

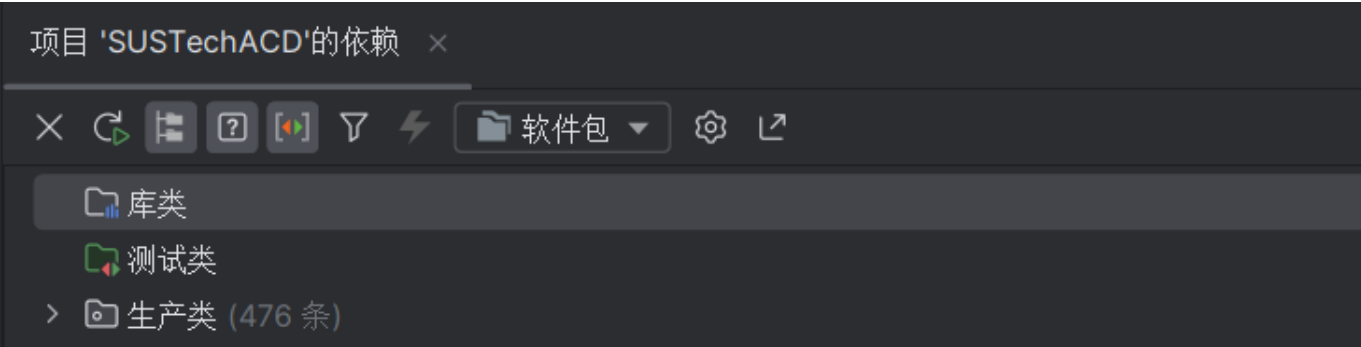
# CS304 SUSTech\_ACD 经费管理系统 Final Report

## 1. Metrics

Use the IDEA plug-in Static to count the number of lines of various types of code, the number and size of files as follows:

Statistic										
Statistic										
Refresh Refresh on selection Settings										
Overview <> css <> html <> js <> py <> vue										
Extension	Count ^	Size SUM	Size MIN	Size MAX	Size AVG	Lines	Lines MIN	Lines MAX	Lines AVG	Lines CODE
development (DEVELOPMENT)	1x	0kB	0kB	0kB	0kB	5	5	5	5	2
editorconfig (EDITORCONFIG)	1x	0kB	0kB	0kB	0kB	14	14	14	14	11
ico (ICO files)	1x	67kB	67kB	67kB	67kB	17	17	17	17	17
production (PRODUCTION files)	1x	0kB	0kB	0kB	0kB	6	6	6	6	2
py (PY files)	1x	22kB	22kB	22kB	22kB	576	576	576	576	451
staging (STAGING files)	1x	0kB	0kB	0kB	0kB	8	8	8	8	3
ttf (TTF files)	1x	11kB	11kB	11kB	11kB	91	91	91	91	73
woff (WOFF files)	1x	6kB	6kB	6kB	6kB	41	41	41	41	40
xlsx (XLSX files)	1x	22kB	22kB	22kB	22kB	152	152	152	152	151
css (CSS files)	2x	424kB	0kB	423kB	212kB	27	1	26	13	26
json (JSON files)	2x	0kB	0kB	0kB	0kB	12	3	9	6	12
pdf (PDF files)	2x	796kB	242kB	553kB	398kB	8466	2976	5490	4233	8397
yml (YML files)	2x	0kB	0kB	0kB	0kB	27	5	22	13	10
hbs (HBS files)	3x	0kB	0kB	0kB	0kB	68	16	26	22	59
csv (CSV files)	4x	3kB	0kB	2kB	0kB	43	1	26	10	43
html (HTML files)	4x	83kB	0kB	54kB	20kB	1213	15	621	303	1014
md (MD files)	4x	22kB	0kB	9kB	5kB	742	15	359	185	437
scss (SCSS files)	18x	93kB	0kB	8kB	5kB	3980	31	320	221	3699
svg (SVG files)	46x	45kB	0kB	3kB	0kB	46	1	1	1	46
js (JS files)	104x	174kB	0kB	12kB	1kB	5999	0	530	57	4714
vue (VUE files)	258x	737kB	0kB	45kB	2kB	28920	0	1779	112	25249
Total:	458x	2,513kB	375kB	1,243kB	779kB	50453	3974	10119	6086	

A total of 476 software packages were obtained using the IDEA code analysis function.



