باسمه تعالی

ارائه‌ی درس برنامه‌سازی وب

مستند Chat Client in Spring AI

مدرس:

یحیی پورسلطانی

مسئول تیم دستیاران آموزشی:

محمدحسین حاجی سیدسلیمان

اعضای تیم:

امیرحسین محمدزاده، دنیا نوابی، آریا ترابی

زمستان ۱۴۰۳

فهرست

مقدمه

در دنیای دیجیتال امروز، تعاملات میان کاربران و سامانه‌های نرم‌افزاری به‌طور فزاینده‌ای به سمت استفاده از هوش مصنوعی و چت‌بات‌های پیشرفته متمایل شده است. یکی از مهم‌ترین چالش‌ها در این حوزه، زمان پاسخ‌دهی و تجربه‌ی کاربری بهینه است. در سیستم‌های سنتی، کاربران پس از ارسال پیام باید تا دریافت کامل پاسخ منتظر بمانند که این امر منجر به افزایش زمان انتظار و کاهش بهره‌وری می‌شود.
با پیشرفت روزافزون فناوری‌های هوش مصنوعی و یادگیری ماشین، استفاده از این تکنولوژی‌ها در نرم‌افزارهای مختلف به شدت افزایش یافته است. یکی از حوزه‌های مهم کاربرد هوش مصنوعی، توسعه چت‌بات‌ها و سیستم‌های مکالمه محور (Chat Clients) است که به منظور تعامل موثرتر با کاربران طراحی می‌شوند. این گزارش به بررسی چگونگی توسعه یک مشتری چت با استفاده از فریم‌ورک Spring AI می‌پردازد و به صورت مفصل به جنبه‌های مختلف این فرآیند می‌پردازد.

بیان مسئله

در بسیاری از سیستم‌های خدمات، آموزشی، پشتیبانی مشتری و ربات‌های مشاوره‌ای، کاربران به پاسخ‌های سریع و دقیق نیاز دارند. به عنوان مثال:

- پشتیبانی مشتری: کاربران هنگام تعامل با یک ربات پشتیبانی نمی‌خواهند برای دریافت یک پاسخ کامل منتظر بمانند. آنها انتظار دارند که بتوانند بخش‌های پاسخ را در حین پردازش اطلاعات توسط ربات مشاهده کنند.
- آموزش الکترونیکی: در سیستم‌های آموزش آنلاین، دانش‌آموزان ممکن است در مورد موضوع خاصی سوالاتی داشته باشند. اگر پاسخ‌ها به صورت بخش بخش ارائه شوند، آنها قادر خواهند بود آن را بهتر درک و تحلیل کنند.
- مشاوره هوشمند: در نرم‌افزارهای مشاوره و تحلیل داده، ربات‌های چت پیشرفته می‌توانند اطلاعات را مرحله به مرحله ارائه دهند تا کاربران بتوانند بر اساس اطلاعات ارائه شده تصمیمات بهتری اتخاذ کنند.

چالش‌های موجود در روش‌های سنتی

1. تأخیر در پردازش و نمایش اطلاعات: در چت‌بات‌های سنتی، پردازش درخواست‌ها به‌صورت یکجا انجام می‌شود و پاسخ تنها پس از تکمیل پردازش به کاربر نمایش داده می‌شود. این امر منجر به افزایش زمان انتظار کاربر و ایجاد تجربه‌ی کاربری نامطلوب می‌شود.

2. عدم نمایش پیشرفت پردازش: در سیستم‌های غیر استریمینگ، کاربران هنگام ارسال درخواست‌های پیچیده یا پردازش‌های طولانی، نمی‌توانند متوجه شوند که سامانه در حال انجام پردازش است یا نه.
3. افزایش بار پردازشی سرور: ارسال پاسخ‌های حجیم به‌صورت یکجا می‌تواند منجر به افزایش مصرف حافظه و منابع پردازشی شود که بهینه‌سازی عملکرد سیستم را دشوار می‌کند.
4. محدودیت تعامل آنی با کاربر: در سیستم‌های بلادرنگ (Real-time)، چت‌بات‌های سنتی ممکن است نیاز به زمان طولانی برای پردازش داشته باشند، که در شرایط خاص (مانند خدمات اورژانسی) غیرقابل‌قبول است.

