

厚德 求真 励学 笃行

Web工程

Web Engineering

→ 汇报时间：2024年6月12日

→ 汇报人：刘昊昕

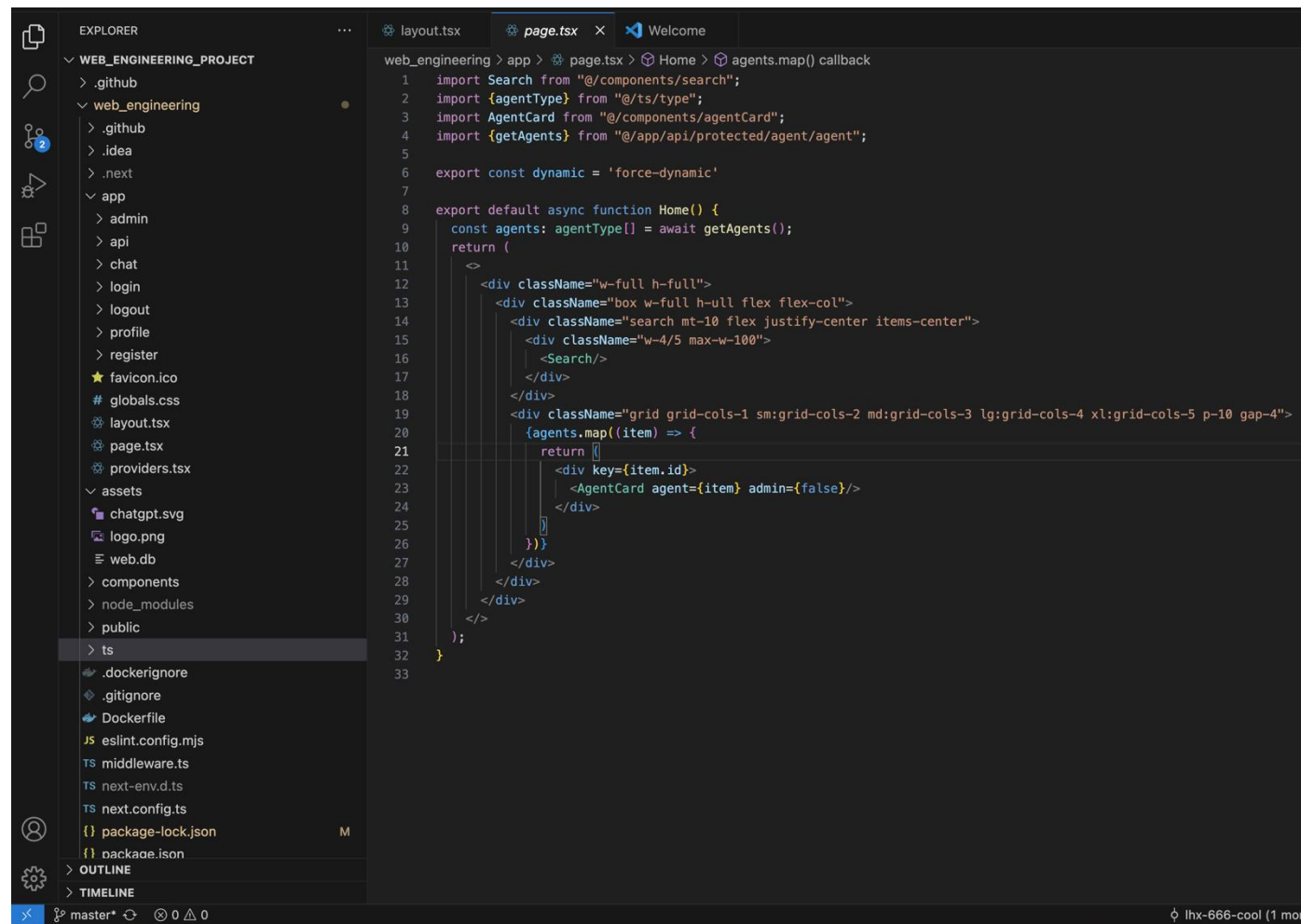


1 *Part One*

Web应用构建与部署



技术栈： NextJS Sqlite



The screenshot shows a VS Code editor with a project named 'WEB_ENGINEERING_PROJECT'. The file explorer on the left shows the project structure, including 'app', 'components', 'public', and 'ts' directories. The main editor displays the 'page.tsx' file, which contains the following code:

```
web_engineering > app > page.tsx > Home > agents.map() callback
1 import Search from "@components/search";
2 import {agentType} from "@ts/type";
3 import AgentCard from "@components/agentCard";
4 import {getAgents} from "@app/api/protected/agent/agent";
5
6 export const dynamic = 'force-dynamic'
7
8 export default async function Home() {
9   const agents: agentType[] = await getAgents();
10   return (
11     <div className="w-full h-full">
12       <div className="box w-full h-full flex flex-col">
13         <div className="search mt-10 flex justify-center items-center">
14           <div className="w-4/5 max-w-100">
15             <Search/>
16           </div>
17         </div>
18         <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 xl:grid-cols-5 p-10 gap-4">
19           {agents.map((item) => {
20             return (
21               <div key={item.id}>
22                 <AgentCard agent={item} admin={false}/>
23               </div>
24             )
25           })}
26         </div>
27       </div>
28     </div>
29   );
30 };
```

```
1 create table agent
2 (
3   name TEXT,
4   id integer
5   constraint agent_pk
6   primary key autoincr
7   desc TEXT,
8   content TEXT,
9   icon TEXT