A total of 6 sub-modules: backend, frontend, build, mock, test, drawing;

The backend is divided into three sub-modules: data (csv), server, and test;

The front end is divided into api, assets, components, directive, filters, icons, layout, router, store, styles, utils, vendor, view multiple sub-modules.

The third-party libraries used by the backend are:

```
3 import numpy as np
4 from flask import Flask, request, jsonify, json, send_from_directory
5 from flask_cors import CORS, cross_origin
6 import pandas as pd
7 import math
8 import os
```

The third-party libraries used by the front end are:

```
"dependencies": { "axios": "0.18.1", "clipboard": "2.0.4", "codemirror": "5.45.0", "core-js": "3.6.5", "driver.js":  
"0.9.5", "dropzone": "5.5.1", "echarts": "4.2.1", "echarts-stat": "^1.2.0", "echarts-wordcloud": "^1.1.3", "element-  
ui": "2.13.2", "file-saver": "2.0.1", "fuse.js": "3.4.4", "js-cookie": "2.2.0", "jsonlint": "1.6.3", "jszip": "3.2.1",  
"normalize.css": "7.0.0", "nprogress": "0.2.0", "path-to-regexp": "2.4.0", "screenfull": "4.2.0", "script-loader":  
"0.7.2", "sortablejs": "1.8.4", "tui-editor": "1.3.3", "vue": "2.6.10", "vue-count-to": "1.0.13", "vue-router": "3.0.2",  
"vue-splitpane": "1.0.4", "vuedraggable": "2.20.0", "vuex": "3.1.0", "xlsx": "0.14.1" }, "devDependencies": {  
"@vue/cli-plugin-babel": "4.4.4", "@vue/cli-plugin-eslint": "4.4.4", "@vue/cli-plugin-unit-jest": "4.4.4",  
"@vue/cli-service": "4.4.4", "@vue/test-utils": "1.0.0-beta.29", "autoprefixer": "9.5.1", "babel-eslint": "10.1.0",  
"babel-jest": "23.6.0", "babel-plugin-dynamic-import-node": "2.3.3", "chalk": "2.4.2", "chokidar": "2.1.5",  
"connect": "3.6.6", "eslint-plugin-vue": "", "html-webpack-plugin": "3.2.0", "husky": "1.3.1", "lint-staged": "",  
"mockjs": "1.0.1-beta3", "plop": "2.3.0", "runjs": "4.3.2", "sass": "1.26.2", "sass-loader": "8.0.2", "script-ext-html-  
webpack-plugin": "2.1.3", "serve-static": "1.13.2", "svg-sprite-loader": "4.1.3", "svgo": "1.2.0", "vue-template-  
compiler": "2.6.10" }
```

There are 60 in total.

## Maintainability

The tool used is: SonarLint The open-source project for comparison is: vue-element-admin

Analysis results of this project: 213 issues

SonarLintCurrent FileReportSecurity HotspotsTaint VulnerabilitiesLog

Found 213 issues in 61 files

> 401.vue (1 issue)

> Activity.vue (4 issues)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> BoxCard.vue (1 issue)

> Logo.vue (2 issues)

> Navbar.vue (1 issue)

> PanelGroup.vue (12 issues)

> PanelGroup.vue (12 issues)

> Project API Document.html (15 issues)

> SingleImage.vue (1 issue)

> SingleImage2.vue (1 issue)

> SingleImage3.vue (2 issues)

> TagsMostUpvotes.vue (5 issues)

> TagsMostUpvotes\_admin.vue (5 issues)

> Todo.vue (1 issue)

> Todo.vue (1 issue)

> Todo.vue (1 issue)

Analysis of 456 files done 3 minutes ago

3 / 15

The analysis result of the comparison project is: 48 issues

```

  ✓ Found 48 issues in 31 files
    > ✓ 401.vue (1 issue)
    > ✓ Activity.vue (4 issues)
    > ✓ BoxCard.vue (1 issue)
    > ✓ Logo.vue (2 issues)
    > ✓ Navbar.vue (1 issue)
    > ✓ SingleImage.vue (1 issue)
    > ✓ SingleImage2.vue (1 issue)
    > ✓ SingleImage3.vue (2 issues)
    > ✓ Todo.vue (1 issue)
    > ✓ autocomplete-suggestions.vue (1 issue)
    > ✓ autocomplete.vue (1 issue)
    > ✓ back-to-top.vue (2 issues)
    > ✓ cascader.vue (2 issues)
    > ✓ count-to.vue (2 issues)
    > ✓ date-table.vue (3 issues)
    > ✓ dropdown-item.vue (1 issue)
    > ✓ image-viewer.vue (1 issue)
    > <> index.html (1 issue)
    > ✓ index.vue (3 issues)
    > ✓ index.vue (2 issues)
    > ✓ index.vue (1 issue)
    > ✓ index.vue (1 issue)

```

Analysis of 41376 files done few seconds ago

As you can see, the

maintainability of our project is very good.

