团队项目报告

1. Metric

代码行数

- **总代码行数**：15204（使用cloc工具计算）

FastAPI Backend:

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Python                           1            137             23            654
Dockerfile                       1              5              6              7
JSON                             1              0              0              6
-------------------------------------------------------------------------------
SUM:                             3            142             29            667
-------------------------------------------------------------------------------
```

Vue 3 Frontend:

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JSON                             3              0              0          12016
Vuejs Component                 20            213            118           2285
JavaScript                       5             14              3            192
Markdown                         1              5              0             19
HTML                             1              0              1             16
Dockerfile                       1              8              7              9
-------------------------------------------------------------------------------
SUM:                            31            240            129          14537
-------------------------------------------------------------------------------
```

包/模块数量

- **总包/模块数量**：20 / 25

FastAPI Backend:

```
Number of packages/modules:
12
1
```

Vue 3 Frontend:

```
Number of packages/modules:
8
24
```

源文件数量

- **总源文件数量**：25

```
Number of source files:
25
```

依赖数量

- **总依赖数量**：14
  - fastAPI backend:
    - fastapi == 0.81.0
    - uvicorn == 0.18.3
    - uvloop == 0.17.0
    - sqlalchemy == 1.4.40
    - pydantic == 1.9.1
    - python-jose == 3.3.0
    - databases == 0.6.0
    - aiohttp == 3.8.1
    - python-multipart == 0.0.5
    - ...
  - vue3 frontend:
    - axios == 1.6.8
    - bootstrap == 5.3.3
    - core-js == 3.8.3
    - vue == 3.2.13
    - vue-router == 4.3.2
    - vuex == 4.1.0
    - ...

