前端设计文档

> 主要负责人：张鹏，江文宾，廖承军
>
> 审核人：廖承军

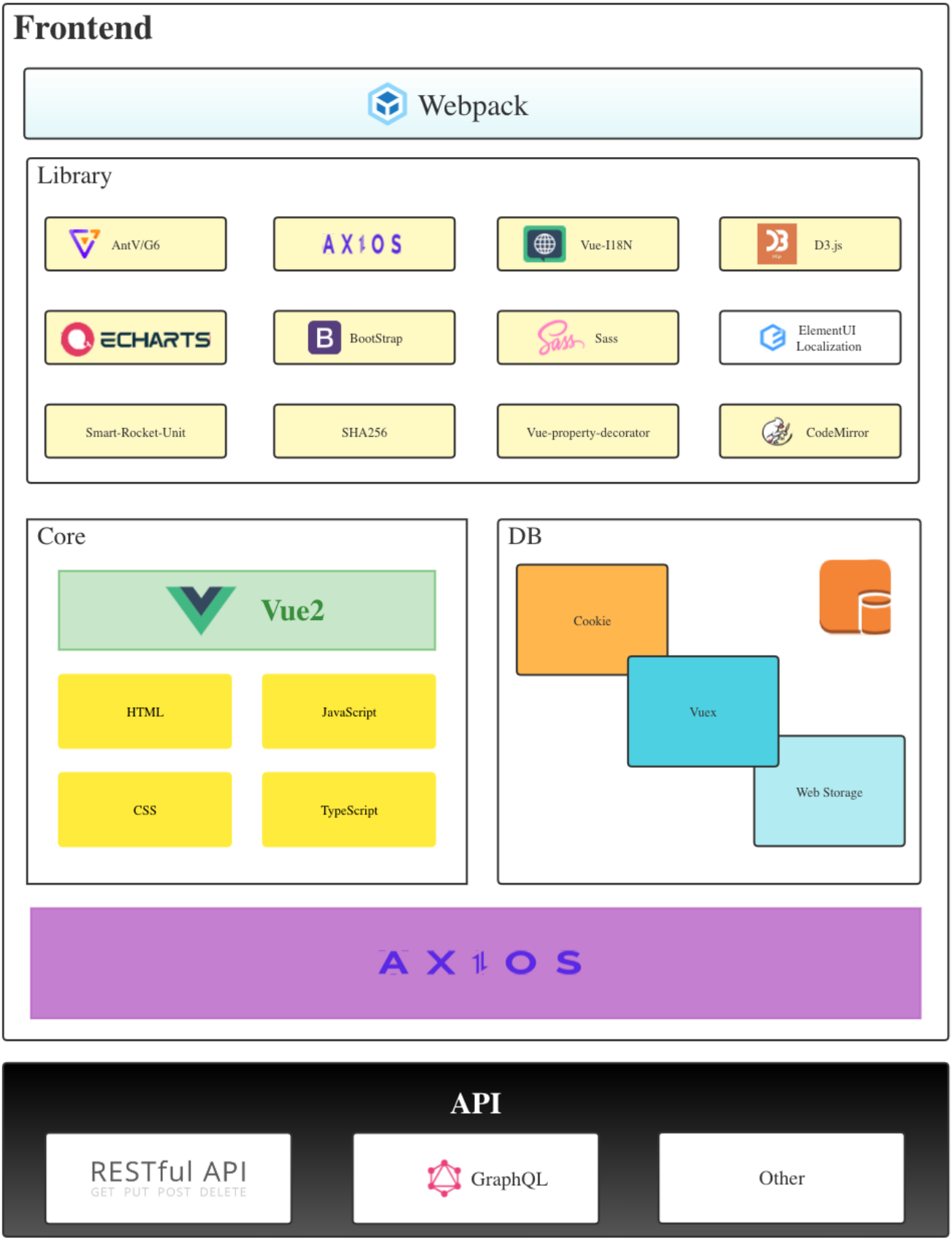
一、 技术选型及理由

本项目的前端选用的技术和使用理由如下：

- **Vue2前端开发框架**
  - **Vue2 框架是当前最流行也最好用的前端框架利器之一**。在github上有很高的热度（201k stars)。该技术有较好的成熟度，开发风险较低，在多个技术场景下都有相对成熟的解决方案，并且社区活跃，遇到难题可以及时求助得到解决。基于此，我们可以快速地进行软件的迭代开发。
  - **Vue2速度较快，性能较优**。和其他框架相比较，Vue采取了一种特立独行的操作DOM的方式——不直接操作DOM，而是引入虚拟DOM的概念，其逻辑安插在 Javascript 和 真实DOM之间，能有效地提高Web性能。
  - **组件模块化开发，可维护性较高**。Vue是基于组件的概念编程。本质是一个单页应用，内部嵌套着多个组件。我们可以单独将某个组件模块化，编写独立的UI组件。当某个组件发生问题时，方便进行隔离和调试。
- **Element UI组件库**
  - Element UI组件库是一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的桌面端组件库。
  - 官方给出Element UI组件的详细教程，降低了学习成本，缩短了界面实现的时间。
- **vue-router 路由框架**
  - vue-router是为Vue框架所定制的一套完整好用的路由解决方案，技术较为成熟，搭配Vue可以帮助我们快速了开发出多页面应用。
  - vue-router官方教程简洁明了，方便快速上手和使用。
- **Nginx**
  - Nginx是一个高性能且轻量级的HTTP和反向代理的服务器，因其稳定性，功能丰富，低系统资源消耗而闻名。基于此，我们使用Nginx来处理浏览器HTTP请求，返回页面文件给用户。

