مستند نهایی معماری – یکپارچه‌سازی فروشگاه آنلاین با سیستم حسابداری محک

---

1. C1 – System Context

توضیح کوتاه:

این لایه بازیگران خارجی و سیستم اصلی را نشان می‌دهد. کاربران (مشتری‌ها) و مدیر سیستم با فروشگاه آنلاین تعامل دارند. درگاه پرداخت برای تراکنش‌ها، سرویس SMS برای اعلان‌ها، و سیستم حسابداری محک برای همگام‌سازی مالی حضور دارد. این نمودار دیدی سطح بالا از اکوسیستم کلی سیستم ارائه می‌دهد و مشخص می‌کند که فروشگاه آنلاین ما با چه سیستم‌ها و موجودیت‌های خارجی در ارتباط است.

بازیگران خارجی (External Actors):

- مشتریان (Customers): کاربران نهایی که از فروشگاه آنلاین بازدید کرده، محصولات را مشاهده، انتخاب و خریداری می‌کنند. تعاملات اصلی آن‌ها شامل مرور محصولات، افزودن به سبد خرید، ثبت سفارش و پرداخت است.
- مدیر سیستم (System Administrator): مسئول مدیریت فروشگاه آنلاین، شامل مدیریت محصولات، سفارشات، کاربران، تنظیمات و نظارت بر عملکرد سیستم.
- درگاه پرداخت (Payment Gateway): سرویس خارجی که مسئولیت پردازش امن تراکنش‌های مالی و دریافت وجه از مشتریان را بر عهده دارد.
- سرویس SMS (SMS Service): سرویس خارجی برای ارسال اعلان‌های مربوط به سفارش، پرداخت، ارسال و سایر رویدادهای مهم به مشتریان و مدیران.
- سیستم حسابداری محک (Mahak Accounting System): سیستم مالی و حسابداری خارجی که وظیفه ثبت و مدیریت تراکنش‌های مالی، فاکتورها، انبارداری و سایر امور حسابداری را بر عهده دارد. این سیستم منبع اصلی داده‌های مالی و حسابداری است که نیاز به همگام‌سازی با فروشگاه آنلاین دارد.

سیستم اصلی (Main System):

- فروشگاه آنلاین (Online Store): سیستم اصلی که شامل تمام بخش‌های مورد نیاز برای عملیات یک فروشگاه اینترنتی، از جمله UI/UX برای مشتریان و مدیران، مدیریت

محصولات، مدیریت سفارشات، سبد خرید، سیستم پرداخت، مدیریت کاربران و منطق تجاری است.

ارتباطات (Interactions):

- مشتریان با رابط کاربری فروشگاه آنلاین تعامل دارند.
- مدیران سیستم نیز از طریق رابط کاربری مدیریتی با فروشگاه آنلاین تعامل دارند.
- فروشگاه آنلاین با درگاه پرداخت برای انجام عملیات پرداخت ارتباط برقرار می‌کند.
- فروشگاه آنلاین با سرویس SMS برای ارسال اعلان‌ها ارتباط برقرار می‌کند.
- فروشگاه آنلاین با سیستم حسابداری محک برای همگام‌سازی داده‌های مالی و عملیاتی (مانند ثبت فاکتورها، به‌روزرسانی وضعیت سفارش، مدیریت موجودی) ارتباط برقرار می‌کند. این ارتباط بخش کلیدی پروژه حاضر است.

تصویر:
Placeholder
(این تصویر یک نمای کلی از تعاملات سیستم اصلی با بازیگران خارجی را نمایش می‌دهد. در این تصویر، "Online Store" به عنوان سیستم مرکزی قرار گرفته و فلش‌ها نشان‌دهنده جریان ارتباط با "SMS Service", "Payment Gateway", "Administrator", "Customers", و "Mahak Accounting System" هستند.)