10 );
11
12 create table favorites
13 (
14   userId int,
15   agentId int,
16   primary key (userId, agentId)
17 );
18
19 create table history
20 (
21   sessionId TEXT not null
22   constraint history_pk
23   primary key,
24   userId integer not null,
25   title text not null,
26   message TEXT not null,
27   "update" TEXT not null
28 );
29
30 create index history_userId_index
31 on history (userId);
32
33 create table sqlite_master
34 (
35   type TEXT,
36   name TEXT,
37   tbl_name TEXT,
38   rootpage INT,
39   sql TEXT
```

Web应用部署

使用Github Action打包

使用Docker部署

```
1 # 第一阶段: 构建阶段
2 # 使用官方 Node.js 镜像作为基础镜像
3 FROM node:lts-alpine as builder
4
5 # 设置工作目录
6 WORKDIR /app
7
8 # 复制 package.json 和 package-lock.json 文件, 优先复制可以利用 Docker 缓存
9 COPY package*.json ./
10
11 # 安装项目依赖
12 # 如果你的项目使用 pnpm, 请使用 pnpm install
13 # 如果你的项目使用 yarn, 请使用 yarn install
14 RUN npm install
15
16 # 复制项目所有文件
17 COPY . .
18
19 # 构建 Next.js 应用
20 # 这里使用了 NEXT_TELEMETRY_DISABLED=1 来禁用构建过程中的遥测, 可选
21 RUN NEXT_TELEMETRY_DISABLED=1 npm run build
22
23 # 第二阶段: 运行阶段
24 # 使用一个轻量级的 Node.js 镜像作为运行阶段的基础镜像
25 # alpine 是一个非常小的 Linux 发行版, 适合生产环境
26 FROM node:lts-alpine
27
28 # 设置工作目录
29 WORKDIR /app
30
31 # 从构建阶段复制构建好的 Next.js 应用文件
32 COPY --from=builder /app/.next ./.next
33 COPY --from=builder /app/public ./public
34 COPY --from=builder /app/package.json ./package.json
35 # 复制你可能需要的其他文件, 例如 prisma 客户端等
36 # COPY --from=builder /app/node_modules/.prisma ./prisma
37
38 # 安装生产环境依赖
39 # --omit=dev 参数只安装生产环境依赖
40 RUN npm install --omit=dev
41
42 # 暴露 Next.js 默认的服务端口
43 EXPOSE 3000
44
45 # 设置环境变量, 告诉 Next.js 启动生产模式
46 ENV NODE_ENV=production
47
48 # 启动 Next.js 应用
49 CMD ["npm", "start"]
```

