


# AI coding 从0到1开发后端服务

 Hello~，这几天在抖音和B站总刷到AI coding的视频，我觉得我们自己也有必要感受一下其牛逼之处。或者验证一下现在的 AI coding 是否真的很牛逼可以稍微调试即可运行一套较为复杂的系统。

经过我一天的体验，事实证明它真的很牛逼，所以我打算将这个过程分享出来，大家也可以自己体验一下。

项目 GitHub: [https://github.com/G-Pegasus/Zhiguang\\_AICoding](https://github.com/G-Pegasus/Zhiguang_AICoding)

## 前置准备

现在市面上的 AI coding 产品很多，比如 Cursor，Trae，Codex 等等。大家如果想感受下国外模型的效果，可以买一下 ChatGPT Plus 里面包含 Codex，或者 Cursor 的 Claude code，当然了，这些有点小贵.....。如果想体验下便宜的，大家可以下载字节的 TRAE 国际版，PRO 只要 3 美刀。

**我买了 ChatGPT Plus，然后它的 codex 可以以插件的形式集成到 TRAE 里面，也很好用。**

我把一些前置准备列举一下：

1. TRAE，字节的 AI coding IDE
2. IDEA 和 Java 开发环境。TRAE 用来生成代码，IDEA 用来调试代码
3. MySQL，本地部署或者连远程服务器都可以；
4. Redis，本地部署或者连远程服务器都可以；

好，就这么多，一点也不难吧~

## 产品原型

想体验就要自己想一个产品原型，即你想让AI帮你实现什么，比如老套的学生管理系统，外卖项目，或者自己的新奇想法都可以。

然后发给ChatGPT，豆包等等，让它们帮你完善总结你的想法，生成一份比较详细地产品文档或者计划书。

你按我说的内容生成一版产品计划书：

- 1、登录注册，用户信息编辑与修改
- 2、发布知识：一期只包含图文和纯文档（md渲染）
- 3、主页Feed流与知识详情页，展示用户发布的图文知识或纯文档知识，类似小红书混合排布
- 4、用户之间的关注取关，对知识的点赞与收藏
- 5、在知识内的评论互动功能
- 6、技术栈采用 Java 21 + MySQL 8.0 + Redis

知光 产品计划书（feed+图文+文档）mvp V0

复制 编辑 下载

我按你的 1-6 点要求，已经在右侧生成了《知光 产品计划书（Feed+图文+文档）MVP v0.9》：覆盖登录注册与资料编辑、图文与Markdown文档发布、首页混合Feed与详情页、关注/点赞/收藏、评论互动，以及Java 21 + MySQL 8.0 + Redis 7的技术方案（ER、API、DDL、缓存策略、性能与安全、12周里程碑）。

要不要再加一份「接口 Swagger/OpenAPI YAML」或把 DDL/错误码表导出成可下载的 Excel/SQL 文件？

📄 🍌 🗨️ ...



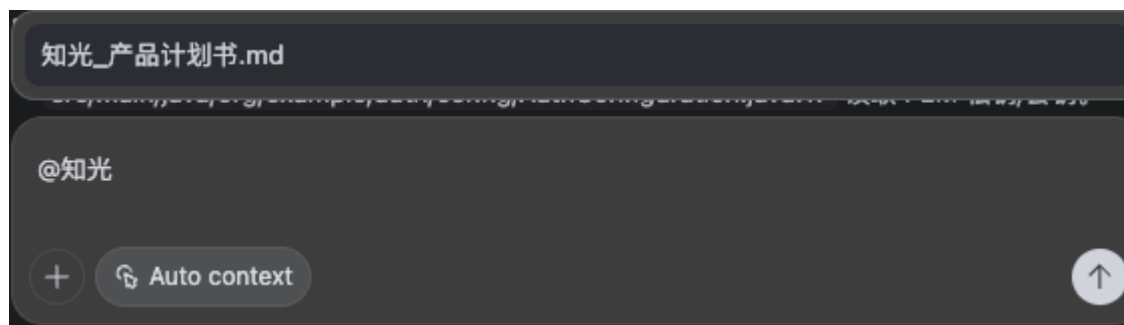
根据它生成的文档，你再按照自己的想法进行补充和完善

最好里面是有规划的，产品一期做什么功能，产品二期做什么功能

比如，一期做登录注册、用户信息编辑和修改；二期做帖子的发布和展示等等，这样发给 AI 的时候比较好梳理逻辑，你可以告诉它，用Java 的 Spring 技术栈完成计划书中一期内容。

## 开发与调试

将你的产品原型或是计划书，导入到 TRAE 里面，然后@这个文件引用，让它按照这个计划书的内容做详细的技术方案，根据技术方案，模型可以更好地理解并创作。



## Codex

← [知光\_产品计划书.md](知光\_产品计划书.md) 按照这个计划书，生成登录注册的技术...