## 2. Documentation

Documentation分为用户端文档和开发者文档两个部分。

User-end document

**SUSTech ACD 经费管理系统 - 用户端使用文档**

URL:

<https://github.com/skylynf/SUSTechACD/blob/main/%E7%94%A8%E6%88%B7%E7%AB%AF%E6%96%87%E6%A1%A3.pdf>

上述文档是一个用户端使用文档的示例，它旨在帮助用户理解和操作系统的各个功能模块。该文档提供了以下内容：

- 1. **简介**：介绍了系统的基本概念，如课题组、经费和支出之间的关系，以及一些限制条件。
- 2. **操作列表**：列出了用户可以执行的各种操作，包括查询、审批、增加/删除、修改等。
- 3. **用户使用内容**：描述了用户在系统中的角色和相关操作，包括登录、注册、导航栏功能和各个模块的具体功能。
- 4. **导出功能**：说明了用户可以使用导出功能将生成的Excel表格导出的方法。

整个文档结构清晰，按照不同功能模块分别进行介绍，并提供了具体的操作步骤和注意事项。文档旨在帮助用户快速上手系统，了解每个功能的用途和操作方法，以提高用户的使用体验。

目录

- 1. 登录和注册
- 2. 导航栏功能说明
- 3. 展示模块
- 4. 支出模块
- 5. 经费模块
- 6. 审批模块
- 7. 导出功能

Screenshot

查询待审批支出

- 1. 在审批模块中选择查询功能。
- 2. 显示待审批的所有支出列表，包含支出的具体情况。

审批支出

- 1. 在审批模块中选择待审批的支出。
- 2. 进行相应的审批操作，包括通过或拒绝。
- 3. 点击确认按钮完成支出的审批操作。

expenseID	expenseName	fundID	amount	operator	category1	category2	abstract	remark	applicationState	
1	培训	1	100	王老博	会议费	培训费	培训费用	培训费用	1	审批
20001	20001测试	1231	10	梅老博	办公费	其他交通费	测试20001测试	测试20001备注	1	审批

7. 导出功能

用户可以使用导出功能将生成的Excel表格导出。

- 1. 在相应功能模块中点击导出按钮。
- 2. 选择导出的文件格式，如Excel。
- 3. 系统将生成对应的文件并提示下载。

文件开始插入页面布局公式数据审阅视图帮助特色功能页面设置描述性指南

格式刷格式

8. 注意事项

每一个支出都必须对应存在的经费号，而且该经费号一定是属于本用户（课题组）的。

经费号、支出号、用户ID都是独一无二的，不允许重复。

查询内容时，不在输入框输入任何内容，默认为查询该用户权限下能查询的所有内容。

Developer-end document

URL: <https://github.com/skyllynf/SUSTechACD/blob/main/Project%20API%20Document.pdf>

这份文档是SUSTech\_ACD API开发者文档，用于介绍SUSTech\_ACD财务管理系统的API接口及其使用方法。该系统用于管理课题组的经费和支出情况。

以下是文档的目录：

- 1. 简介
- 2. 操作列表
  - 总览
    - 查询某课题组名下的全部经费的完成情况（支出情况）
    - 查询一笔支出的申请情况
    - 可视化查看一笔经费的使用情况和支出项目
    - 查询若干个fundID的使用情况和执行率
  - 审批
    - 查询待审批的所有支出列表

- 审批支出
  - 增加/删除
    - 增加或删除一个经费
    - 增加或删除一个支出
  - 修改
    - 修改一个支出并重新送审
    - 修改一个经费
3. Json对象内容示例
- 经费 (fund)
  - 支出 (expense)
  - 课题组 (user)
4. 后端API
- 获取课题组全部经费完成情况（支出情况）
  - 查询若干个fundID的使用情况和执行率，或是所有fundID的使用情况和执行率
  - 查询一笔支出的申请情况
  - 查询待审批的所有支出列表
  - 增加或删除一个经费
  - 增加或删除一个支出
  - 送审一个支出
  - 修改一个支出并重新送审
  - 修改一个经费
  - 用户操作
    - 增加用户
    - 修改用户
    - 删除用户
    - 用户登录
5. 注意事项