Build and Push Docker Image #3

Summary

Jobs

build

Run details

Usage

Workflow file

Annotations

18 warnings

build

succeeded 2 hours ago in 2m 39s

Set up job

Checkout code

Set up Docker Buildx

Login to DockerHub

Build and push Docker image

Post Build and push Docker image

Post Login to DockerHub

Post Set up Docker Buildx

Post Checkout code

Complete job

2 *Part Two*

Web应用测试



Cypress端到端测试



3 *Part Three*

Web应用运维



SEO策略

1. 服务器端渲染 (SSR) 和静态生成 (SSG):

Next.js 的核心优势之一就是支持 SSR 和 SSG。这使得搜索引擎爬虫可以直接抓取到完整的 HTML 内容，而不是一个空的 HTML 文件，这对 SEO 至关重要。搜索引擎更喜欢能够直接读取内容，而不是依赖 JavaScript 渲染。

2. 元标签 (Meta Tags) 优化:

优势: 元标签是告诉搜索引擎关于页面内容的关键信息。Next.js 使得在每个页面轻松管理元标签成为可能。

方法:

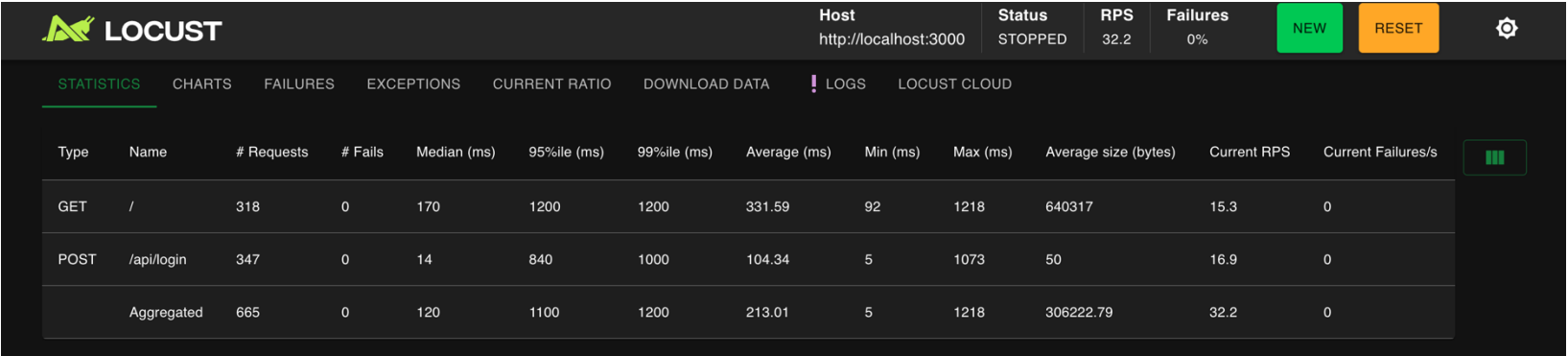
`<title>` 标签: 每个页面都应该有一个独特、描述性强且包含主要关键词的标题。Next.js 允许你在页面组件中轻松设置 `<title>`。
`<meta name="description">` 标签: 提供一个简洁、有吸引力且包含关键词的页面描述。这通常会显示在搜索结果中。
`<meta name="keywords">` 标签 (可选但仍有一定作用): 虽然关键词标签的重要性不如以前，但仍然可以包含一些相关的关键词。
`<meta name="robots">` 标签: 控制搜索引擎爬虫的行为，例如是否允许索引页面 (index, noindex) 和是否跟踪链接 (follow, nofollow)。
`<meta property="og:...">` 和 `<meta name="twitter:...">` (开放图谱和 Twitter 卡片): 优化页面在社交媒体上的分享展示，虽然不是直接的搜索引擎排名因素，但可以带来更多流量。Next.js 可以通过 `next/head` 组件轻松实现