راه‌حل با استفاده از چت‌بات استریمینگ

چت‌بات استریمینگ با استفاده از Spring Boot و Spring AI، امکان ارسال تدریجی پاسخ‌ها را فراهم می‌کند. این روش نه‌تنها باعث بهبود تجربه‌ی کاربری می‌شود، بلکه مشکلات پردازشی و تأخیرهای ناشی از روش‌های سنتی را نیز کاهش می‌دهد. راه‌حل پیشنهادی شامل موارد زیر است:

1. استفاده از پردازش و ارسال داده‌ها به‌صورت استریمینگ: با استفاده از Spring WebFlux و Spring AI، پاسخ‌ها به‌صورت جریان داده (Streaming) برای کاربر ارسال می‌شوند، به این صورت که بخش‌های پردازش‌شده‌ی پاسخ بدون نیاز به تکمیل کل پردازش، نمایش داده خواهند شد.
2. افزایش تعامل‌پذیری و کاهش زمان انتظار: کاربران می‌توانند همزمان با پردازش درخواستشان، بخشی از پاسخ را دریافت کرده و فرآیند تعامل را بهبود دهند.
3. مدیریت بهینه‌ی منابع سرور: به جای ارسال یک پاسخ حجیم و پردازش تمام داده‌ها به‌صورت یکجا، داده‌ها به‌صورت تدریجی پردازش و ارسال می‌شوند که باعث کاهش بار پردازشی سرور خواهد شد.
4. ایجاد تجربه‌ی کاربری پیشرفته‌تر: نمایش تدریجی اطلاعات و تعامل همزمان کاربر با چت‌بات باعث می‌شود که کاربران حس واقعی‌تری از مکالمه‌ی طبیعی داشته باشند.

سناریوی کاربردی

تصور کنید یک سازمان بزرگ خدمات مشتریان دارد که روزانه هزاران درخواست پشتیبانی دریافت می‌کند. در مدل سنتی:

- کاربران باید برای دریافت پاسخ‌های طولانی مدت‌زمان زیادی منتظر بمانند.
- پاسخ‌های طولانی و یکجا باعث کاهش تعامل با کاربر و خستگی او می‌شود.

- بار پردازشی بالا برای پردازش همزمان چندین درخواست روی سرور تحمیل می‌شود.

اما با استفاده از چت‌بات استریمینگ:

- پاسخ‌ها به‌صورت تدریجی نمایش داده می‌شوند و کاربر سریع‌تر اطلاعات اولیه را دریافت می‌کند.
- تجربه‌ی کاربری بهبود می‌یابد و کاربر احساس تعامل بهتری با چت‌بات دارد.
- پردازش داده‌ها بهینه‌تر شده و مصرف منابع سرور کاهش می‌یابد.

Spring AI چیست؟

Spring AI یک فریم‌ورک قدرتمند است که به توسعه‌دهندگان کمک می‌کند تا برنامه‌های هوشمند و کاربردی بسازند. این فریم‌ورک بخشی از اکوسیستم Spring است و ابزارها و قابلیت‌های هوش مصنوعی را در اختیار برنامه‌نویسان قرار می‌دهد تا بتوانند نرم‌افزارهای پیشرفته‌تری ایجاد کنند.

یکی از ویژگی‌های مهم Spring AI توانایی پردازش زبان طبیعی است. این یعنی برنامه‌ها می‌توانند متن‌هایی که کاربران می‌نویسند را درک کنند، احساسات موجود در جملات را تحلیل کنند، زبان‌ها را به یکدیگر ترجمه کنند و کارهای مشابه انجام دهند. همچنین این فریم‌ورک از یادگیری ماشین پشتیبانی می‌کند، به این معنا که می‌توان مدل‌های پیش‌بینی ایجاد کرد و سیستم‌های هوشمند طراحی کرد که با دریافت داده‌های جدید عملکرد بهتری داشته باشند.

قابلیت دیگر Spring AI، امکان پیاده‌سازی و آموزش شبکه‌های عصبی است. این شبکه‌ها در حل مسائل پیچیده مثل شناسایی تصاویر، تحلیل داده‌های بزرگ و تصمیم‌گیری‌های خودکار به کار می‌روند. همچنین این فریم‌ورک می‌تواند به سرویس‌های ابری مختلف مثل AWS، Azure و Google Cloud متصل شود تا از قدرت پردازشی و ذخیره‌سازی این سرویس‌ها استفاده کند.

