گزارش پروژه تست تپسل برای کار با اسپرینگ و ابزارهای پردازش داده حجیم

سینا ممکن

۱۰ اسفند ۱۴۰۳

۱. ابزارهای استفاده شده

اولین کار برای توسعه یک پروژه فراهم آوردن ابزارهای مورد نیاز است. بدین منظور لازم بود که **سیستم‌عامل**، **IDE** و **پایگاه‌داده‌ها** در **محیط توسعه** ایجاد شوند. به منظور انعطاف بیشتر از سیستم‌عامل Ubuntu 24.04 و ابزار توسعه IntelliJ IDEA 2024.1.4 (Ultimate Edition) استفاده شد که از قبل بر روی کامپیوتر نصب بودند.

برای بالا آوردن پایگاه‌داده‌ها ابتدا docker-cli (از طریق فیلترشکن) نصب شد، تصاویر[1] apache/kafka و cassandra:latest (از طریق shecan) از docker hub بارگیری شد، سپس کانتینر[2]های آن‌ها بر روی داکر بالا آمدند و پورت‌های پیش‌فرض پایگاه‌داده‌ها بر روی localhost نگاشت[3] شدند. جهت تست صحت عملکرد کاساندرا از طریق Database Tool در داخل خود Intellij به کاساندرا متصل شده و کوئری‌های موجود در فایل cassandra-scripts.txt بر روی پایگاه‌داده اجرا شدند.

۲. ساختار پروژه

ساختار این پروژه مطابق درخواست از ماژول اصلی consumer و ماژول ساده producer تشکیل شده است؛ البته ساختار پروژه از ابتدا بدین شکل نبود و ابتدا consumer جهت تست اتصال به کافکا و کاساندرا نوشته شد و پس از ریفکتور[4] به دو ماژول میون جداگانه تقسیم شد.

پروژه producer علاوه بر فایل اصلی ProducerApplication.kt از پیکربندی[5]های لازم برای ساخت topic در کافکا و ایجاد یک ProdcuerFactory با serializerهای لازم برخوردار است. رخدادها در پکیج entity.kafka نوشته شده‌اند و فرآیند ایجاد و ارسال رخدادها هر ۱ ثانیه یکبار توسط ProducerScheduler.produce انجام می‌شود.

1 image
2 container
3 map
4 refactor
5 config

نکته خاص این پیاده‌سازی این است که هر دو رخداد ImpressionEvent و ClickEvent تبدیل بـه JSON شده و بر روی **یک events_topic در کافکا** نوشته می‌شوند و از ایجاد صف‌های مختلف خودداری شده است.

پروژه اصلی consumer نـیز علاوه بـر فایـل اصـلی ConsumerApplication.kt از پیکربنـدی‌های لازم بـرای ایجاد ConsumerFactory با deserializerهای لازم و RecordMessageConverter شخصی‌سازی‌شـده برای برگرداندن JSONها به کلاس‌های رخـداد مرتبـط بهـره می‌بـرد. کلاس‌های رخـدادها و موجودیت‌ها در پکیج entity نوشته شده‌اند و ریپازیتوری[۶] کار با کاساندرا در پکیج repository نوشته شده است.

برای نگاشت بین موجودیت‌های مختلف از کتابخانه و ابزار مولد کد[۷] MapStruct استفاده شده تا استفاده از set و getهای مکرر را به کلی حذف کرده و به خوانایی کد کمک کند. نگارنده[۸]ی مورد اسـتفاده در پکیج mapper قرار دارد.

اصـل واکنش بـه رخـدادهای کافکـا در کلاس MultiTypeKafkaListener.kt مـدیریت می‌شـود و بـه دلیـل سادگی، بیزینس مرتبط با هر رخداد نـیز در همین کلاس مـدیریت می‌شـود و از نوشـتن کلاس سرویس[۹] جداگانـه خودداری شده است. از آنجایی که بیزینس در همین کلاس نوشته شده است متدهای آن موجـودیت ذخیره‌شـده را بـر می‌گردانـنـد کـه بعـداً در کلاس MultiTypeKafkaListenerTest از این خروجی‌ها بـرای صحت‌سـنجی عمل‌کرد بیزینس‌ها استفاده شده است. تست‌ها از نوع JUnit هستند و برای تقلید[۱۰] لایه‌ی زیرین یعنی ریپازیتوری از کتابخانه‌ی mockito استفاده شده است. هر تست از ۳ قسمت پیش‌فرض[۱۱]، موقع تست[۱۲] و آنگاه بررسی نتیجـه تست[۱۳] تشکیل شده است که همگی آن‌ها پاس شده‌اند.

برای تست پروژه‌ها و دیـدن خروجی هر ماژول کـافی اسـت متـد main هر دو اپلیکیشـن را همزمان اجـرا کنیـد و خروجی کنسول[۱۴] هر دو اپلیکیشـن را بررسـی نماییـد؛ ایجاد رخـدادها و نوشـتن آن‌ها در کافکـا در کنسـول ماژول producer و دریافت رخـدادها، درج و به‌روزرسانی موجـودیت کاساندرا در کنسـول ماژول consumer قابـل مشاهده است.

6 repository
7 code generator
8 mapper
9 service
10 mock
11 given
12 when
13 then
14 console

۳. زمان انجام پروژه

فراهم آوردن ابزارهای مورد نیاز، تحقیق و جستجو در مورد کار با ابزارها و کتابخانه‌ها، نوشتن پله بـه پلـه ماژول‌ها و اشکال‌یابی[۱۵] و تسـت آن‌ها همگی در ۲-۳ روز (شـامل دو روز نصـفه کـاری و یـک روز تعطیـل) انجام شـد. سابقه[۱۶] تغییرات انجام‌شده در مخزن گیت[۱۷] پروژه به آدرس زیر در دسترس است:

https://github.com/sinamomken/tapsell-kafka-task

۴. منابع استفاده شده

- Google
- ChatGPT
- Baeldung

15 debug
16 history
17 git repository