---

2. C2 – Container Diagram

توضیح کوتاه:

نمایش تمام سرویس‌ها/کانتینرهای اصلی معماری شامل PostgreSQL، API Gateway/BFF، Object Storage (MinIO/S3)، Mahak API Adapter، Mahak Sync Worker، RabbitMQ و Frontend. ارتباط و جریان داده بین این اجزا نمایش داده شده است. این نمودار معماری سیستم را در سطح کانتینرها (معمولاً Deployable Units مانند میکروسرویس‌ها یا برنامه‌های مستقل) نشان می‌دهد.

کانتینرها (Containers):

- Frontend: رابط کاربری فروشگاه آنلاین که برای مشتریان و مدیران قابل دسترسی است. این کانتینر مسئول نمایش اطلاعات، دریافت ورودی کاربر و ارسال درخواست‌ها به Backend از طریق API Gateway است.
- API Gateway / BFF (Backend for Frontend): نقطه ورودی واحد برای تمامی درخواست‌های Frontend. این کانتینر وظایف مسیریابی درخواست‌ها به میکروسرویس‌های مناسب، احراز هویت (Authentication) و مجوزدهی

(Authorization)، و همچنین تجمیع یا تبدیل داده‌ها برای Frontend را بر عهده دارد. وجود BFF تضمین می‌کند که هر رابط کاربری (مثلاً وب، موبایل) یک API بهینه شده مختص خود را دریافت کند.

- Online Store Core Service (مثال): یک سرویس اصلی که منطق تجاری هسته فروشگاه آنلاین را پیاده‌سازی می‌کند (مدیریت محصولات، سبد خرید، سفارشات، کاربران). این سرویس معمولاً با دیتابیس اصلی (PostgreSQL) تعامل دارد.
- Mahak Sync Worker: سرویس تخصصی مسئول دریافت داده از سیستم Mahak، پردازش آن، و اعمال تغییرات در سیستم فروشگاه آنلاین یا بالعکس. این کانتینر با Mahak API Adapter برای ارتباط با Mahak و با PostgreSQL برای ذخیره داده‌های همگام‌سازی شده تعامل دارد.
- Mahak API Adapter: کانتینری که وظیفه واسطه‌گری و تبدیل درخواست‌ها و پاسخ‌های بین معماری داخلی ما و APIهای خارجی سیستم Mahak را بر عهده دارد. این کانتینر مشکلات مربوط به پروتکل‌ها، فرمت داده‌ها و مدیریت خطا در سطح API سیستم Mahak را مدیریت می‌کند.
- PostgreSQL (Database): پایگاه داده رابطه‌ای اصلی که داده‌های فروشگاه آنلاین (محصولات، مشتریان، سفارشات، تاریخچه تراکنش‌ها و غیره) را ذخیره می‌کند.
- RabbitMQ (Message Broker): سیستم صف پیام که برای ارتباط ناهمگام بین سرویس‌ها استفاده می‌شود. این سیستم برای اطمینان از تحویل پیام‌ها، پیاده‌سازی الگوهای Pub/Sub، مدیریت صف‌های Retry و Dead-Letter Queues (DLQ) برای پردازش خطا حیاتی است.
- Object Storage (MinIO/S3 Compatible): سرویس ذخیره‌سازی اشیاء که برای نگهداری فایل‌های بزرگ مانند تصاویر محصولات، مستندات، یا لاگ‌ها استفاده می‌شود.

ارتباطات (Interactions):

- Frontend با API Gateway ارتباط برقرار می‌کند.
- API Gateway درخواست‌ها را به سرویس‌های Backend مربوطه (مانند Online Store Core Service) مسیریابی می‌کند.
- سرویس‌های Backend با PostgreSQL برای ذخیره و بازیابی داده‌ها تعامل دارند.
- Mahak Sync Worker از طریق Mahak API Adapter با سیستم Mahak ارتباط برقرار می‌کند.
- Mahak Sync Worker پیام‌ها را از RabbitMQ دریافت کرده و به پردازش ادامه می‌دهد.
- Mahak Sync Worker نتایج پردازش خود را در PostgreSQL ذخیره می‌کند.
- RabbitMQ بین Mahak Sync Worker و سایر سرویس‌های احتمالی (برای ارسال رویدادها یا صف‌بندی وظایف) ارتباط برقرار می‌کند.
- Object Storage برای ذخیره فایل‌های رسانه‌ای مورد استفاده قرار می‌گیرد.