Spring AI به‌گونه‌ای طراحی شده که بتواند حجم زیادی از داده‌ها و درخواست‌ها را مدیریت کند، بنابراین برای پروژه‌هایی که نیاز به مقیاس‌پذیری دارند، گزینه‌ی مناسبی است. این فریم‌ورک با ابزارها و کتابخانه‌هایی که ارائه می‌دهد، فرآیند ساخت چت‌بات‌ها را ساده‌تر می‌کند.

در ادامه، با معماری و مراحل توسعه‌ی یک چت‌بات با استفاده از Spring AI آشنا خواهیم شد.

مزایای چت‌بات استریمینگ

- کاهش زمان انتظار کاربر: کاربر می‌تواند بخش‌هایی از پاسخ را بلافاصله دریافت کند، بدون اینکه منتظر تکمیل کل پاسخ بماند.
- تعامل بهینه‌تر: نمایش تدریجی پاسخ باعث می‌شود کاربر حس بهتری از ارتباط با سیستم داشته باشد.

- کاهش بار پردازشی سرور: ارسال تدریجی داده‌ها باعث کاهش فشار بر منابع سرور می‌شود، زیرا نیازی به ارسال یک پاسخ حجیم در یک درخواست نیست.
- پشتیبانی از مدل‌های مختلف زبانی: استفاده از Spring AI به ما این امکان را می‌دهد که با حداقل تغییر در کد (صرفا با چند تغییر در pom و application.properties، از مدل‌های مختلفی مانند OpenAI، Anthropic Claude، و Google Gemini بهره ببریم.

معماری مشتری چت با استفاده از Spring AI

معماری یک مشتری چت که با Spring AI توسعه داده می‌شود، معمولا به‌صورت چند لایه‌ای طراحی می‌شود تا مدیریت، نگهداری و توسعه‌ی آن ساده‌تر باشد. این لایه‌ها شامل بخش‌های مختلفی هستند که هر کدام وظایف خاص خود را دارند:

- لایه نمایش (Presentation Layer): این لایه وظیفه‌ی تعامل مستقیم با کاربر را بر عهده دارد. معمولا از فناوری‌های وب مانند HTML، CSS و JavaScript برای توسعه‌ی این بخش استفاده می‌شود. همچنین، فریم‌ورک‌های جاوا اسکریپت مانند Angular، React یا Vue.js می‌توانند برای ایجاد یک رابط کاربری پویا و کاربرپسند به کار گرفته شوند. این لایه اطلاعات را به کاربر نمایش می‌دهد و ورودی‌های او را برای پردازش جمع‌آوری می‌کند.
- لایه سرویس (Service Layer): این لایه مسئول اجرای منطق اصلی برنامه است. در این بخش، درخواست‌هایی که از لایه نمایش دریافت می‌شود، پردازش شده و ارتباطات لازم با لایه داده برقرار می‌شود. همچنین این لایه مسئول ارتباط با مدل‌های هوش مصنوعی برای تحلیل و تولید پاسخ است. Spring Boot یک فریم‌ورک ایده‌آل برای ساخت این لایه محسوب می‌شود، زیرا قابلیت‌هایی مانند مدیریت تراکنش‌ها، امنیت و کنترل درخواست‌ها را به توسعه‌دهنده ارائه می‌دهد.
- لایه داده (Data Layer): این لایه وظیفه‌ی ذخیره و بازیابی داده‌ها را بر عهده دارد. انتخاب نوع پایگاه داده به ساختار داده‌ها و حجم آن‌ها بستگی دارد. پایگاه‌های داده رابطه‌ای مانند MySQL و PostgreSQL برای ذخیره‌ی داده‌های ساختاریافته مناسب هستند، در حالی که پایگاه‌های داده‌ی NoSQL مانند MongoDB و Cassandra گزینه‌های مناسبی برای داده‌های انعطاف‌پذیر و توزیع‌شده محسوب می‌شوند. Spring Data مجموعه‌ای از ابزارهای کارآمد را برای مدیریت پایگاه‌های داده در اختیار توسعه‌دهندگان قرار می‌دهد.
- لایه مدل (Model Layer): این لایه شامل مدل‌های هوش مصنوعی است که وظیفه‌ی پردازش زبان طبیعی (NLP) و تحلیل نیت کاربر را بر عهده دارند. در این بخش از مدل‌های از پیش آموزش‌دیده مانند BERT و GPT یا مدل‌های سفارشی که بر روی مجموعه‌داده‌های خاص آموزش دیده‌اند، استفاده می‌شود. این مدل‌ها می‌توانند ورودی‌های متنی را پردازش کرده و پاسخ‌های مناسب را تولید کنند.