该文档提供了系统中各种操作的API接口说明，包括请求方式、路径、参数和响应等信息。开发者可以根据文档中的接口定义和示例，进行后端API的实现和功能完善。此外，文档还提供了一些注意事项，如占位符替换和后端实现建议等。

## ScreenShot

修改

- 1. 修改一个支出并重新送审
  - 请求方式: PUT
  - 路径: /api/expenses/{expenseID}
  - 参数: expenseID (待修改的支出ID) , 支出相关信息 (expenseName、amount等)
  - 响应: 返回修改成功的支出信息
- 2. 修改一个经费
  - 请求方式: PUT
  - 路径: /api/funds/{fundID}
  - 参数: fundID (待修改的经费ID) , 经费相关信息 (fundName、totalQuota等)
  - 响应: 返回修改成功的经费信息

Json对象内容 示例

经费 (fund)

- 一个唯一的经费号 fundID
- 经费名称 fundName
- 一个课题组 userID
- 经费总额度 totalQuota
- 已使用额度 usedQuota
- 经费内容摘要 abstract
- 经费备注 remark

```
code{
  "fundID": 5433912,
  "fundName": "国自然",
  "userID": 12,
  "totalQuota": 500,
  "usedQuota": 10,
  "abstract": "this is a abstract",
  "remark": "no remark"
}
```

支出 (expense)

- 唯一识别: 支出编号 expenseID
- 支出名称 expenseName
- 该支出对应的经费号 fundID

3. Tests

3.1 Front-end Test

**3.1.1** The front-end testing is completed using Jest, an open-source JavaScript testing framework from Facebook that automatically integrates all the testing tools required by developers such as assertions, JSDom, and coverage reports. It is an almost zero configuration testing framework.

**3.1.2** Test code example:



```

new *
26 ▶ describe( name: 'url', fn: () => {
27 ▶   test( name: 'should return the correct result', fn: () => {
28     const result =validURL( url: 'ww.56789')[1];
29     expect(result).toBe( expected: false);
30   });
31 });
32
new *
33 ▶ describe( name: 'validLowerCase', fn: () => {
34 ▶   test( name: 'should return the correct result', fn: () => {
35     const result =validLowerCase( str: 'a');
36     expect(result).toBe( expected: true);
37   });
38 });
39
new *
39 ▶ describe( name: 'validUpperCase', fn: () => {
40 ▶   test( name: 'should return the correct result', fn: () => {
41     const result =validUpperCase( str: 'A');
42     expect(result).toBe( expected: true);
43   });
44 });
45
new *
45 ▶ describe( name: 'validAlphabets', fn: () => {
46 ▶   test( name: 'should return the correct result', fn: () => {
47     const result =validAlphabets( str: 'AsdsQW');
48     expect(result).toBe( expected: true);
49   });
50 });
51
new *

```

### 3.1.3 Test Coverage:

```

===== Coverage summary =====
Statements   : 93.22% ( 165/177 )
Branches     : 81.72% ( 76/93 )
Functions    : 89.47% ( 17/19 )
Lines        : 93.06% ( 134/144 )
=====

```

## 3.2 Back-end Test

### 3.2.1 后端测试使用python的unittest, 一个典型的测试方法实现如下

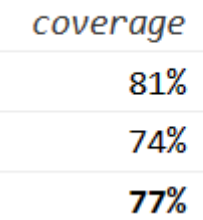
```

def test_get_expense_by_id(self):
    # 发送 GET 请求并获取响应
    response = self.app.get('/api/expenses/1')
    # 验证状态码是否为 200
    assert response.status_code == 200
    # 获取响应数据
    data = response.get_json()
    # 验证数据内容
    assert 'error' not in data # 没有错误信息表示成功

```

```
assert isinstance(data, dict) # 数据类型为字典
assert 'expenseID' in data # 包含 expenseID 字段
```

3.2.2 测试覆盖率 use python's coverage to Using the Python library coverage to gather relevant information.



