طراحی میکروسرویس پروژه اجاره مسکن آنلاین SweetHome

سهند نوعی – سپیهر نوعی – امیرحسین صلاحی

طراحی معماری اولیه

برای شروع، سیستم را به صورت یک سیستم تک‌لایه (Monolithic) طراحی می‌کنیم که شامل تمام سرویس‌های مورد نیاز است. این سرویس‌ها می‌توانند شامل موارد زیر باشند:

۱. سرویس مدیریت کاربران: برای ثبت‌نام و احراز هویت کاربران (مسافران و میزبانان).

۲. سرویس رزرو: برای مدیریت رزروها و پرداخت‌ها.

۳. سرویس مدیریت مسکن: برای مدیریت اطلاعات مسکن‌ها توسط میزبانان.

۴. سرویس امتیازدهی و بازخورد: برای امتیازدهی و ارائه نظرات پس از پایان اقامت.

این معماری برای شروع با تعداد کمی کاربر مناسب است و توسعه آن سریع و ساده است.

روند بزرگ کردن مقیاس برنامه

۱. افقی‌سازی(Horizontal Scaling)

- با افزایش تعداد کاربران، اولین قدم افزودن سرورهای بیشتر و تقسیم بار میان آن‌هاست.
- استفاده از Load Balancer برای توزیع ترافیک میان سرورها.

۲. تقسیم به میکروسرویس‌ها:

- سیستم خود را به میکروسرویس‌های کوچکتر تقسیم می‌کنیم. هر میکروسرویس باید مسئولیت خاصی داشته باشد.
- مثلاً سرویس‌های مدیریت کاربران، رزرو، مدیریت مسکن و امتیازدهی را به میکروسرویس‌های جداگانه تبدیل کنید.

- استفاده از API Gateway برای مدیریت درخواست‌ها و توزیع آن‌ها به میکروسرویس‌های مناسب.

۳. استفاده از پایگاه‌داده‌های توزیعی:

- از پایگاه‌داده‌های توزیعی مانند Cassandra، MongoDB یا Amazon DynamoDB استفاده می‌کنیم تا بتوانیم داده‌ها را در چندین سرور ذخیره و مدیریت کنیم.

- از تکنیک Replication برای افزایش مقیاس‌پذیری و دسترس‌پذیری استفاده کنید.

۴. پیاده‌سازی سیستم‌های صف و پیام‌رسانی:

- از سیستم‌های صف مانند Kafka یا RabbitMQ برای مدیریت بارهای سنگین و ارتباط بین میکروسرویس‌ها استفاده می‌کنیم.

۵. استفاده از معماری Cloud-Native:

- از سرویس‌های ابری مانند Amazon Web Services (AWS)، Google Cloud Platform (GCP) یا Microsoft Azure برای میزبانی و مدیریت میکروسرویس‌ها استفاده می‌کنیم.

- از ابزارهای مدیریت کانتینر مانند Kubernetes برای مدیریت کانتینرها و میکروسرویس‌ها استفده می‌کنیم.

۶. پیاده‌سازی روش‌های مانیتورینگ و Logging:

- از ابزارهای مانیتورینگ مانند Prometheus، Grafana و ELK Stack برای مانیتورینگ و تحلیل لاگ‌ها استفاده می‌کنیم.

- تعریف alertها برای شناسایی مشکلات و رخدادهای غیرمنتظره و واکنش سریع به آن‌ها.

این مراحل کمک می‌کند تا به مرور زمان سیستم خود را با افزایش تعداد کاربران و حجم داده‌ها به مقیاس بزرگتری تبدیل کنیم.

تقسیم سیستم به میکروسرویس‌ها

با توجه به نیازهای سامانه اجاره مسکن و رشد پیش‌بینی‌شده، سیستم را به پنج میکروسرویس زیر تقسیم می‌کنیم:

۱. سرویس مدیریت کاربران (User Management Service):

- وظایف: این سرویس مسئول ثبت‌نام، احراز هویت و مدیریت اطلاعات کاربران (مسافران و میزبانان) است.

- دامنه: شامل ثبت‌نام کاربران جدید، احراز هویت، بازیابی رمز عبور و مدیریت پروفایل کاربران.

۲. سرویس مدیریت مسکن (Property Management Service):

- وظایف: این سرویس مسئول ثبت و مدیریت اطلاعات مسکن‌ها توسط میزبانان است.

- دامنه: شامل اضافه کردن مسکن جدید، ویرایش اطلاعات مسکن‌ها، مدیریت تصاویر و تعیین قیمت‌ها.