این معماری لایه‌ای به توسعه‌دهندگان کمک می‌کند تا بخش‌های مختلف سیستم را به‌صورت مستقل توسعه داده و بهینه‌سازی کنند. همچنین، به دلیل انعطاف‌پذیری بالای Spring AI، امکان ادغام این ساختار با سرویس‌های ابری و سایر سیستم‌های مبتنی بر هوش مصنوعی به‌راحتی فراهم می‌شود.

مراحل توسعه مشتری چت

توسعه یک مشتری چت در Spring AI شامل مراحل مختلفی است که در ادامه به شرح مختصر هر مرحله می‌پردازیم:

1. طراحی و برنامه‌ریزی: در این مرحله نیازمندی‌های کاربر، اهداف کسب‌وکار و ویژگی‌های مورد انتظار مشتری چت به دقت بررسی و تحلیل می‌شوند. سپس طرح کلی معماری سیستم، مدل داده و رابط کاربری طراحی می‌شود.

2. توسعه لایه مدل: این مرحله شامل انتخاب و آموزش مدل‌های هوش مصنوعی مناسب برای پردازش زبان طبیعی است. در این مرحله ممکن است نیاز به جمع‌آوری و آماده‌سازی داده‌های آموزشی برای آموزش مدل‌های سفارشی باشد.

3. توسعه لایه سرویس: در این مرحله منطق کسب‌وکار، مدیریت درخواست‌ها و پاسخ‌ها، فراخوانی مدل‌های هوش مصنوعی و برقراری ارتباط با لایه داده پیاده‌سازی می‌شود. از Spring Boot و ابزارهای مرتبط برای ساخت این لایه استفاده می‌شود.

4. توسعه لایه نمایش: در این مرحله رابط کاربری مشتری چت با استفاده از فناوری‌های وب و فریم‌ورک‌های جاوا اسکریپت طراحی و پیاده‌سازی می‌شود. این لایه وظیفه ارائه اطلاعات به کاربر و دریافت ورودی از او را بر عهده دارد.

5. آزمون و تست: بعد از اتمام توسعه، سیستم به دقت مورد آزمایش و تست قرار می‌گیرد تا از عملکرد صحیح و بدون خطا بودن آن اطمینان حاصل شود. تست‌های واحد، تست‌های ادغام و تست‌های کارایی از جمله تست‌هایی هستند که باید انجام شوند.

6. استقرار و نگهداری: در این مرحله مشتری چت در محیط مورد نظر استقرار داده می‌شود. پس از استقرار، به منظور رفع اشکالات احتمالی و بهبود عملکرد، نگهداری و پشتیبانی از آن ضروری است.

راه‌اندازی پروژه در Spring Boot

برای شروع، به Spring Initializr مراجعه کرده و تنظیمات اولیه پروژه را انجام می‌دهیم:

- **گروه (Group):** edu.sharif.edu
- **نام پروژه:** streaming
- **زبان:** Java 23
- **نوع پروژه:** Maven
- **وابستگی‌ها:**
    - Spring Web
    - Spring AI

بعد از تکمیل این تنظیمات، فایل ZIP پروژه را دانلود کرده و آن را در محیط توسعه‌ای مانند IntelliJ IDEA باز می‌کنیم.

افزودن وابستگی‌های موردنیاز

Spring AI از مدل‌های مختلف پردازش زبان طبیعی (NLP) پشتیبانی می‌کند. برای استفاده از Anthropic Claude به‌عنوان مدل زبانی، وابستگی زیر را در فایل pom.xml پروژه اضافه می‌کنیم:

```
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-anthropic</artifactId>
    <version>آخرین نسخه</version>
</dependency>
```

پس از اضافه کردن این وابستگی، پروژه را مجددا بارگذاری (reload) می‌کنیم.

