为0时可以不写。例如，迭代一的第一个功能发布后，版本号应为1.1。初始版本号为1.0。对迭代二的第一个发布版本进行修复后，版本号应为2.1.1。
         - hotfix分支：用于修复已发布版本中的bug。版本发布完成后，将分支内容合并到master和develop分支，并删除该分支。

           该分支命名为 `hotfix/[错误名]` 。
      ii. prettier控制代码风格（自动）
      iii. commitlint控制提交信息

每次提交必须有明确的意义，并且在Commit Message中描述清楚。具体细节如下：

- 格式：`type[(scope)]: message`，如 `fix(search): some message`。其中，type为提交类型（具体类型将在下一点中说明），scope为本次修改的范围（功能），message为本次提交的信息。
- 提交类型如下：
  - build：修改项目构建信息
  - ci：调整CI流程
  - chore：调整项目配置
  - docs：修改文档
  - feat：发布功能
  - fix：修复bug
  - perf：优化性能
  - refactor：进行重构
  - revert：回退版本
  - style：调整格式
  - test：增加测试
  - wip：尚未完成的功能
- 提交信息使用小驼峰。

如果提交遇到冲突，必须与当事人沟通解决，**禁止**私自对他人的分支和代码进行修改。

1. 提交前使用rebase减少冗余提交信息

iv. CI自动执行

b. 前端

i. eslint + prettier

ii. commitlint

c. 后端

i. maven + prettier

ii. commitlint

4. 需求分析
5. 领域设计
6. 制定目标
7. 分工
   a. 职责划分
      i. 架构、运维、前端：syl
      ii. 服务端开发：
         1. ycj、yzj、syl
         2. 顾问：tcj
      iii. 数据处理：
         1. zyq、syl
         2. 顾问：zwq
   b. 加入github organization
   c. 统一依赖版本
      i. 虽不必须，但可以减少开发成本
      ii. 不是所有本地依赖都可以容器化，本地的开发环境一致是有必要的
      iii. 建议配置：
         1. maven 3.6.3，Java 8
         2. node 12.18.3（可以通过nvm管理，以备不时之需）

Sprint 1

考虑到网络情况，使用国内的Git管理工具？
但是需要利用第三方的CI/CD工具，所以还是使用github进行管理。

1. 基础设施建设
   a. CI/CD
   b. 项目结构搭建、配置
   c. *mock**

    d. 连通性测试
2. 需求
3. 架构
4. 设计
5. 实现
6. 测试