تصویر:
Placeholder

(این تصویر ساختار کانتینرها و ارتباطات اصلی بین آن‌ها را نشان می‌دهد. کانتینرها به صورت بلوک‌های مجزا نمایش داده شده‌اند و خطوط ارتباطی جریان داده و فراخوانی‌ها را مشخص می‌کنند.)

C3 – Component Diagram (Mahak Sync .3 (Worker

توضیح کوتاه:

جزئیات داخلی Mahak Sync Worker شامل ماژول‌های Bulk ،RowVersion Tracker Fetcher/Saver (با Paging)، Logical Delete ،(Conflict Resolution) Upsert Engine Handler، Retry & Error Manager (با DLQ)، و Data Transformer. این نمودار به بررسی اجزای داخلی یک کانتینر کلیدی (Mahak Sync Worker) می‌پردازد.

اجزای داخلی (Mahak Sync Worker (Internal Components:

- Data Transformer: مسئول تبدیل فرمت داده‌های دریافتی از Mahak API به فرمت قابل فهم و استفاده برای سیستم داخلی ما و بالعکس. این شامل نگاشت فیلدها، تبدیل انواع داده و استانداردسازی است.
- Bulk Fetcher/Saver (with Paging): ماژولی که مسئولیت استخراج دسته‌ای (Bulk) داده‌ها از سیستم Mahak از طریق API Adapter و ذخیره دسته‌ای آن‌ها در دیتابیس محلی را بر عهده دارد. استفاده از Paging (صفحه‌بندی) برای مدیریت حجم بالای داده‌ها و جلوگیری از Memory Overload ضروری است.
- RowVersion Tracker: مکانیزمی برای پیگیری نسخه‌های داده‌ها (RowVersion یا Timestamp) که از Mahak دریافت می‌شود. این به ما امکان می‌دهد تا تشخیص دهیم کدام رکوردها تغییر کرده‌اند و نیاز به به‌روزرسانی دارند، و از بارگذاری مجدد داده‌های بدون تغییر جلوگیری می‌کند.
- Upsert Engine (Conflict Resolution): قلب منطق همگام‌سازی. این موتور مسئول انجام عملیات "Upsert" (Update or Insert) در دیتابیس محلی بر اساس داده‌های دریافتی است. همچنین منطق حل تعارض (Conflict Resolution) را پیاده‌سازی می‌کند؛ مثلاً در صورت بروز تداخل بین داده‌های جدید و داده‌های موجود، تصمیم می‌گیرد که کدام نسخه اولویت دارد.
- Logical Delete Handler: مسئولیت مدیریت حذف منطقی (Soft Delete) داده‌ها. به جای حذف فیزیکی رکوردها، این ماژول یک فلگ "IsDeleted" را در رکورد مربوطه تنظیم می‌کند تا نشان دهد داده‌ها دیگر فعال نیستند اما تاریخچه آن‌ها حفظ شود. این برای حفظ یکپارچگی داده‌ها و امکان بازیابی ضروری است.

- Retry & Error Manager (with DLQ): بخشی حیاتی برای مدیریت خطاها و پایداری سیستم. این ماژول وظیفه دارد تا عملیات ناموفق را مجدداً امتحان کند (Retry) با استفاده از مکانیزم‌های صف‌بندی. در صورت شکست مکرر یک عملیات، آن را به یک صف Dead-Letter Queue (DLQ) منتقل می‌کند تا توسط تیم فنی بررسی و رسیدگی شود. این از دست رفتن داده‌ها و توقف فرآیند همگام‌سازی جلوگیری می‌کند.
- Scheduler Integration: ماژولی که با سیستم زمان‌بندی (Scheduler) هماهنگ می‌شود تا فرآیند همگام‌سازی را در فواصل زمانی مشخص اجرا کند.