پیکربندی کلید API و مدل زبانی

در فایل application.properties تنظیمات مربوط به کلید API و مدل زبانی را انجام می‌دهیم:

```
spring.ai.anthropic.api-key=${ANTHROPIC_API_KEY}
spring.ai.anthropic.model=claude-3.5-sonnet
```

ذخیره‌سازی کلید API در متغیرهای محیطی، یک روش استاندارد برای افزایش امنیت و جلوگیری از افشای اطلاعات حساس است. در صورت نیاز، می‌توان مدل زبانی را به راحتی تغییر داد و از دیگر مدل‌های پشتیبانی‌شده بهره برد.

پیاده‌سازی کنترلر چت‌بات استریمینگ

در این مرحله، یک کنترلر برای پردازش درخواست‌های کاربران و ارسال پاسخ‌های استریمینگ پیاده‌سازی می‌کنیم:

```
@RestController
@RequestMapping("/chat")
public class ChatController {
    private final ChatService chatService;
    public ChatController(ChatService chatService) {
        this.chatService = chatService;
    }
    @GetMapping(value = "/stream", produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
    public Flux<String> streamChat(@RequestParam String query) {
        return chatService.getStreamingResponse(query);
    }
}
```

پیاده‌سازی سرویس پردازش پیام‌ها

سرویس زیر درخواست‌های کاربران را پردازش کرده و پاسخ‌های مدل زبانی را به‌صورت استریم ارسال می‌کند.

```
@Service
public class ChatService {
    private final ChatClient chatClient;
    public ChatService(ChatClient chatClient) {
        this.chatClient = chatClient;
    }
    public Flux<String> getStreamingResponse(String query) {
        return chatClient.streamChatResponse(query);
    }

}
```

ایجاد رابط کاربری برای چت‌بات استریمینگ

برای نمایش پاسخ‌های استریمینگ در یک رابط کاربری کاربرپسند، از JavaScript، HTML و fetch API استفاده می‌کنیم:

```
<!DOCTYPE html>
<html lang="fa">
<head>
    <meta charset="UTF-8">
    <title>چت‌بات استریمینگ</title>
    <script>
        async function fetchStreamWithRetry(url, retries = 3) {
            for (let i = 0; i < retries; i++) {
                try {
                    const response = await fetch(url);
                    if (!response.ok) throw new Error("خطای اتصال به سرور");
                    return response.body;
                } catch (error) {
                    if (i === retries - 1) throw error;
                }
            }
        }

        async function sendMessage() {
            let message = document.getElementById("chat-input").value;
            let chatMessages = document.getElementById("chat-messages");
            chatMessages.innerHTML += `<div>کاربر: ${message}</div>`;

            const responseStream = await
fetchStreamWithRetry(`/chat/stream?query=${message}`);
            const reader = responseStream.getReader();

            function push() {
                reader.read().then(({done, value}) => {
                    if (done) return;
                    chatMessages.innerHTML += `<div>ربات: ${new
TextDecoder().decode(value)}</div>`;
                    push();
                });
            }
            push();
        }
    </script>
</head>
<body>
    <div id="chat-messages" class="p-4 border"></div>
    <input id="chat-input" type="text" class="border p-2" placeholder="پیام خود را وارد
کنید...">
    <button onclick="sendMessage()" class="bg-blue-500 text-white p-2">ارسال</button>
</body>
</html>
```

پیاده‌سازی ما و خروجی آن

```java
@RestController  AHMZ
@CrossOrigin
public class ChatController {
    private final ChatClient chatClient;  3 usages

    public ChatController(ChatClient.Builder builder) {  AHMZ
        this.chatClient = builder.build();
    }

    @PostMapping("/chat")  AHMZ
    public String chat(@RequestParam String message) {
        return chatClient.prompt() ChatClientRequestSpec
                .user(message)
                .call() CallResponseSpec
                .content();
    }

    @GetMapping("/stream")  AHMZ
    public Flux<String> chatWithStream(@RequestParam String message) {
        return chatClient.prompt() ChatClientRequestSpec
                .user(message)
                .stream() StreamResponseSpec
                .content();
    }
}
```