二、 架构设计

项目的架构设计如下图所示，该项目使用经典的B/S架构，使用Nginx进行反向代理，提供网页文件服务，并使用多个dockers运行后台程序，docker与数据库之间进行数据交互，可以很方便提高服务器的抗压能力、性能。



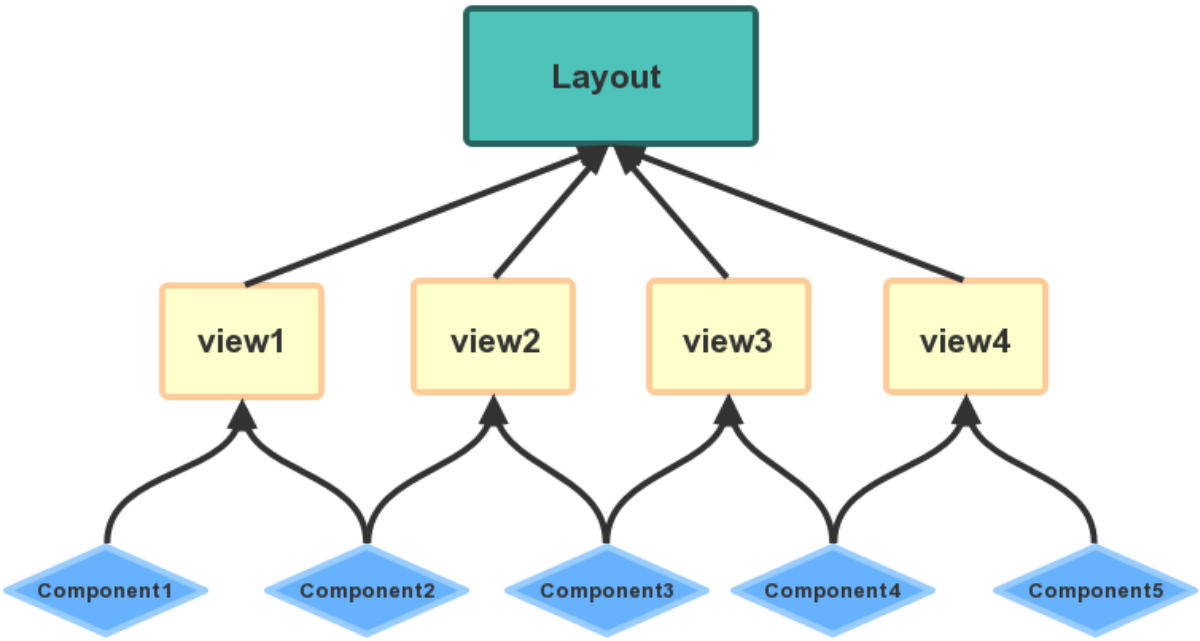
三、 模块划分

本项目的前端模块划分如下：

```
configuration:
 |--- env        // 环境配置
    |--- check.env.js    // 检测中心
    |--- demo.env.js     // demo
    |--- dev.env.js      // 本地开发
    |--- fudev.env.js    // future开发
    |--- futest.env.js   // future测试
    |--- pre.env.js      // 预生产
    |--- prod.env.js     // 生产
    |--- rdev.env.js     // 开发
    |--- sample.env.js   // 配置样本
    |--- test.env.js     // 测试环境
    |--- unittest.env.js // 单元测试
 |--- webpack.config.base.js  // webpack基础配置
 |--- webpack.config.dev.js   // webpack开发配置
 |--- webpack.config.local.js // webpack本地开发配置
 |--- webpack.config.prod.js  // webpack发布配置
 |--- webpack.config.test.js  // webpack单元测试配置
docs:  // 项目文档
 |--- TestGrid_Design.md  // TestGrid前端设计文档
src:
 |--- api              // 调用后端接口
 |--- components       // 全局公共组件
 |--- directives       // 自定义vue指令
 |--- filters          // 自定义页面过滤器
 |--- i18n             // 国际化
 |--- interface        // TS类型定义
 |--- pwa              // pwa配置
 |--- router           // 路由
 |--- store            // 状态仓库
 |--- style            // 公共样式
 |--- theme            // 主题
 |--- utils            // 工具类
 |--- views            // 视图
     |--- components       // 页面组件
     |--- systemManagement // 管理员相关页面(Role)
     |--- testManager      // 测试经理相关页面(Role)
     |--- tester           // 测试人员相关页面(Role)
     |--- *.vue            // 无角色页面
 |--- App.vue          // 项目入口vue文件
 |--- main.ts          // 项目入口JS文件
 |--- permission.ts    // 路由权限
 |--- version.json     // 版本信息
static:  // 素材文件夹
 |--- fonts  // icon字体库
 |--- help   // 帮助文档
 |--- img    // 图片
test:    // 单元测试
 |--- unit:      // 单元测试文件
 |--- setup.js   // 单元测试启动文件
types:           // TS类型
babel.config.js  // babel配置文件
index.html       // 顶层入口页面文件
```

```
jest.config.js    // 单元测试配置文件
Makefile          // make编译配置
package.json      // 项目配置文件
```