۳. سرویس رزرو و پرداخت (Booking and Payment Service):

- وظایف: این سرویس مسئول مدیریت فرآیندهای رزرو و پرداخت است.

- دامنه: شامل جستجو و رزرو مسکن، مدیریت تقویم‌های رزرو، محاسبه هزینه‌ها و مدیریت پرداخت‌ها.

۴. سرویس امتیازدهی و بازخورد (Rating and Feedback Service):

- وظایف: این سرویس مسئول جمع‌آوری و مدیریت امتیازات و نظرات کاربران است.

- دامنه: شامل امکان امتیازدهی به مسکن‌ها و میزبانان، ثبت نظرات کاربران و نمایش بازخوردها به سایر کاربران.

۵. سرویس پیشنهادات و توصیه‌ها (Recommendation Service):

- وظایف: این سرویس مسئول ارائه پیشنهادات شخصی‌سازی‌شده به کاربران بر اساس تاریخچه رزروها و ترجیحات آنها است.

- دامنه: شامل تحلیل داده‌های کاربران، ایجاد الگوریتم‌های پیشنهاددهی و نمایش پیشنهادات مرتبط به کاربران.

دلایل انتخاب میکروسرویس‌ها

این تقسیم‌بندی به ما کمک می‌کند تا بتوانیم به صورت مستقل هر سرویس را توسعه داده، تست کنیم و دیپلوی کنیم. همچنین، با این ساختار می‌توانیم به راحتی هر میکروسرویس را به صورت مجزا مقیاس‌پذیر کنیم و در صورت نیاز منابع بیشتری به آن اختصاص دهیم.

با این تقسیم‌بندی، هر تیم توسعه می‌تواند بر روی یک میکروسرویس خاص متمرکز شود و کارایی و بهره‌وری بالاتری داشته باشد. همچنین، از تداخل‌های ناخواسته بین بخش‌های مختلف سیستم جلوگیری می‌شود.

این پاسخ‌ها را در یک فایل جداگانه قرار داده و در فایل نهایی پروژه اضافه می‌کنیم تا تمامی جزئیات مرتبط با تقسیم سیستم به میکروسرویس‌ها به صورت کامل و دقیق مستند شوند.

انتخاب رابط پیام برای سیستم **(امتیازی)**

برای سیستم خود، رابط پیام Kafka را انتخاب می‌کنیم. دلایل انتخاب Kafka به شرح زیر است:

۱.قابلیت مقیاس‌پذیری بالا: Kafka قادر است حجم زیادی از داده‌ها را به صورت همزمان مدیریت کند و این امکان را به ما می‌دهد که با افزایش تعداد کاربران، سیستم به‌راحتی مقیاس‌پذیر باشد.

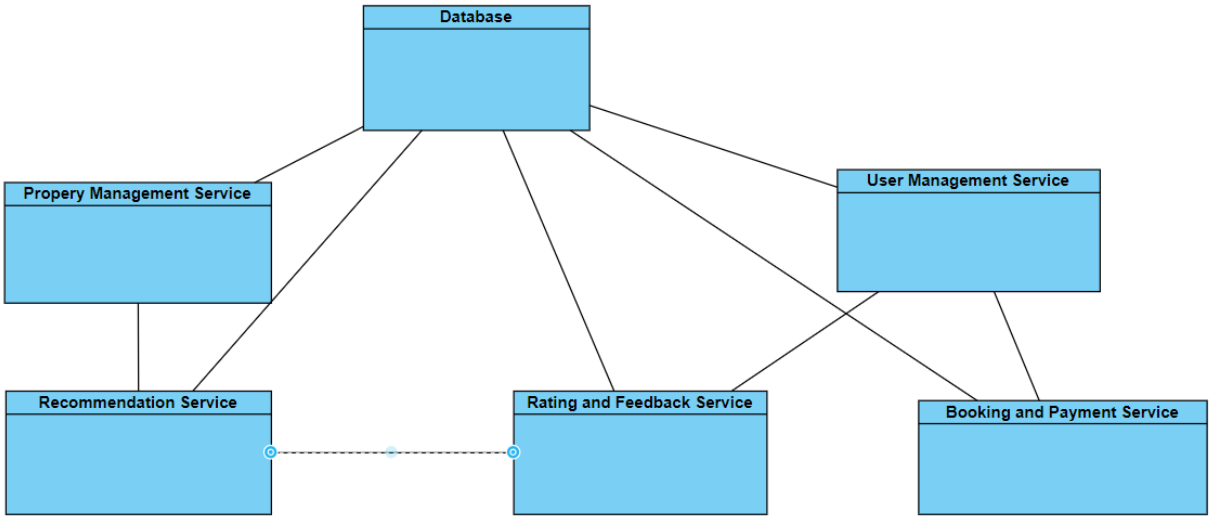
۲. پایداری و تحمل خطا: Kafka به گونه‌ای طراحی شده است که تحمل خطا و پایداری بالایی داشته باشد. این ویژگی برای یک سیستم بزرگ و حیاتی بسیار مهم است.