FastAPI Backend:

```
Number of dependencies:
8
```

Vue 3 Frontend:

```
Number of dependencies:
6
```

2. 文档

用户文档

- 我们提供了一份详细的 **README.md** 文件，包含软件的安装、启动和使用步骤。
- 用户可以通过我们的GitHub页面访问：GitHub链接

开发者文档

- 我们的API文档详细描述了所有公开接口的使用方法和参数，便于开发者理解和扩展代码。
- 开发者文档也可通过GitHub页面查看：API文档链接

3. 测试

测试技术/工具

- 使用 **pytest** 进行自动化测试
- 使用 **postman** 提升测试有效性

测试代码

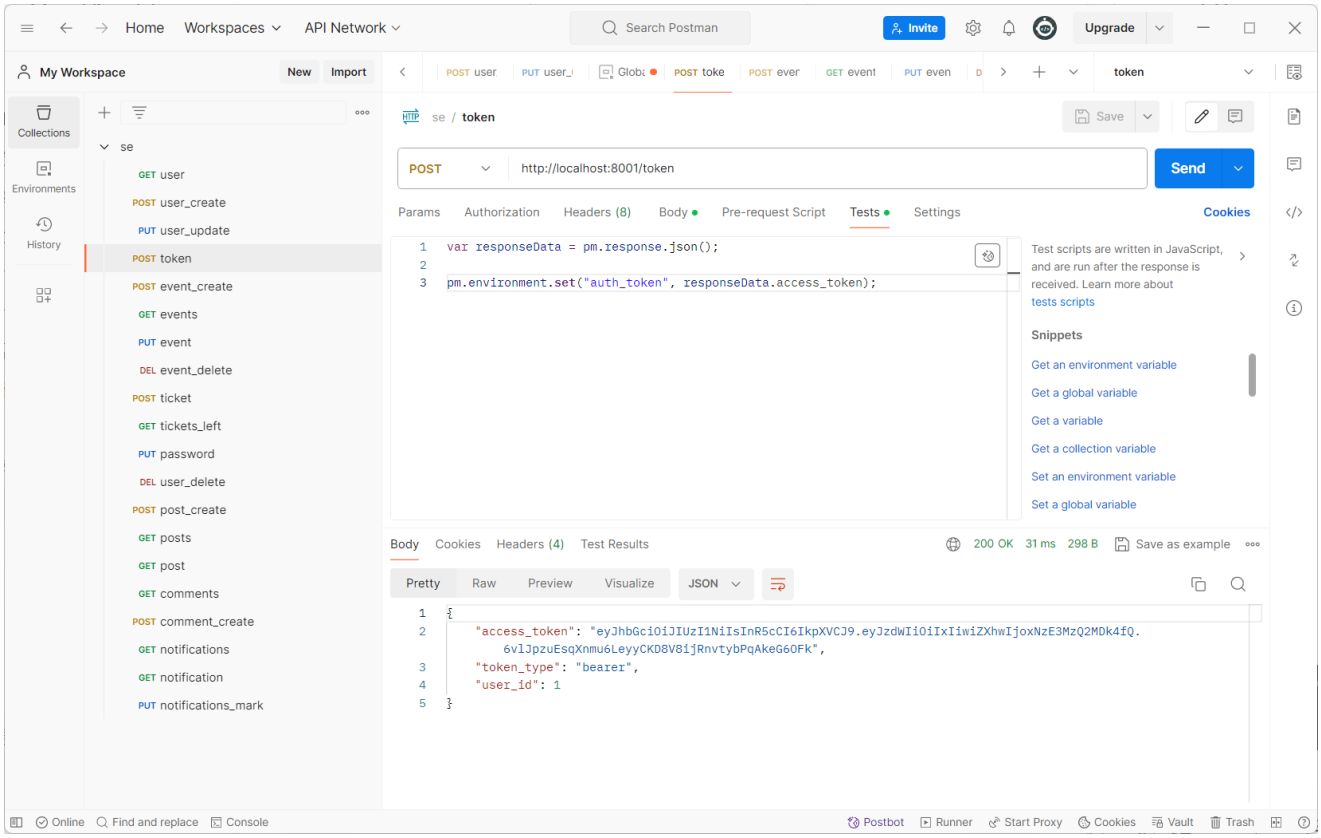
pytest测试:

- 测试代码包含在 `test_main.py`
- GitHub链接：[测试代码]

```
test_main.py::test_read_notification
  C:\Users\lerrorgk\anaconda3\Lib\site-packages\pydantic\main.py:1211: PydanticDeprecatedSince20: The `from_orm` method is deprecated; set `model_config['from_attributes']=True` and use `m
odel_validate` instead. Deprecated in Pydantic V2.0 to be removed in V3.0. See Pydantic V2 Migration Guide at https://errors.pydantic.dev/2.7/migration/
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
========================================================================== 20 passed, 8 warnings in 4.13s ==========================================================================
```

postman测试:



测试效果

- 当前的测试覆盖率为 90%，能够有效捕捉大部分关键功能和http接口的异常。

4. 构建

构建技术/工具

- 使用 **vue3**构建前端，**Docker** 容器化整个应用。

- 使用 Jenkins 自动化构建和测试流程。

构建产物

- 成功构建后产生的主要产物是一个Docker镜像。

构建文件

- 构建配置文件为 **package.json**, **Dockerfile**，可在我们的GitHub项目中找到：Dockerfile链接

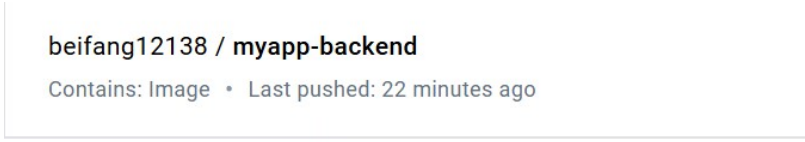
5. 部署

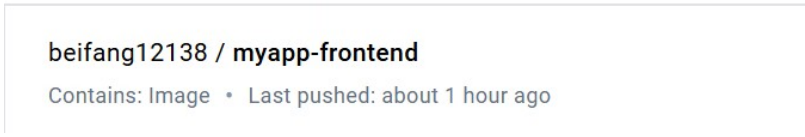
部署技术/工具

- 使用 **Docker** 进行现代化的容器化部署。
- 前端部署的dockerfile链接：Dockerfile
- 后端部署的dockerfile链接：Dockerfile
- 容器化成功的证明：

```
#13 exporting to image
#13 exporting layers done
#13 writing image sha256:1f693ad57c0f8b799c0487c0df352fc9c9be5344fd97bd2b07703a591afa1079 done
#13 naming to docker.io/library/myapp-frontend:latest done
#13 DONE 0.0s
[Pipeline] sh
+ docker build -t myapp-backend:latest -f backend/Dockerfile ./backend
#0 building with "default" instance using docker driver
```