چالش‌ها و فرصت‌ها

توسعه‌ی یک چت‌بات با استفاده از Spring AI می‌تواند هم چالش‌برانگیز باشد و هم فرصت‌های خوبی را در اختیار توسعه‌دهندگان قرار دهد. یکی از مشکلات اصلی، پیچیدگی کار با مدل‌های هوش مصنوعی است. آموزش این مدل‌ها و استفاده از آن‌ها زمان‌بر است و نیاز به دانش فنی بالایی دارد. همچنین، چون چت‌بات‌ها با حجم زیادی از پیام‌ها و داده‌های کاربران سروکار دارند، پردازش این اطلاعات می‌تواند به منابع زیادی نیاز داشته باشد. مسئله‌ی امنیت هم بسیار مهم است، چون اطلاعات کاربران باید به‌خوبی محافظت شود تا از سوءاستفاده‌های احتمالی جلوگیری شود. علاوه بر این، چت‌بات‌ها باید سریع و دقیق پاسخ بدهند تا کاربران تجربه‌ی بهتری داشته باشند. بهینه‌سازی عملکرد سیستم برای رسیدن به این هدف، یکی از دغدغه‌های اصلی توسعه‌دهندگان است.

چالش دیگری که در این مسیر وجود دارد، پردازش حجم بالای داده‌های متنی است. چت‌بات‌ها باید بتوانند در لحظه پاسخ کاربران را پردازش کرده و جواب مناسب را ارائه دهند، اما این کار نیاز به منابع پردازشی قوی دارد. اگر حجم داده‌ها بالا باشد، ممکن است عملکرد سیستم کاهش پیدا کند و زمان پاسخ‌دهی بیشتر شود.

با این حال، استفاده از چت‌بات‌های استریمینگ مزایای زیادی دارد. یکی از مهم‌ترین مزیت‌ها، بهبود تجربه‌ی کاربری است. کاربران دیگر نیازی ندارند که مدت زیادی منتظر پاسخ بمانند، بلکه می‌توانند همزمان با پردازش، بخش‌هایی از پاسخ را مشاهده کنند. این تعامل باعث می‌شود که احساس بهتری نسبت به مکالمه‌ی خود با چت‌بات داشته باشند. همچنین، این فناوری می‌تواند بهره‌وری کسب‌وکارها را افزایش دهد، چون بسیاری از کارهای مربوط به پشتیبانی مشتریان و پاسخ‌گویی به سوالات تکراری به‌صورت خودکار انجام می‌شود. این موضوع باعث کاهش هزینه‌های عملیاتی می‌شود و به شرکت‌ها این امکان را می‌دهد که نیروی انسانی خود را برای انجام وظایف پیچیده‌تر اختصاص دهند.

یکی دیگر از مزایای مهم چت‌بات‌های استریمینگ، امکان نوآوری در خدمات است. این چت‌بات‌ها می‌توانند در حوزه‌هایی مانند پشتیبانی آنلاین، آموزش هوشمند و حتی مشاوره‌ی دیجیتال استفاده شوند. بسیاری از کسب‌وکارها می‌توانند با استفاده از این تکنولوژی، خدمات جدید و خلاقانه‌ای ارائه دهند که تا پیش از این امکان‌پذیر نبود.

نتیجه‌گیری

به‌طور کلی، استفاده از Spring AI برای توسعه‌ی چت‌بات‌های استریمینگ، یک راهکار بهینه و موثر برای ایجاد سیستم‌های مکالمه محور هوشمند است. این فناوری به شرکت‌ها کمک می‌کند تا تعامل بهتری با کاربران داشته باشند، کارایی سیستم‌های خود را افزایش دهند و در عین حال هزینه‌های خود را کاهش دهند. با توجه به این مزایا، انتظار می‌رود که در آینده‌ی نزدیک، کسب‌وکارهای بیشتری به سمت استفاده از چت‌بات‌های هوشمند بروند. البته برای موفقیت در این مسیر، باید چالش‌های موجود را شناسایی کرد و راهکارهای مناسبی برای حل آن‌ها در نظر گرفت.

لینک ویدئوی تحقیق

https://drive.google.com/file/d/1vo6uSsudAYZcvuU1vPmQjDnS5YCX8I1M/view?usp=sharing

منابع

- استفاده از تعاریف و کدهای این سایت (که در شیت قرارداده شده بود)، در پیاده‌سازی:

https://docs.spring.io/spring-ai/reference/api/chatclient.html

- استفاده از تعاریف و راه‌اندازی spring ai chat:

https://youtu.be/q2p0mG4RICM?si=0z_wuBnzvm-8pc6G