جریان داده (Data Flow):

1. Scheduler، Mahak Sync Worker را برای شروع فرآیند همگام‌سازی فراخوانی می‌کند.
2. Bulk Fetcher/Saver با استفاده از RowVersion Tracker، داده‌های جدید یا تغییر یافته را از Mahak API (از طریق Mahak API Adapter) واکشی می‌کند.
3. Data Transformer داده‌ها را به فرمت داخلی تبدیل می‌کند.
4. Upsert Engine داده‌های تبدیل شده را دریافت کرده و با استفاده از منطق حل تعارض، عملیات Upsert را در دیتابیس انجام می‌دهد.
5. Logical Delete Handler داده‌های حذف شده منطقی را مدیریت می‌کند.
6. در صورت بروز هرگونه خطا در طول فرآیند، Retry & Error Manager وارد عمل شده و فرآیند را مجدداً امتحان می‌کند یا به DLQ منتقل می‌کند.

تصویر:
Placeholder
(این تصویر ساختار داخلی کانتینر Mahak Sync Worker را با اجزای مختلف و نحوه تعامل آن‌ها با یکدیگر نشان می‌دهد. فلش‌ها جریان پردازش و داده را در داخل این کانتینر مشخص می‌کنند.)

---

4. C4 – Code / Implementation Details

توضیح کوتاه:

تکنولوژی‌ها و ابزار استفاده شده شامل .NET 8 یا JWT، RabbitMQ، PostgreSQL 15، Go، Kubernetes، MinIO/S3، Auth برای اسکیل، و استفاده از معماری Hybrid. این نمودار جزئیات فنی و پشته فناوری (Tech Stack) مورد استفاده در کل معماری را مشخص می‌کند.

تکنولوژی‌ها و ابزار (Technologies & Tools):

- Backend Language/Framework:
  - .NET 8 / Go: هر دو زبان گزینه‌های قدرتمندی برای توسعه میکروسرویس‌ها هستند.
    - .NET 8: ارائه دهنده فریم‌ورک غنی (ASP.NET Core)، اکوسیستم قوی، عملکرد بالا، و پشتیبانی عالی از توسعه Cloud-Native. مناسب برای توسعه Backend Services، API Gateway و Mahak Sync Worker.
    - Go: زبان سبک، با کارایی بالا، مناسب برای توسعه سرویس‌های باConcurrency بالا و مقیاس‌پذیری. می‌تواند جایگزین یا مکمل .NET باشد.
- Database:
  - PostgreSQL 15: پایگاه داده رابطه‌ای قدرتمند، قابل اعتماد، متن‌باز و با پشتیبانی گسترده از ویژگی‌های پیشرفته مانند JSONB، Partitioning و Replication. انتخاب ایده‌آل برای داده‌های ساختاریافته و نیمه‌ساختاریافته.
- Message Broker:
  - RabbitMQ: یک Message Broker قابل اعتماد و با قابلیت‌های فراوان که از الگوهای مختلف پیام‌رسانی (مانند Queues، Pub/Sub) پشتیبانی می‌کند. برای اجرای ناهمگام‌سازی، مدیریت Retry و DLQ بسیار مناسب است.
- API Gateway / Authentication:
  - JWT Auth (JSON Web Token): استاندارد امن برای انتقال اطلاعات بین طرفین به صورت امن به عنوان یک شیء JSON. برای احراز هویت و مجوزدهی در API Gateway و سرویس‌ها استفاده می‌شود.
- Object Storage:
  - MinIO / S3 Compatible: MinIO یک Object Storage سرور متن‌باز با کارایی بالا است که با API آمازون S3 سازگار است. امکان ذخیره‌سازی تصاویر محصولات، فایل‌های لاگ و سایر داده‌های غیرساختاریافته را فراهم می‌کند.
- Orchestration & Deployment:
  - Kubernetes: پلتفرم متن‌باز برای اتوماسیون استقرار (Deployment)، مقیاس‌دهی (Scaling) و مدیریت برنامه‌های کاربردی کانتینری. برای اطمینان از در دسترس بودن بالا (High Availability)، Self-healing و مقیاس‌پذیری خودکار سرویس‌ها حیاتی است.
- Architecture Style:
  - Hybrid Architecture: ترکیبی از رویکردهای مختلف، که در اینجا احتمالاً به معنای استفاده از معماری میکروسرویس برای بخش‌های اصلی و منطق تجاری، و استفاده از یکپارچه‌سازی ناهمگام (Async Integration) با سیستم Mahak از طریق Message Broker است. این رویکرد انعطاف‌پذیری و مقیاس‌پذیری را فراهم می‌کند.