@知光\_产品计划书.md 按照这个计划书，生成登录注册的技术方案md文件

Finished working

整理并输出了登录注册端到端技术方案，覆盖范围、流程、架构、数据、接口、安全与测试要求。  
**主要内容**

- 新增登录注册技术方案文档，细化注册/登录/找回流程、模块划分、数据模型及接口契约，结合 Redis 限流、JWT 会话和安全基线要求，贴合产品计划书约束；详见 登录注册技术方案.md:1 .

**建议下一步**

1. 与产品、前端、风控团队评审方案，确认接口契约与限流策略。
2. 根据方案拆解开发任务并同步第三方短信/邮件渠道配置。
3. 准备测试用例，覆盖验证码、JWT 轮换与安全场景。

好的，接下来在这个Project里面用Spring生态完成登录注册功能的开发

Failed

- Determining need for project plan  
|
- Reviewing POM file  
|
- Planning Spring Boot setup  
|
- Checking for source files  
|
- Checking for list requirement  
|
- Drafting multi-step Spring Boot plan  
|

这里生成了大部分的逻辑，但是和我的想法有很大的出入。文件里面一堆 get set，并且使用了我不熟悉的 ORM 框架 Hibernate。所以我让它加入 Lombok 和 MyBatis 依赖，并使用这两者进行开发。

使用 MyBatis 框架开发，并且使用lombok简化代码

Finished working

实现了 Spring Boot 登录/注册闭环，整合 MyBatis + Lombok 持久层、Redis 验证码与 JWT 会话体系。

- 端到端业务编排： `src/main/java/org/example/auth/service/AuthService.java:55` 完成验证码发送、注册、登录、刷新/注销会话与重置密码，并在成功后写入 Redis 刷新令牌及登录日志。
- REST 接口与鉴权： `src/main/java/org/example/auth/api/AuthController.java:38` 提供 `/send-code`、`/register`、`/login` 等入口，  
`src/main/java/org/example/auth/config/SecurityConfig.java:18` 限制开放接口并启用 JWT 资源服务器。
- JWT 与密钥管理： `src/main/java/org/example/auth/token/JwtService.java:31` 用 RS256 生成/解析 Access & Refresh Token，配合  
`src/main/java/org/example/auth/config/AuthConfiguration.java:17` 读取 PEM 私钥/公钥。
- 数据与验证码支撑： `src/main/java/org/example/auth/user/UserMapper.java:7` +  
`src/main/resources/mapper/UserMapper.xml:5` 切换至 MyBatis，  
`src/main/java/org/example/auth/verification/RedisVerificationCodeStore.java:26` 与  
`VerificationService.java:18` 管理验证码存储、限流与校验。
- 配置与测试： `src/main/resources/application.yml:1` 补充数据源/Redis/JWT 配置与示例 RSA 密钥，  
`src/test/java/org/example/auth/token/JwtServiceTest.java:12` 验证令牌签发与解码。

当然，也不是一帆风顺的，AI 生成的代码会有点小错误，比如某个库没有这个方法，但是却无中生有。

这时候可以直接引用报错的那个文件，并把报错信息发给它修改。最后也是成功修改了。

The type `org.springframework.security.oauth2.jwt.JoseHeader` is not visible 解决这个问题

Finished working

已将 `JwtService` 中的 Header 构建换成

`org.springframework.security.oauth2.jose.jws.JwsHeader`，避免使用不可见的 `JoseHeader`。现在类开头导入 `JwsHeader`，并在 `encodeToken`、`encodeRefreshToken` 两处调用 `JwsHeader.withAlgorithm(SignatureAlgorithm.RS256)` 生成 Header（路径：  
`src/main/java/org/example/auth/token/JwtService.java:57`、`:71`）。重新编译即可。

@`JwtService.java` The import `org.springframework.security.oauth2.jose.jws.JwsHeader` cannot be resolved 还是不行

Finished working

`JwtService` 现在直接使用 `JwtEncoderParameters.from(claims)`，不再依赖不可见的 `JoseHeader/JwsHeader`。相关更新位于  
`src/main/java/org/example/auth/token/JwtService.java:46` 和 `:60`，两处都改为 `jwtEncoder.encode(JwtEncoderParameters.from(claims))`。编译即可通过。