Due to the complexity of certain functions in server.py, such as generating charts or graphical representations in the frontend, achieving a high test coverage rate has been challenging. However, it should be noted that the majority of the CRUD (Create, Read, Update, Delete) operations have been covered by tests. These tests have proven to be immensely valuable during the development of our applications. While the coverage may not be exhaustive due to the specific nature of certain functionalities, the implemented tests have provided significant support and confidence in the overall robustness of the system

4. Build

Front-end Build (Vue.js):

1. Technology/Tools/Frameworks:

- Node.js: JavaScript runtime used to execute build scripts and manage dependencies.
- npm (Node Package Manager): Used to install and manage project dependencies.
- Vue CLI (Command-Line Interface): A development tool for scaffolding and managing Vue.js projects.

2. Tasks executed in the build and final artifacts produced:

- Installation of project dependencies: This includes Vue.js itself and any additional libraries or packages required by your project.
- Compilation and bundling of front-end assets: The build process compiles Vue.js components, CSS styles, and other static assets into optimized and minified files.
- Generation of production-ready artifacts: The final output typically includes HTML, CSS, and JavaScript files that can be served by a web server.

```
npm install
npm run dev
```

3. Buildfile or related artifacts/scripts:

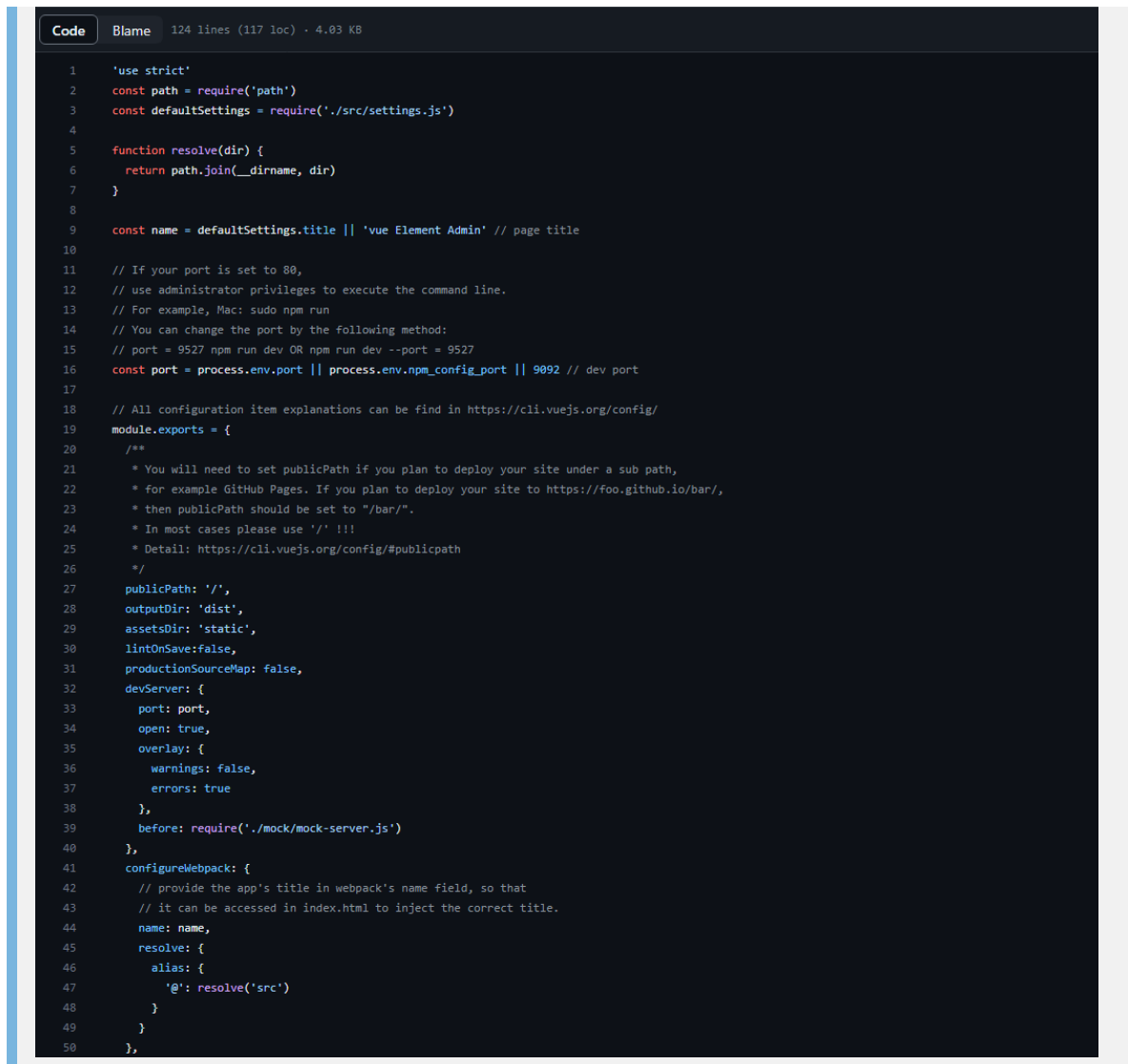
- Vue CLI generates a configuration file called `vue.config.js` in project's root directory. This file can be customized to modify the build process, configure webpack, and handle other build-related tasks.