۳. تاخیر پایین: Kafka تاخیر کمی در ارسال و دریافت پیام‌ها دارد که برای بسیاری از عملیات‌های بلادرنگ ضروری است.

۴. انعطاف‌پذیری در مصرف‌کنندگان: Kafka به ما این امکان را می‌دهد که مصرف‌کنندگان متعددی به صورت مستقل و موازی از پیام‌ها استفاده کنند. این ویژگی برای میکروسرویس‌های مختلف که نیاز به دسترسی به داده‌های مشابه دارند، بسیار مفید است.

۵. پشتیبانی از پردازش جریان: Kafka با ابزارهایی مانند Kafka Streams و KSQL به ما اجازه می‌دهد پردازش جریان داده‌ها را به راحتی انجام دهیم.

در ادامه، نمودار ارجاع سیستم میکروسرویس‌های سامانه اجاره مسکن را ترسیم می‌کنیم. این نمودار نشان‌دهنده روابط و وابستگی‌های بین میکروسرویس‌ها و همچنین نحوه ارتباط آنها از طریق Apache Kafka است.



توضیحات نمودار:

- User Management Service: ارتباط مستقیم با پایگاه داده کاربران و Kafka برای ارسال و دریافت پیام‌های احراز هویت و مدیریت پروفایل.

- Property Management Service: مدیریت اطلاعات مسکن‌ها و ارتباط با Kafka برای اطلاع‌رسانی تغییرات به سایر سرویس‌ها.

- Booking and Payment Service: پردازش رزروها و پرداخت‌ها و ارتباط با Kafka برای مدیریت تراکنش‌ها و پیام‌های مرتبط.

- Rating and Feedback Service: جمع‌آوری و مدیریت امتیازات و نظرات کاربران و ارتباط با Kafka برای اشتراک نظرات و امتیازات.

- Recommendation Service: تحلیل داده‌های کاربران و ارائه پیشنهادات و ارتباط با Kafka برای دریافت داده‌های تحلیل و ارائه پیشنهادات.

سند معماری میکروسرویس مدیریت کاربران(User Management Service)

۱. مروری کلی

میکروسرویس کاربران مسئولیت مدیریت اطلاعات کاربران، ثبت‌نام، ورود، احراز هویت و پروفایل کاربری را بر عهده دارد. این میکروسرویس باید امنیت بالا، مقیاس‌پذیری و پایداری را تضمین کند.

۲. معماری

زیرساخت و فناوری‌ها:

- زبان برنامه‌نویسی: Spring Boot (Java)
- پایگاه داده: PostgreSQL برای ذخیره اطلاعات کاربران
- احراز هویت: JSON Web Tokens (JWT) برای مدیریت جلسات کاربران
- Cache: Redis برای ذخیره موقت داده‌های کاربر و توکن‌های احراز هویت
- ارتباط بین سرویس‌ها: gRPC یا REST برای ارتباط با سایر میکروسرویس‌ها

ساختار معماری:

- Controllers:
  - مدیریت درخواست‌های HTTP از کاربران.
  - Endpointها برای ثبت نام، ورود، ویرایش پروفایل و بازیابی رمز عبور.

- Services:
  - منطق کسب‌وکار اصلی شامل مدیریت کاربران، احراز هویت و صدور توکن‌های JWT.

- Repositories:
  - ارتباط با پایگاه داده و اجرای عملیات CRUD (Create, Read, Update, Delete).

- Middleware:

- فیلترهای امنیتی برای احراز هویت و مجوزدهی کاربران.

۳. جریان داده‌ها

۱. ثبت نام: کاربر با ارسال اطلاعات خود (مانند ایمیل و رمز عبور) به Endpoint ثبت نام، حساب کاربری جدید ایجاد می‌کند.

۲.ورود: کاربر با ارسال اطلاعات ورود خود به Endpoint ورود، پس از احراز هویت، توکن JWT دریافت می‌کند.

۳.ویرایش پروفایل: کاربر با ارسال درخواست به Endpoint مربوطه، می‌تواند اطلاعات پروفایل خود را ویرایش کند.

۴. احراز هویت: توکن JWT در هر درخواست ارسال می‌شود و میکروسرویس با بررسی اعتبار آن، درخواست کاربر را احراز هویت می‌کند.