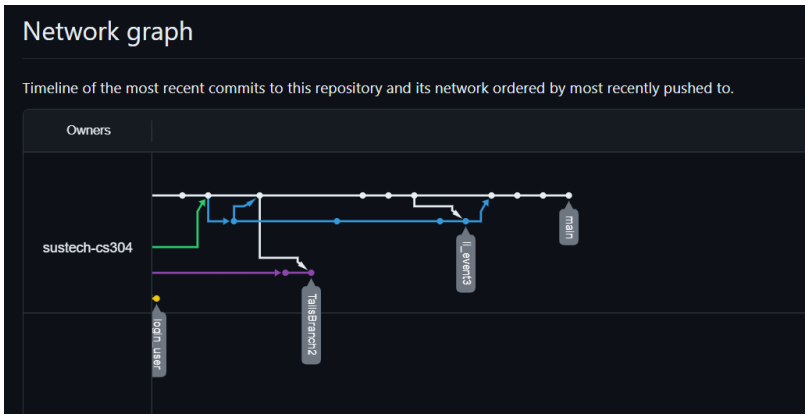
```
#11 exporting to image
#11 exporting layers 0.0s done
#11 writing image sha256:b9b591e9966eae7b610d7a18c093edb92d59aa205ae0f4964c54ad9f1965c86e done
#11 naming to docker.io/library/myapp-backend:latest done
#11 DONE 0.0s
```





团队协作与AI使用

- 项目使用 **git** 进行版本控制和团队协作



项目中AI的使用情况

在我们的项目中，我们大量借鉴了ChatGPT-4来简化开发过程并提高代码质量。AI的主要使用包括生成代码片段、提供架构建议以及解决复杂的编程问题。以下是我们具体使用AI的情况：

1. **代码生成**：
   - **函数和方法**：项目中许多函数和方法最初是由ChatGPT-4生成的，这些代码为我们提供了一个良好的基础，然后我们根据具体需求进行细化和修改。
   - **算法实现**：对于复杂或需要优化的算法，ChatGPT-4提供了初始实现，我们的团队随后对其进行了审查和优化。
2. **架构指导**：
   - **设计模式**：ChatGPT-4建议了适用于项目各个组件的设计模式，这些建议帮助我们维护了一个干净且可扩展的架构。
   - **系统设计**：总体系统设计和组件之间的通信策略也受到了ChatGPT-4建议的影响。
3. **问题解决**：
   - **调试**：当遇到代码中的错误或意外行为时，ChatGPT-4帮助我们识别潜在问题并提供解决方案。
   - **优化**：对于性能关键的代码段，ChatGPT-4提供了优化技巧，我们的团队随后进行了验证和实现。
4. **文档和注释**：
   - **代码注释**：AI生成的注释被添加到代码中，以解释复杂的逻辑，确保代码的可维护性。
   - **文档编写**：ChatGPT-4协助创建了各个模块的全面文档，确保未来的开发者能够轻松理解和扩展项目。

AI贡献的说明

我们承认项目中的很大一部分代码和架构指导借鉴了ChatGPT-4。为了遵守项目的协作政策，所有AI生成的代码在代码库中都进行了适当的注释。这些注释明确指出了由ChatGPT-4生成的部分，并强调了我们团队所做的修改。

遵守协作政策

根据项目指南：

- **版本控制**：我们使用Git进行版本控制，确保所有贡献，无论是来自AI还是团队成员，都被正确记录。
- **AI注释**：代码中的每一段AI生成的代码都进行了清晰的注释，以符合项目的协作政策。

正如冲刺文档中所述，如果不遵守这些政策，将会导致分数扣减。我们已严格遵循这些指南，以确保项目开发过程的透明性和完整性。