最后，可以让它生成 SQL 建表文件以及详细的接口文档

生成目前代码中需要的数据库表，并把建表语句放在单独的SQL文件里

Finished working

新增 `db/schema.sql`，囊括当前代码用到的 MySQL 表结构：

- `users`：包含手机号/邮箱唯一索引、JSON 标签字段以及创建/更新时间列，对应 `UserMapper` 的 CRUD。
- `login_logs`：保存登录审计信息，并按 `user_id + created_at` 建索引，对应 `LoginLogMapper` 的写入。

导入办法：在目标库执行 `mysql -u<user> -p<pass> zhiguang < db/schema.sql` 即可完成建表。

而且用 TRAE 的一点好处是，还可以用 TRAE 自带的 AI 对工程修改，对生成的代码不懂的问题也可以问它

完成项目所有接口的解释及请求示例，并放在单独的md文件里

## Builder

### > 思考过程

I've identified the controller; now I'll scan for all REST endpoints across the project.

> 在工作区搜索 'Find all REST controllers and their mappings: classes annotated with @RestController and methods annotate...

⌕ Thought

📄 src/main/java/org/example/auth/config/SecurityConfig.java 1-200

⌕ Thought

> 在工作区搜索 'Locate PasswordResetRequest.java and its fields to document the request body.'

⌕ Thought

> 在工作区搜索 'Locate LogoutRequest.java to confirm field structure for the documentation.'

🔄 0/3 已完成

🔄 汇总并确认所有 REST 接口与鉴权规则

☐ 为每个接口编写说明、请求与响应示例

☐ 创建 docs/API接口文档.md 并填入完整内容

⌕ Thought

🔄 1/3 已完成

✅ 汇总并确认所有 REST 接口与鉴权规则

🔄 为每个接口编写说明、请求与响应示例

☐ 创建 docs/API接口文档.md 并填入完整内容

我将创建一份完整的接口文档并保存为独立的 Markdown 文件。

📄 docs/API接口文档.md

+328 -0

查看变更

🔄 2/3 已完成

✅ 汇总并确认所有 REST 接口与鉴权规则

✅ 为每个接口编写说明、请求与响应示例

## 开发效果

综合来看，效果相当好，繁琐的 Maven, 项目配置文件 yaml, 包括 MyBatis 生成的 Mapper 文件都可以搞定，并且很少有错误，有一些小错误也很容易修改，大大节省了开发时间。

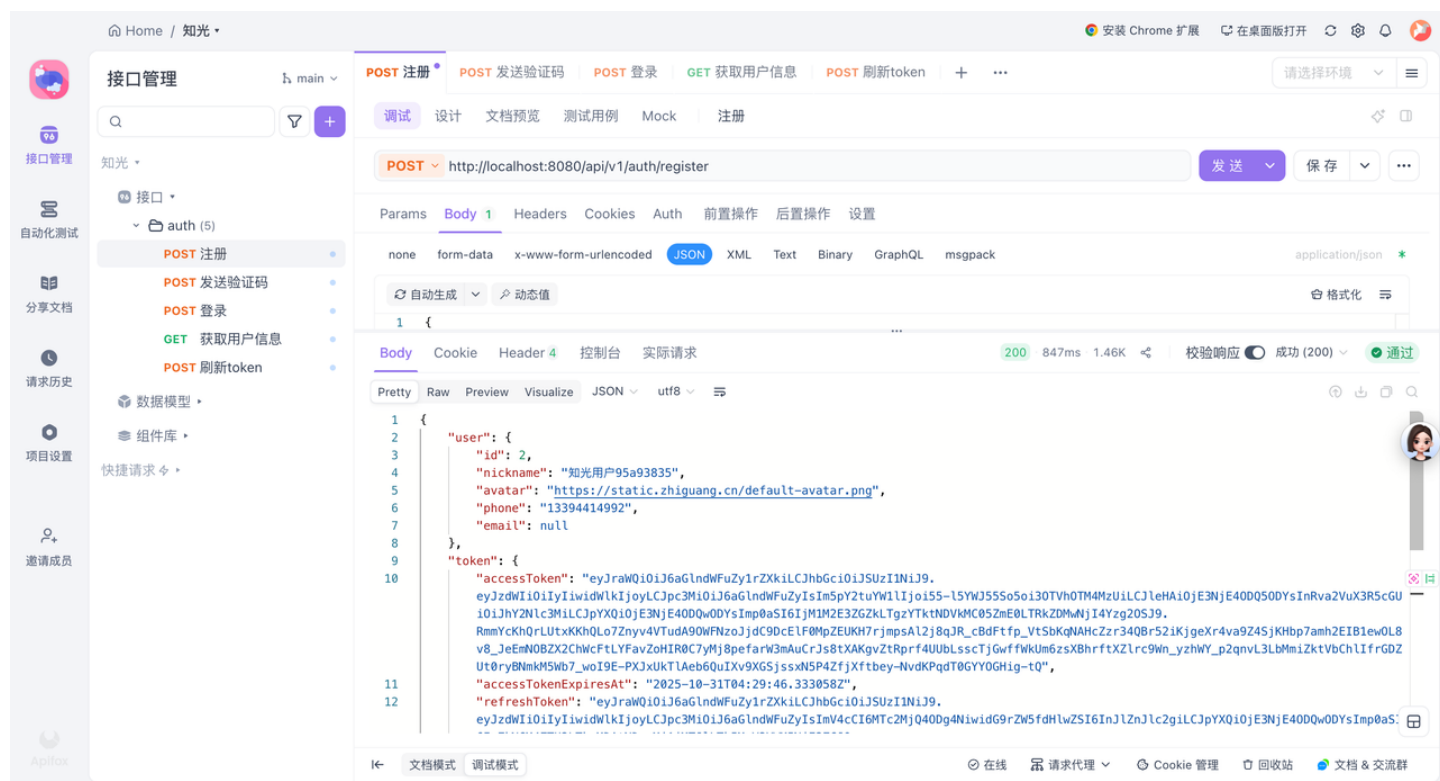
只需要开发者补充一下 MySQL 和 Redis 地址即可

```
src > main > resources > application.yml

1  spring:
2    application:
3      name: zhiguang-auth
4    datasource:
5      url: jdbc:mysql://[redacted]306/zhiguang?allowPublicKeyRetrieval=true&sslMode=DISABLED
6      username: root
7      password: [redacted]
8      driver-class-name: com.mysql.cj.jdbc.Driver
9    jackson:
10     serialization:
11       write-dates-as-timestamps: false
12    data:
13     redis:
14       host: [redacted]
15       port: 6379
16
17  mybatis:
18    mapper-locations: classpath*:mapper/*.xml
19    configuration:
20      map-underscore-to-camel-case: true
```