- Find more information about `vue.config.js` in the Vue CLI documentation: [Vue CLI Configuration Reference](#).
- URL of `vue.config.js` :<https://github.com/skylynf/SUSTechACD/blob/main/vue.config.js>
- Part of Snapshot:



```
Code Blame 124 lines (117 loc) · 4.03 KB
1  'use strict'
2  const path = require('path')
3  const defaultSettings = require('./src/settings.js')
4
5  function resolve(dir) {
6    return path.join(__dirname, dir)
7  }
8
9  const name = defaultSettings.title || 'vue Element Admin' // page title
10
11 // If your port is set to 80,
12 // use administrator privileges to execute the command line.
13 // For example, Mac: sudo npm run
14 // You can change the port by the following method:
15 // port = 9527 npm run dev OR npm run dev --port = 9527
16 const port = process.env.port || process.env.npm_config_port || 9092 // dev port
17
18 // All configuration item explanations can be find in https://cli.vuejs.org/config/
19 module.exports = {
20   /**
21    * You will need to set publicPath if you plan to deploy your site under a sub path,
22    * for example GitHub Pages. If you plan to deploy your site to https://foo.github.io/bar/,
23    * then publicPath should be set to "/bar/".
24    * In most cases please use '/' !!!
25    * Detail: https://cli.vuejs.org/config/#publicpath
26    */
27   publicPath: '/',
28   outputDir: 'dist',
29   assetsDir: 'static',
30   lintOnSave: false,
31   productionSourceMap: false,
32   devServer: {
33     port: port,
34     open: true,
35     overlay: {
36       warnings: false,
37       errors: true
38     },
39     before: require('./mock/mock-server.js')
40   },
41   configureWebpack: {
42     // provide the app's title in webpack's name field, so that
43     // it can be accessed in index.html to inject the correct title.
44     name: name,
45     resolve: {
46       alias: {
47         '@': resolve('src')
48       }
49     }
50   },
```

- The `package.json` file is a manifest file that contains metadata about your project and its dependencies. It is also used to define various scripts that can be executed using npm.
- URL of `package.json` : <https://github.com/skylynf/SUSTechACD/blob/main/vue.config.js>
- Part of Snapshot:



```

Code Blame 124 lines (117 loc) · 4.03 KB
1  'use strict'
2  const path = require('path')
3  const defaultSettings = require('./src/settings.js')
4
5  function resolve(dir) {
6    return path.join(__dirname, dir)
7  }
8
9  const name = defaultSettings.title || 'vue Element Admin' // page title
10
11 // If your port is set to 80,
12 // use administrator privileges to execute the command line.
13 // For example, Mac: sudo npm run
14 // You can change the port by the following method:
15 // port = 9527 npm run dev OR npm run dev --port = 9527
16 const port = process.env.port || process.env.npm_config_port || 9092 // dev port
17
18 // All configuration item explanations can be find in https://cli.vuejs.org/config/
19 module.exports = {
20   /**
21    * You will need to set publicPath if you plan to deploy your site under a sub path,
22    * for example Github Pages. If you plan to deploy your site to https://foo.github.io/bar/,
23    * then publicPath should be set to "/bar/".
24    * In most cases please use '/' !!!
25    * Detail: https://cli.vuejs.org/config/#publicpath
26    */
27   publicPath: '/',
28   outputDir: 'dist',
29   assetsDir: 'static',
30   lintOnSave: false,
31   productionSourceMap: false,
32   devServer: {
33     port: port,
34     open: true,
35     overlay: {
36       warnings: false,
37       errors: true
38     },
39     before: require('./mock/mock-server.js')
40   },
41   configureWebpack: {
42     // provide the app's title in webpack's name field, so that
43     // it can be accessed in index.html to inject the correct title.
44     name: name,
45     resolve: {
46       alias: {
47         '@': resolve('src')
48       }
49     }
50   },

```

## Back-end Build (Python Flask):

### 1. Technology/Tools/Frameworks:

- Python: Programming language used for back-end development.
- pip: Package installer for Python used to manage project dependencies.
- Flask: Micro web framework for Python used to build the back-end server.