4 *Part four*

Web应用性能与可用性分析



Locust 压测



我们使用了locust对部分功能进行了压测，结果如下：在高并发场景下累计发起665次请求，吞吐量稳定在32.2 RPS，全程零失败，展示了系统的可靠性与稳定性。登录接口平均响应仅104ms，中位14ms，主页面中位响应170ms，整体平均213ms以内，证明服务在绝大多数请求中都能实现低延迟。接口响应波动较小，资源消耗平稳，说明了我们的系统性能较好，可用性较高。

性能分析与调优

项目中已采取的性能优化措施

1. 使用 `dynamic import` 对登录表单组件进行懒加载，减少首屏 JS 体积。
2. 启用 Next.js 的图片优化 (`next/image`)，对 logo、背景等图片资源进行懒加载和格式压缩

还可以优化的地方：

1. 静态资源体积与加载效率

登录页面、主页面等依赖的 JS、CSS 资源较多，若未压缩会影响加载速度。

动态组件未懒加载时，首屏资源体积较大。

2. 首屏渲染速度

页面未设置合适的 loading 状态，用户感知加载慢。

可用性分析与调优

项目中已采取的性能优化措施

1. 使用 Tailwind CSS 保证响应式布局，适配多终端。
2. 支持深色模式，适应不同用户偏好。

还可以优化的地方：

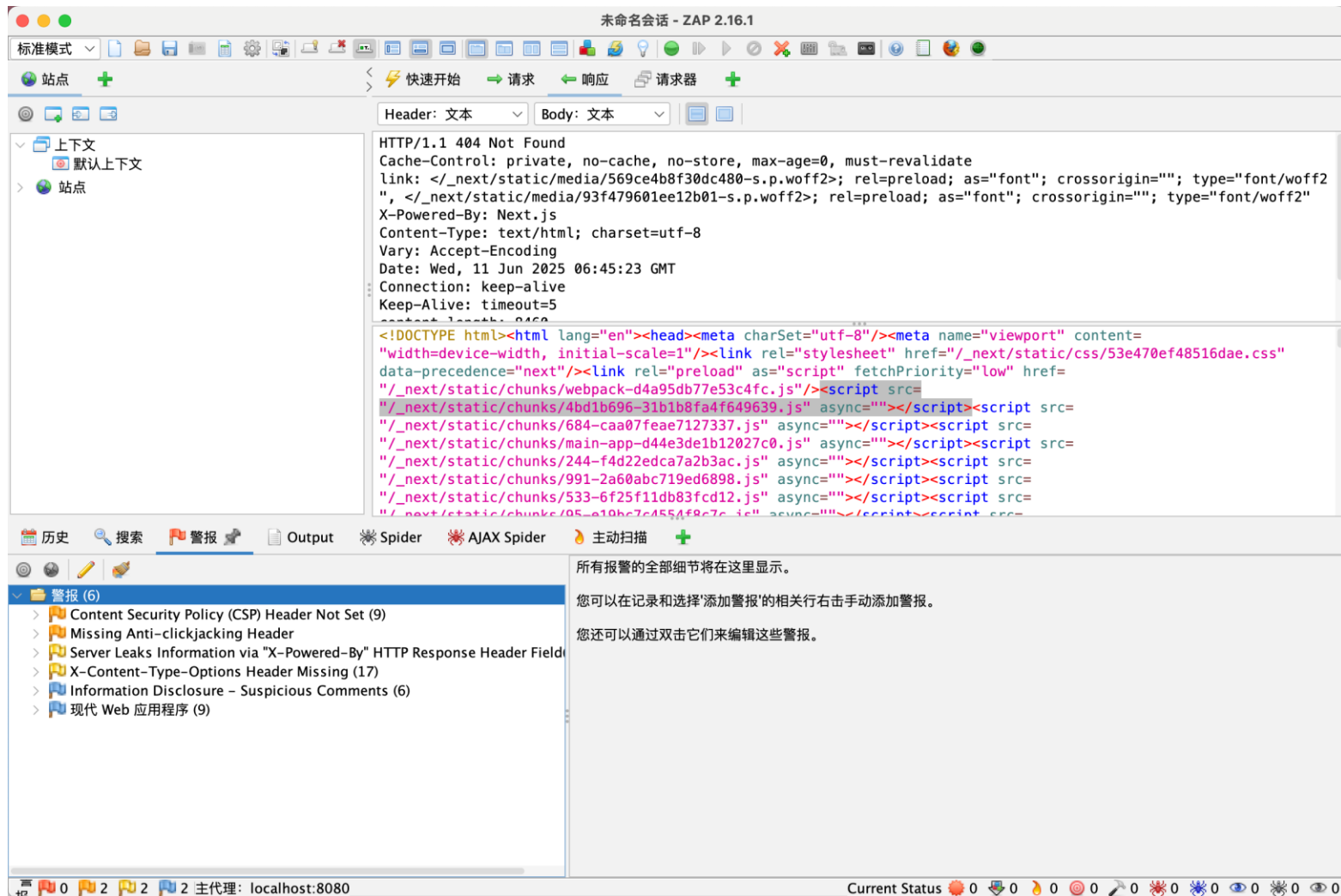
1. 登录、注册、聊天等按钮添加 loading 状态，提升操作反馈。
2. 登录、注册、主页面结构简单，但辅助入口不够突出。