دلایل انتخاب (Rationale):

- قابلیت مقیاس‌پذیری (Scalability): Kubernetes امکان مقیاس‌دهی افقی و عمودی سرویس‌ها را فراهم می‌کند. .NET Core و Go نیز برای توسعه سرویس‌های مقیاس‌پذیر بهینه شده‌اند.
- قابلیت اطمینان (Reliability): RabbitMQ با ویژگی‌های Retry و DLQ، و Kubernetes با مکانیزم‌های Self-healing و Rolling Updates، به افزایش قابلیت اطمینان سیستم کمک می‌کنند.
- انعطاف‌پذیری (Flexibility): معماری میکروسرویس امکان توسعه، استقرار و مقیاس‌دهی مستقل هر سرویس را می‌دهد. انتخاب زبان‌های مدرن مانند .NET 8 یا Go نیز این انعطاف‌پذیری را افزایش می‌دهد.
- مدیریت داده (Data Management): PostgreSQL یک دیتابیس قوی برای مدیریت داده‌های فروشگاه آنلاین و Mahak Sync Worker است. MinIO/S3 راهکاری استاندارد برای ذخیره‌سازی فایل است.
- امنیت (Security): JWT Auth راهی امن و استاندارد برای مدیریت احراز هویت است.

تصویر:
Placeholder
(این تصویر نمایانگر پشته فناوری مورد استفاده در معماری کلی پروژه است. اجزای مختلف مانند پایگاه داده، Message Broker، زبان‌های برنامه‌نویسی و ابزارهای استقرار نمایش داده شده‌اند.)

---

5. Sequence Diagram – BulkSync Simple Flow

توضیح کوتاه:

نسخه ساده‌شده جریان همگام‌سازی داده‌ها برای ارائه به تیم غیر فنی. شامل فازهای درخواست داده از Mahak API، دریافت پاسخ، و ذخیره‌سازی در دیتابیس محلی. این نمودار به زبان ساده نحوه کارکرد همگام‌سازی دسته‌ای را توضیح می‌دهد.

فازهای اصلی (Key Phases):

1. درخواست داده از Mahak API (Request Data from Mahak API):
    - سیستم فروشگاه آنلاین (یا یک Scheduler درون آن) تشخیص می‌دهد که نیاز به همگام‌سازی داده‌ها با سیستم Mahak وجود دارد.
    - درخواستی به Mahak API (از طریق Mahak API Adapter) ارسال می‌شود تا مجموعه‌ای از داده‌های مورد نیاز (مثلاً لیست محصولات یا فاکتورهای جدید)

واکشی شود. این درخواست شامل پارامترهایی برای تعیین بازه زمانی یا فیلترهای دیگر است.

2. دریافت پاسخ (Receive Response):

- Mahak API پاسخ درخواست را ارسال می‌کند. این پاسخ شامل داده‌های درخواستی است که معمولاً در قالبی استاندارد (مانند JSON یا XML) ارائه می‌شود.
- ممکن است پاسخ در چندین بخش (Paging) دریافت شود اگر حجم داده‌ها زیاد باشد.

3. پردازش و ذخیره‌سازی در دیتابیس محلی (Process and Store in Local Database):

- داده‌های دریافتی ابتدا در صورت نیاز پردازش یا تبدیل می‌شوند (مثلاً فرمت‌دهی مجدد).
- سپس، این داده‌ها در پایگاه داده محلی فروشگاه آنلاین ذخیره می‌شوند. بسته به ماهیت داده‌ها، این عملیات می‌تواند "Insert" (درج داده‌های جدید) یا "Update" (به‌روزرسانی داده‌های موجود) باشد.