这里简单介绍一下组件的划分，组件的架构大致如下图所示：



- views/layout：主界面。将调用下层的views/*.vue组件，组成完整的界面。
- views/*.vue：子界面。依赖于views/components的各个子组件，构成一个个具有交互功能的子界面。
- views/components：子组件。接受views/*.vue的调用，完成各个子组件行为的渲染。

四、软件设计技术

1. 面向对象编程

在ES6语法中，Vue2组件开发常常使用类的形式来组织组件，以面向对象的方式来定义组件的内容和行为；

与此同时，此项目也支持TypeScript类型，那么，ES与TS可同时存在和使用。

在layout，views目录下所有组件以继承vue-property-decorator中的vue

```
view:
 |--- systemManagement  // 系统管理人员相关界面
     |--- TestReportTemplate                          // 测试报告模板
     |--- AddCompileEnv.vue                         // 创建编译环境
     |--- AllTaskMonitorList.vue                      // 所有任务
     |--- CompileEnvList.vue                          // 编译环境列表
     |--- EditCompileEnv.vue                          // 编辑编译环境
     |--- EditEnvLib.vue                              // 编辑环境库
     |--- License.vue                                 // 许可证管理
```

```
      |--- LogView.vue                                  // 日志查看
      |--- UserDefinedFieldsIntegrationTest.vue   // 集成测试函数/用例信息
      |--- UserDefinedFieldsUnitTest.vue          // 测试用例函数/用例信息
      |--- UserEdit.vue                           // 编辑用户
      |--- UserForm.vue                           // 添加用户
      |--- UserList.vue                           // 用户列表
  |--- testManager       // 测试经理相关页面
      |--- TestDashBoard.vue                      // 测试仪表盘
      |--- TestManagerStaticAnalysisOverview.vue  // 静态分析总览
      |--- TestOverviewInfoList.vue               // 测试项目总监
  |--- tester  // 测试人员相关页面
      |--- AddProject                          // 创建项目
      |--- AddVersion                          // 创建版本
      |--- AnalysisHome.vue                 // 静态分析-主页
      |--- DetectionTemplateEdit            // 修改检测模板
      |--- DetectionTemplateOperation          // 静态分析检测模板添加
      |--- IntegrationHome                     // 集成测试-主页
      |--- IntegrationList                     // 集成测试-文件列表
      |--- IntegrationOverviewInfo             // 集成测试-项目总览
      |--- OverviewInfo                     // 单元测试-项目总览
      |--- OverviewInfoList                 // 单元测试-文件列表
      |--- OverviewInfoListForIntegration      // 集成测试-总览列表
      |--- ProjectInvokeGraph                  // 项目调用关系图
      |--- ProjectMain                         // 单元测试-主页
      |--- StaticAnalysisOverview              // 静态分析-总览
      |--- ConfigVersion.vue                // deprecated
      |--- DetectionTemplateList.vue        // 静态分析-检测模板列表
      |--- FunctionSelect.vue                  // 版本功能选择页面
      |--- KnowledgeCenter.vue                 // 知识中心
      |--- McdcDetails.vue                     // MCDC详情
      |--- MyTaskMonitor.vue                // 我的任务
      |--- ProjectList.vue                     // 项目列表
      |--- ReUpload.vue                     // 重新上传源代码
      |--- VersionList.vue                     // 版本列表
  |--- ChangePassword.vue         // 修改密码
  |--- FAQ.vue                    // 常见问题
  |--- InvalidLicense.vue      // 无效许可证
  |--- Login.vue               // 登录
  |--- Manual.vue                 // 手册
  |--- NotFound.vue             // 404
  |--- Redirect.vue             // 重定向
  |--- TestComponents.vue       // 组件测试
  |--- VersionInfo.vue            // 版本信息

layout:
  |--- index.vue             // 主界面跳转
```

五、TestGrid版本

1. v2.5.0
2. v2.5.1