سند معماری میکروسرویس مدیریت مسکن‌ها (Property Management Servic)

۱. مروری کلی

میکروسرویس مسکن‌ها وظیفه مدیریت اطلاعات مسکن‌ها، افزودن، ویرایش، حذف و جستجوی مسکن‌ها را بر عهده دارد. این میکروسرویس باید قابلیت مدیریت حجم زیادی از داده‌ها و جستجوی بهینه را فراهم کند.

۲. معماری

زیرساخت و فناوری‌ها:

- زبان برنامه‌نویسی: Node.js (Express)

- پایگاه داده: MongoDB برای ذخیره اطلاعات مسکن‌ها به صورت داکیومنتی

- Cache: Redis برای ذخیره نتایج جستجوی پرتکرار

- ارتباط بین سرویس‌ها: REST یا GraphQL برای ارتباط با سایر میکروسرویس‌ها

ساختار معماری:

- Controllers:

- مدیریت درخواست‌های HTTP از کاربران.

- Endpointها برای افزودن، ویرایش، حذف و جستجوی مسکن‌ها.

- Services:

- منطق کسب‌وکار اصلی شامل مدیریت اطلاعات مسکن‌ها و عملیات جستجو.

- Repositories:

- ارتباط با پایگاه داده MongoDB و اجرای عملیات .CRUD

- Middleware:

- فیلترهای امنیتی برای اطمینان از مجوز دسترسی به عملیات مدیریتی.

۳. جریان داده‌ها

۱. افزودن مسکن: میزبان با ارسال اطلاعات مسکن به Endpoint مربوطه، یک مسکن جدید را به سیستم اضافه می‌کند.

۲. ویرایش مسکن: میزبان می‌تواند با ارسال درخواست به Endpoint مربوطه، اطلاعات مسکن خود را ویرایش کند.

۳. حذف مسکن: میزبان می‌تواند با ارسال درخواست به Endpoint مربوطه، مسکن خود را از سیستم حذف کند.

۴. جستجوی مسکن: کاربران با ارسال درخواست جستجو به Endpoint مربوطه، می‌توانند مسکن‌های موجود را براساس فیلترهای مختلف جستجو کنند. نتایج جستجو از کش Redis بازیابی می‌شود.

سند معماری میکروسرویس رزروها (Booking Service)

۱. مروری کلی

میکروسرویس رزروها مسئولیت مدیریت فرآیندهای رزرو و اجاره مسکن‌ها را بر عهده دارد. این میکروسرویس باید قابلیت هماهنگی بالا، مدیریت وضعیت رزروها و جلوگیری از رزروهای همزمان را فراهم کند.

۲. معماری

زیرساخت و فناوری‌ها:

- زبان برنامه‌نویسی: Spring Boot (Java)
- پایگاه داده: MySQL برای ذخیره اطلاعات رزروها
- Cache: Redis برای مدیریت موقت وضعیت رزروها
- ارتباط بین سرویس‌ها: gRPC یا REST برای ارتباط با سایر میکروسرویس‌ها

ساختار معماری:

- Controllers:
  - مدیریت درخواست‌های HTTP از کاربران.
  - Endpointها برای ایجاد، تأیید، لغو و مدیریت رزروها.

- Services:
  - منطق کسب‌وکار اصلی شامل ایجاد، تأیید، لغو و مدیریت وضعیت رزروها.

- Repositories:

- ارتباط با پایگاه داده MySQL و اجرای عملیات CRUD.

- Middleware:

- فیلترهای امنیتی برای اطمینان از مجوز دسترسی به عملیات مدیریتی.

۳. جریان داده‌ها

۱. ایجاد رزرو: کاربر با ارسال درخواست به Endpoint ایجاد رزرو، یک رزرو جدید ایجاد می‌کند. وضعیت رزرو ابتدا به صورت "در انتظار تأیید" ذخیره می‌شود.

۲. تأیید رزرو: میزبان با بررسی درخواست رزرو، می‌تواند آن را تأیید یا رد کند. وضعیت رزرو به "تأیید شده" یا "رد شده" تغییر می‌کند.

۳. لغو رزرو: کاربر یا میزبان می‌توانند با ارسال درخواست به Endpoint مربوطه، رزرو را لغو کنند. وضعیت رزرو به "لغو شده" تغییر می‌کند.

۴. مدیریت وضعیت رزرو: هر تغییر در وضعیت رزرو در پایگاه داده و کش Redis ذخیره می‌شود تا از رزروهای همزمان جلوگیری شود.