## 接口运行

这次的登录注册模块，有注册、登录、登出、修改密码、发送验证码、刷新Token，获取用户信息等接口，最后测试的时候，都是一次成功，部分效果图如下：





Home / 知光

安装 Chrome 扩展 在桌面版打开

接口管理

main

POST 注册 POST 发送验证码 POST 登录 GET 获取用户信息 POST 刷新token

请选择环境

知光

接口

auth (5)

POST 注册

POST 发送验证码

POST 登录

GET 获取用户信息

POST 刷新token

数据模型

组件库

快捷请求

POST

http://localhost:8080/api/v1/auth/send-code

发送

保存

Params

Body 1

Headers

Cookies

Auth

前置操作

后置操作

设置

none

form-data

x-www-form-urlencoded

JSON

XML

Text

Binary

GraphQL

msgpack

application/json

自动生成

动态值

格式化

1

2

3

4

5

{

"scene": "REGISTER",

"identifierType": "PHONE",

"identifier": "13394414992"

}

Body

Cookie

Header 4

控制台

实际请求

200

335ms

67B

校验响应

成功 (200)

通过

Pretty

Raw

Preview

Visualize

JSON

utf8

1

2

3

4

5

{

"identifier": "13394414992",

"scene": "REGISTER",

"expireSeconds": 300

}

文档模式

调试模式

在线

请求代理

Cookie 管理

回收站

文档 & 交流群

Home / 知光

安装 Chrome 扩展 在桌面版打开

接口管理

main

POST 注册 POST 发送验证码 POST 登录 GET 获取用户信息 POST 刷新token

请选择环境

知光

接口

auth (5)

POST 注册

POST 发送验证码

POST 登录

GET 获取用户信息

POST 刷新token

数据模型

组件库

快捷请求

GET

http://localhost:8080/api/v1/auth/me

发送

保存

Params

Body

Headers 1

Cookies

Auth

前置操作

后置操作

设置

Headers

参数名

参数值

类型

说明

Authorization

Bearer eyJraWQiOiJ6aGlndWVhZy1rZX

string

\*

添加参数

Body

Cookie

Header 4

控制台

实际请求

200

182ms

134B

校验响应

成功 (200)

通过

Pretty

Raw

Preview

Visualize

JSON

utf8

1

2

3

4

5

6

7

{

"id": 2,

"nickname": "知光用户95a93835",

"avatar": "https://static.zhiguang.cn/default-avatar.png",

"phone": "13394414992",

"email": null

}

文档模式

调试模式

在线

请求代理

Cookie 管理

回收站

文档 & 交流群