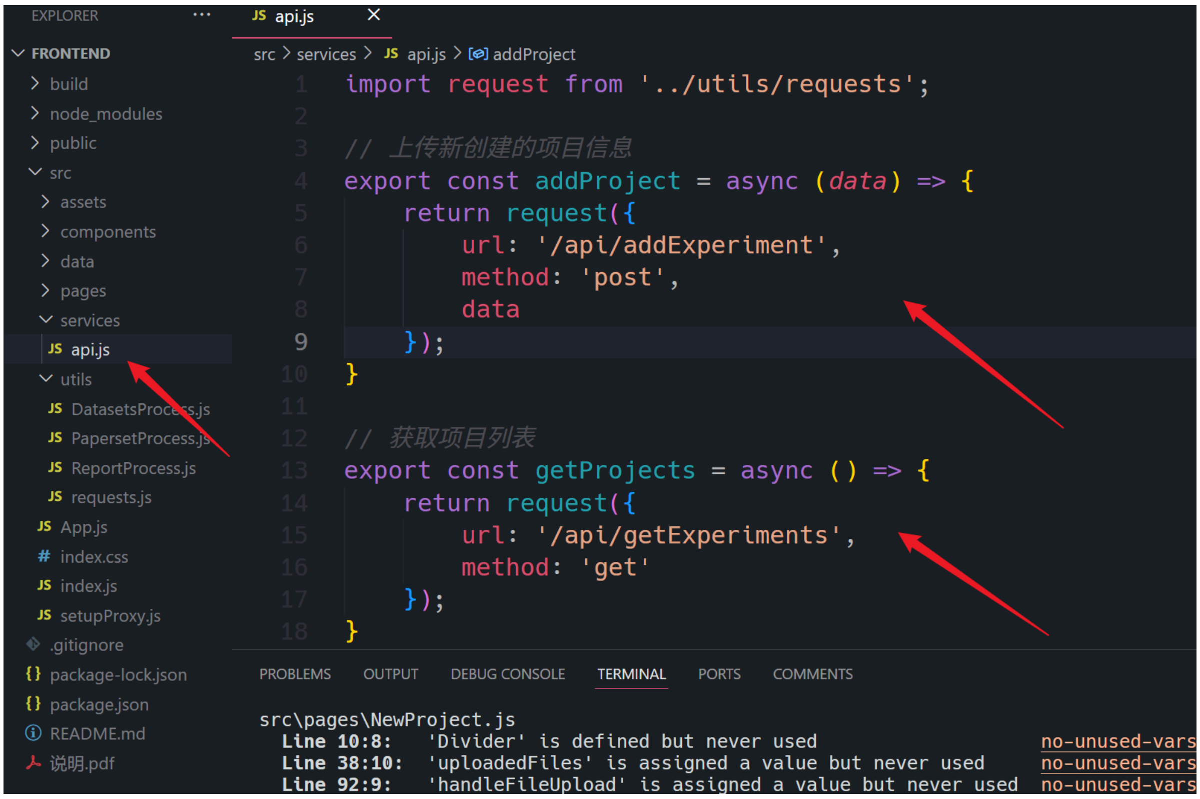
前端发送请求

通过 axios 发送请求，项目里把 axios 封装为 request 了。（这个不重要）

所有请求接口都写在 services 文件夹下的 api.js 文件里。



```javascript
import request from '../utils/requests';

// 上传新创建的项目信息
export const addProject = async (data) => {
    return request({
        url: '/api/addExperiment',
        method: 'post',
        data
    });
}

// 获取项目列表
export const getProjects = async () => {
    return request({
        url: '/api/getExperiments',
        method: 'get'
    });
}
```

```
src\pages\NewProject.js
  Line 10:8:   'Divider' is defined but never used                  no-unused-vars
  Line 38:10:  'uploadedFiles' is assigned a value but never used    no-unused-vars
  Line 92:9:   'handleFileUpload' is assigned a value but never used  no-unused-vars
```

比如已经写了两个请求：

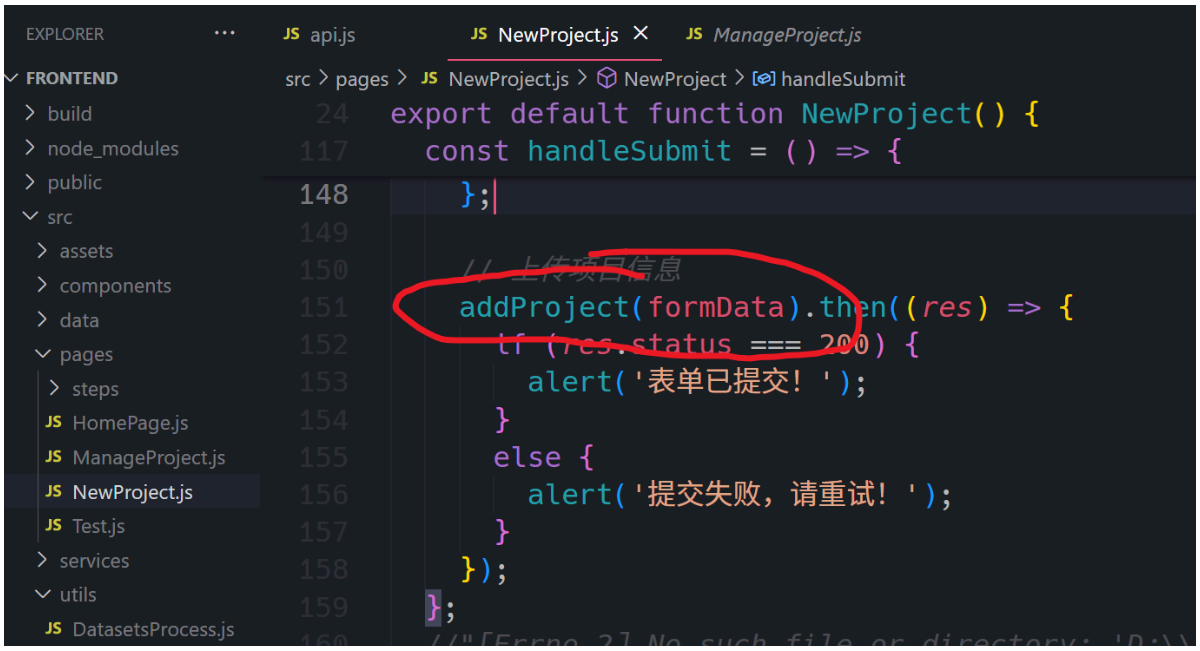
- addProject: post方法，把项目信息发送到后端
- getProjects: get 方法，获取已经存在的项目信息