5 *Part Five*

Web应用安全分析



使用ZAP进行攻击



使用Sqlmap进行攻击

```
[H]
[ ] {1.9.5.22#dev}
|_| . [ ] |'| .|
|_| [ ] |_| ,|_|
|_| IV... |_| https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 14:59:34 /2025-06-11/

JSON data found in POST body. Do you want to process it? [Y/n/q] Y
[14:59:34] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('auth_token=eyJhbGciOiJmGTNVSQoUI'). Do you want to use those [Y/n] Y
[14:59:34] [INFO] testing if the target URL content is stable
[14:59:34] [INFO] target URL content is stable
[14:59:34] [INFO] testing if (custom) POST parameter 'JSON username' is dynamic
[14:59:35] [INFO] (custom) POST parameter 'JSON username' appears to be dynamic
[14:59:35] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON username' might not be injectable
[14:59:35] [INFO] testing for SQL injection on (custom) POST parameter 'JSON username'
[14:59:35] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:59:35] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[14:59:35] [INFO] testing 'Generic inline queries'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[14:59:35] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:59:35] [WARNING] (custom) POST parameter 'JSON username' does not seem to be injectable
[14:59:35] [INFO] testing if (custom) POST parameter 'JSON password' is dynamic
[14:59:35] [INFO] (custom) POST parameter 'JSON password' appears to be dynamic
[14:59:35] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON password' might not be injectable
[14:59:35] [INFO] testing for SQL injection on (custom) POST parameter 'JSON password'
[14:59:35] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:59:35] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[14:59:35] [INFO] testing 'Generic inline queries'
[14:59:35] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:59:35] [WARNING] (custom) POST parameter 'JSON password' does not seem to be injectable
[14:59:35] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
```

还可以做什么？

密码加盐后使用MD5做哈希再保存在数据库里。

在注册登陆接口使用人机验证码。

使用 WAF 对应用进行防护。

给前端代码加入混淆。

.....



XDU

THANK FOR ATTENTION

厚德 求真 励学 笃行