شرکت‌کنندگان (Participants):

- Scheduler/Online Store: نهادی که درخواست همگام‌سازی را آغاز می‌کند.
- Mahak API Adapter: واسط ارتباط با Mahak API.
- Mahak API: سیستم حسابداری محک.
- Local Database: پایگاه داده فروشگاه آنلاین.

جریان ساده (Simple Flow):

1. `Scheduler/Online Store` -> `Mahak API Adapter` : درخواست واکشی داده‌ها (Fetch Data Request)
2. `Mahak API Adapter` -> `Mahak API` : ارسال درخواست API (Send API Request)
3. `Mahak API` -> `Mahak API Adapter` : پاسخ حاوی داده‌ها (Response with Data)
4. `Mahak API Adapter` -> `Scheduler/Online Store` (یا Mahak Sync Worker): ارسال داده‌های واکشی شده (Forward Fetched Data)
5. `Scheduler/Online Store` -> `Local Database` : عملیات ذخیره‌سازی (Save Operation - Insert/Update)

تصویر:

Placeholder
(این نمودار توالی ساده‌ای از ارتباطات بین موجودیت‌ها برای همگام‌سازی دسته‌ای را نشان می‌دهد، با تمرکز بر جنبه‌های کلی جریان.)

---

6. Sequence Diagram – BulkSync Detailed Flow

توضیح کوتاه:

جزئیات فنی فرآیند Bulk Sync شامل استفاده از RowVersion، مدیریت Deleted Flag، Upsert با Conflict Resolution، پیاده‌سازی Retry و DLQ، و سناریوهای Timeout/Rate Limit. این نمودار به لایه‌های فنی و پیچیدگی‌های همگام‌سازی می‌پردازد.

جزئیات فنی (Technical Details):

- استفاده از RowVersion: برای جلوگیری از پردازش داده‌های قدیمی یا از دست رفته، هر رکورد در Mahak دارای یک شناسه نسخه (مانند RowVersion یا Timestamp) است. در درخواست‌های بعدی، سیستم از آخرین RowVersion دریافت شده استفاده می‌کند تا فقط داده‌های جدیدتر یا تغییر یافته را درخواست کند.
- مدیریت Deleted Flag: Mahak ممکن است از فلگ حذف منطقی (Logical Delete) برای نشان دادن اینکه یک رکورد دیگر فعال نیست، استفاده کند. سیستم ما نیز باید این فلگ را در هنگام همگام‌سازی رعایت کرده و در پایگاه داده محلی خود نیز به طور منطقی رکورد را غیرفعال کند.
- Upsert با Conflict Resolution: هنگام ذخیره داده‌ها، اگر رکوردی با همان کلید اصلی در پایگاه داده محلی وجود داشته باشد، سیستم باید تصمیم بگیرد که آیا آن را به‌روزرسانی کند یا خیر. اگر داده‌های دریافتی و داده‌های موجود در پایگاه داده محلی تفاوت‌های قابل توجهی داشته باشند (مثلاً تاریخچه تغییرات متفاوت)، ممکن است نیاز به منطق پیچیده‌تری برای حل تعارض (Conflict Resolution) باشد.
- پیاده‌سازی Retry و DLQ: اگر ارتباط با Mahak API با خطا مواجه شود، یا اگر پردازش یک دسته از داده‌ها با مشکل روبرو شود (مانند خطای اعتبارسنجی، یا مشکل در دیتابیس)، فرآیند نباید متوقف شود. سیستم باید تلاش‌های مجدد (Retries) را برای آن عملیات انجام دهد. اگر پس از چندین تلاش، عملیات همچنان ناموفق باشد، آن را به یک Dead-Letter Queue (DLQ) منتقل می‌کند تا بعداً توسط یک تیم فنی بررسی شود.
- سناریوهای Timeout/Rate Limit: Mahak API ممکن است محدودیت زمانی (Timeout) برای پاسخگویی یا محدودیت تعداد درخواست در واحد زمان (Rate Limit) داشته باشد. سیستم ما باید بتواند این شرایط را مدیریت کرده و در صورت نیاز، درخواست‌ها را با تاخیر ارسال کند یا تلاش‌های مجدد با فاصله زمانی بیشتر انجام دهد.