为什么这里的url都以api开头？

主要是为了解决跨域问题，借助了 http-proxy-middleware 这个包，这里就不再展开。

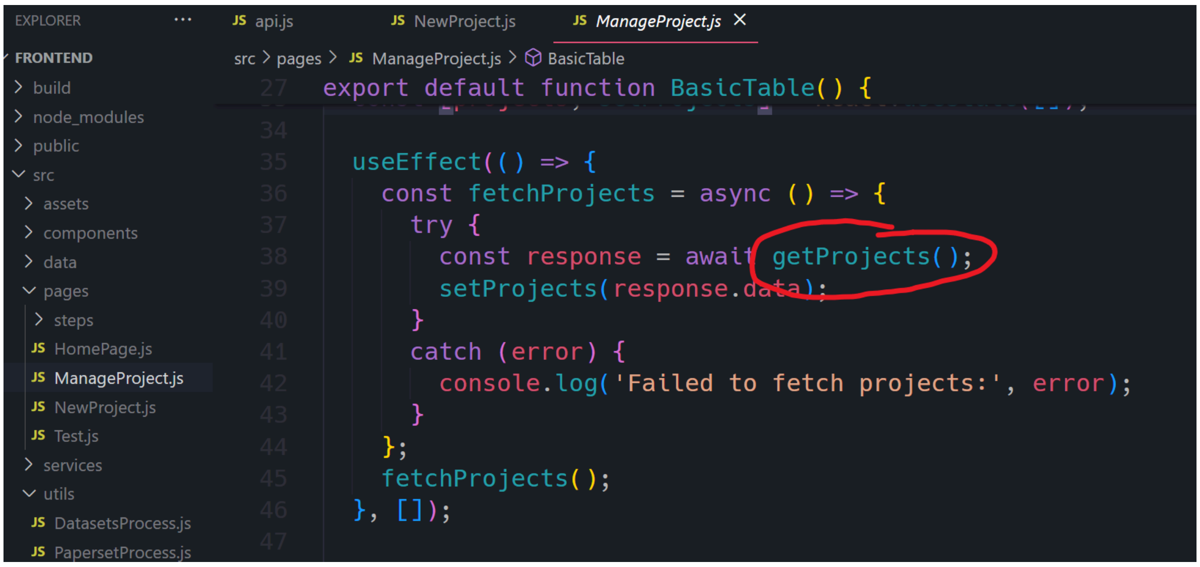
结果就是：`/api/addExperiment` 经过处理后变为 `http://127.0.0.1:8000/addExperiment`

然后在页面引入这两个函数并调用。



```
export default function NewProject() {
  const handleSubmit = () => {
    };

    // 上传项目信息
    addProject(formData).then((res) => {
      if (res.status === 200) {
        alert('表单已提交！');
      }
      else {
        alert('提交失败，请重试！');
      }
    });
  };
  //"[Errno 2] No such file or directory: 'D:\\
```



```
export default function BasicTable() {

  useEffect(() => {
    const fetchProjects = async () => {
      try {
        const response = await getProjects();
        setProjects(response.data);
      }
      catch (error) {
        console.log('Failed to fetch projects:', error);
      }
    };
    fetchProjects();
  }, []);
```

后端接收请求并返回数据

后端的接口都写在了 routers 文件夹。

以创建项目和管理项目为例：routers 文件夹下有个 experiments.py 文件，

首先定义了一个数据模型：和前端传来的数据相对应，存储实验相关信息

```
class PaperSet(BaseModel):
    PMC: bool
    PubMed: bool

class Dataset(BaseModel):
    GEO: bool
    NCBI: bool
    cBioPortal: bool

class Experiment(BaseModel):
    expName: str
    expPurpose: str
    expCondition: str
    expRequirement: str
```

```python
    paperset: PaperSet
    dataset: Dataset
    llmModel: str
    refNum: Optional[int] = None
    reviewerRound: Optional[int] = None
```

然后定义 addExperiment 接口，前端发送 `/api/addExperiment` 请求后，后端就会在这里接收到并处理。

```python
@router.post("/addExperiment")
async def add_experiment(experiment: Experiment):
    try:
        experiment_dict = experiment.model_dump()
        experiment_dict["id"] = generate_id()   # 生成一个唯一id
        experiments = read_experiments()        # 读取已经创建的项目信息
        experiments.append(experiment_dict)     # 加入新创建的项目信息
        write_experiments(experiments)          # 写入json文件

        # 如果向创建项目后立即调用llm，可以在这里加上调用llm的代码

        return {"message": "Experiment added successfully"}
    except Exception as e:
        return {"error": str(e)}
```

其他接口类似。

现在正在做的就是把**数据处理**相关的东西从前端迁移到fastapi里面，数据处理完全交给python来做，方便一点。

调用大模型也不用js里面的`child_process`了，直接在python文件里调用，方便很多。