### 2. Tasks executed in the build and final artifacts produced:

- Dependency installation: Installation of required Python packages, including Flask and any additional libraries.
- Configuration and setup: The build process might involve setting up environment variables, configuring the database connection, and other initialization steps.

- ```
pip install  
python -m flask run
```

### 3. Buildfile or related artifacts/scripts:

- In Python Flask projects, build scripts are typically not as standardized as in front-end frameworks.

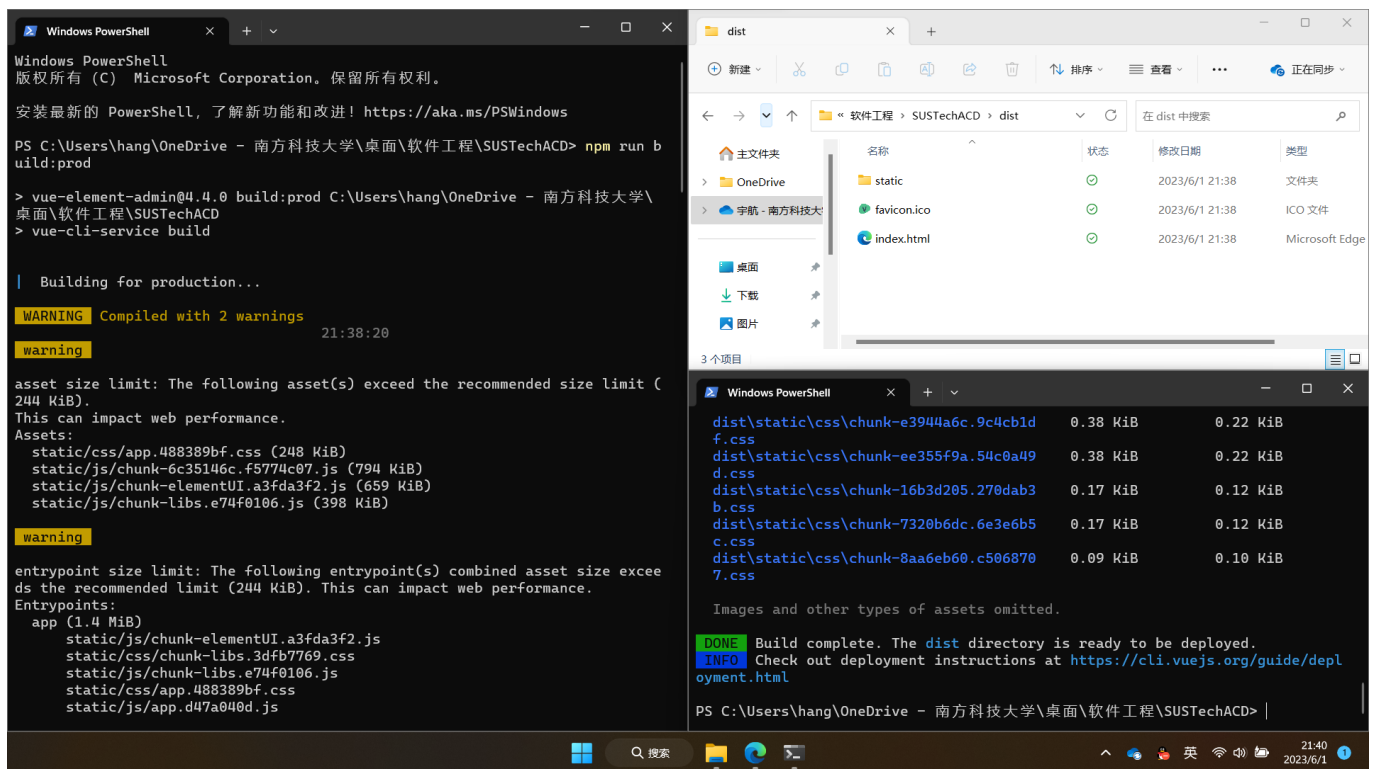
## 5. Deployment

1. Introduce the containerization technology/tools/frameworks used in your project.

In our project, the containerization tools are Nginx and WSGI. This solution is small in size, easy to install, fast in response, easy to use, no additional modification to the code, and completely realize the front end and back end separation.

2. The script or related artifacts used for containerization. Briefly introduce how you assigned different services into different containers.

For the frontend, when you run **npm run dev**, the vue code will run directly, while when you run **npm run build:prod**, the vue code will be transcribed into html/css/js and stored in the dist folder in the root directory, which is done. This way, the packaged code is decoupled from node.js and can be copied to run on any machine without having to install vue and node.js.



Here, another computer is simulated as a WSL virtual machine. The frontends are deployed by first installing the Nginx library and then starting the Nginx server by binding a network port (e.g., localhost:9000) to the dist folder path.

```

dyh@LAPTOP-REDMIG: ~$ sudo apt-get install -y nginx
[sudo] password for dyh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  nginx
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/3620 B of archives.
After this operation, 46.1 kB of additional disk space will be used.
Selecting previously unselected package nginx.
(Reading database ... 72062 files and directories currently installed.)
Preparing to unpack .../nginx_1.18.0-0ubuntu1.4_all.deb ...
Unpacking nginx (1.18.0-0ubuntu1.4) ...
Setting up nginx (1.18.0-0ubuntu1.4) ...
dyh@LAPTOP-REDMIG:~$ gedit /etc/nginx/nginx.conf

```

```

1 user www-data;
2 worker_processes auto;
3 pid /run/nginx.pid;
4 include /etc/nginx/modules-enabled/*.conf;
5
6 events {
7     worker_connections 768;
8     # multi_accept on;
9 }
10
11 http {
12     #
13     # Basic Settings
14     #
15     sendfile on;
16     tcp_nopush on;
17     tcp_nodelay on;
18     keepalive_timeout 65;
19     types_hash_max_size 2048;
20     # server_tokens off;
21
22     # server_names_hash_bucket_size 64;
23     # server_name_in_redirect off;
24
25     include /etc/nginx/mime.types;
26     default_type application/octet-stream;
27
28     #
29     # SSL Settings
30     #
31     ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping
32     #
33     #
34     #
35     #
36     #
37     #
38     #

```

For the backend, our programs are written on python's Flask server. However, Flask is intended for development only, so it should be deployed with a WSGI interface. As a result, the server can run automatically and sleep processes when there are no user requests to reduce power consumption.

```

dyh@LAPTOP-REDMIG:~/bac$ sudo nginx
[sudo] password for dyh:
dyh@LAPTOP-REDMIG:~/backend$ cd backend
dyh@LAPTOP-REDMIG:~/backend$ uwsgi --http :5000 --wsgi-file server.py --callable app
*** Starting uWSGI 2.0.21 (64bit) on [Thu Jun 1 21:35:52 2023] ***
compiled with version: 9.4.0 on 01 June 2023 08:06:00
os: Linux-5.15.90.1-microsoft-standard-WSL2 #1 SMP Fri Jan 27 02:56:13 UTC 2023
nodename: LAPTOP-REDMIG
machine: x86_64
clock source: unix
pcre jit disabled
detected number of CPU cores: 16
current working directory: /home/dyh/backend
detected binary path: /home/dyh/.local/bin/uwsgi
*** WARNING: you are running uWSGI without its master process manager ***
your processes number limit is 63631
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uWSGI http bound on :5000 fd 4
spawned uWSGI http 1 (pid: 1401)
uwsgi socket 0 bound to TCP address 127.0.0.1:45401 (port auto-assigned) fd 3
Python version: 3.8.10 (default, Mar 13 2023, 10:26:41) [GCC 9.4.0]

```

The reason why we adopt the separation of front and back ends is mainly because the requirements of front and back ends are different. Our front-end packaging and deployment has addressed the issues of time-consuming, error-prone, copy-controlled vue and nodejs; Backend applications need to access the database

and can't run without data, so packaging a copy of the database is a very insecure behavior and should not be packaged with the frontend.

3. A proof of successful containerization.

The running results of the program after packaging and deployment are as follows. The UI and various functions are completely the same as without packaging and deployment.