شرکت‌کنندگان (Participants):

- Mahak Sync Worker: مسئول اصلی اجرای فرآیند همگام‌سازی.
- Mahak API Adapter: واسط ارتباطی با Mahak API.
- Mahak API: سرویس خارجی.
- Message Broker (RabbitMQ): برای مدیریت Retry و DLQ.
- Local Database: پایگاه داده فروشگاه آنلاین.

جریان دقیق (Detailed Flow):

1. `Mahak Sync Worker` -> `Mahak API Adapter` : درخواست واکشی داده‌ها با آخرین RowVersion (Fetch Data Request with Last RowVersion)
2. `Mahak API Adapter` -> `Mahak API` : ارسال درخواست API، با در نظر گرفتن Rate Limits (Send API Request, respecting Rate Limits)
3. `Mahak API` -> `Mahak API Adapter` : پاسخ حاوی داده‌ها و RowVersion جدید (Response with Data & New RowVersion)
   - Scenario: اگر API Timeout دهد، `Mahak Sync Worker` به `Retry &` `Error Manager` اطلاع می‌دهد.
4. `Mahak Sync Worker` (Bulk Fetcher/Saver): دریافت داده‌ها در بسته‌ها (Receive Data in Batches)
5. `Mahak Sync Worker` (Data Transformer): تبدیل داده‌ها (Transform Data)
6. `Mahak Sync Worker` (Upsert Engine): تلاش برای Upsert در `Local Database`
   - Scenario: اگر Upsert ناموفق بود (مانند Conflict)، `Mahak Sync Worker` (Upsert Engine) از مکانیزم Conflict Resolution استفاده می‌کند.
   - Scenario: اگر `Local Database` خطا داد (مانند Network Issue)، `Mahak Sync Worker` (Retry & Error Manager) پیام خطا را پردازش می‌کند.
7. `Mahak Sync Worker` (Logical Delete Handler): اعمال حذف منطقی در صورت لزوم.
8. مدیریت خطا:
   - در صورت خطا در هر مرحله، `Mahak Sync Worker` (Retry & Error Manager) عملیات را به صف Retry در `Message Broker` می‌فرستد.
   - پس از چند بار تلاش ناموفق، عملیات به DLQ در `Message Broker` منتقل می‌شود.

تصویر:

Placeholder
(این نمودار جزئیات فنی و مراحل پردازش در هنگام همگام‌سازی دسته‌ای را نشان می‌دهد، با تمرکز بر مدیریت خطا، نسخه‌ها و تعارضات.)

---

7. تکنولوژی‌ها و دلیل انتخاب

انتخاب تکنولوژی‌ها برای هر پروژه، تأثیر بسزایی بر موفقیت، مقیاس‌پذیری، نگهداری و کارایی آن دارد. در این پروژه، دلایل زیر مبنای انتخاب تکنولوژی‌های مطرح شده بوده‌اند:

- PostgreSQL 15:
  - رایگان و متن‌باز: کاهش هزینه‌های لایسنس و امکان دسترسی به کد منبع برای سفارشی‌سازی در صورت نیاز.
  - سازگاری با لینوکس/Kubernetes: اطمینان از اجرای آسان و یکپارچه در محیط‌های مدرن استقرار.
  - پشتیبانی از JSONB: امکان ذخیره و کوئری گرفتن از داده‌های نیمه‌ساختاریافته (مانند داده‌های دریافتی از APIها) به صورت کارآمد در کنار داده‌های رابطه‌ای.
  - قابلیت‌های پیشرفته: مانند Partitioning، GIS Extension، Full-Text Search، Replication و انواع داده‌های سفارشی که انعطاف‌پذیری بالایی را ارائه می‌دهند.
  - قابلیت اطمینان و بلوغ: PostgreSQL یکی از پایدارترین و قابل اعتمادترین سیستم‌های مدیریت پایگاه داده رابطه‌ای در جهان است.
- RabbitMQ:
  - پشتیبانی از Pub/Sub: امکان انتشار پیام‌ها به چندین مشترک، که برای اطلاع‌رسانی رویدادها بین سرویس‌های مختلف بسیار مفید است.
  - صف‌های Retry و DLQ: مکانیزم داخلی برای مدیریت خطاها و تلاش‌های مجدد، که به طور قابل توجهی قابلیت اطمینان فرآیندهای ناهمگام را افزایش می‌دهد.
  - تضمین تحویل پیام‌ها: پروتکل‌های انتقال پیام مطمئن، اطمینان حاصل می‌کنند که پیام‌ها حتی در صورت بروز مشکل در شبکه یا سرویس، از بین نمی‌روند.
  - قابلیت کار با پروتکل‌های مختلف: پشتیبانی از STOMP، AMQP و MQTT انعطاف‌پذیری را در انتخاب روش‌های ارتباطی فراهم می‌کند.
  - مقیاس‌پذیری: قابلیت مقیاس‌دهی افقی برای مدیریت حجم بالای پیام‌ها.
- MinIO / S3 Compatible:
  - راهکار ذخیره‌سازی شیءگرا: ایده‌آل برای ذخیره‌سازی فایل‌های حجیم مانند تصاویر محصولات، ویدئوها، یا فایل‌های لاگ که نیازی به کوئری‌های پیچیده ندارند.

- سازگار با S3: امکان استفاده از ابزارها و کتابخانه‌هایی که با AWS S3 کار می‌کنند، و همچنین سهولت مهاجرت به سرویس‌های ابری دیگر در آینده.
- متن‌باز و قابل استقرار در محل (On-Premise): کنترل کامل بر داده‌ها و زیرساخت ذخیره‌سازی.
- مقیاس‌پذیری بالا: طراحی شده برای مدیریت حجم عظیمی از داده‌ها.

- .NET 8 / Go:

- عملکرد بالا: هر دو زبان، به خصوص در نسخه‌های جدیدتر، عملکردی بسیار بالا در اجرای کد و مدیریت درخواست‌ها ارائه می‌دهند که برای یک فروشگاه آنلاین و سرویس‌های همگام‌سازی حیاتی است.
- سازگاری با ساختار ماژولار و میکروسرویس: فریم‌ورک‌های مدرن مانند ASP.NET Core در .NET 8، و ماهیت Go، توسعه و مدیریت میکروسرویس‌ها را تسهیل می‌کنند.
- اکوسیستم قوی: .NET دارای جامعه کاربری بزرگ و کتابخانه‌های فراوان است. Go نیز به دلیل سادگی و کارایی بالا محبوبیت زیادی دارد.
- قابلیت توسعه Cross-Platform: امکان توسعه و استقرار بر روی سیستم‌عامل‌های مختلف (Windows, Linux, macOS).

- JWT Auth:

- احراز هویت امن و ساده: استاندارد صنعتی برای احراز هویت بر اساس توکن، که امکان ارتباط بدون نیاز به جلسه (Stateless) را بین کلاینت و سرور فراهم می‌کند.
- قابلیت حمل: توکن‌های JWT اطلاعات لازم برای احراز هویت را در خود دارند و می‌توانند بین سرویس‌های مختلف منتقل شوند.
- امضای دیجیتال: تضمین عدم دستکاری توکن و اطمینان از منبع آن.

- Kubernetes:

- مدیریت و اسکیل سرویس‌ها: امکان خودکارسازی استقرار، مقیاس‌دهی، و مدیریت برنامه‌های کاربردی در کانتینرها.
- Zero Downtime Deploy: امکان به‌روزرسانی سرویس‌ها بدون ایجاد وقفه در سرویس‌دهی به کاربران.
- قابلیت Self-healing: خودکارسازی بازیابی سرویس‌های از کار افتاده یا ناپایدار.
- مدیریت منابع: تخصیص و مدیریت بهینه منابع سخت‌افزاری (CPU، RAM) برای سرویس‌ها.
- پلتفرم استاندارد برای Microservices: فراهم کردن زیرساخت لازم برای اجرای معماری میکروسرویس.

---

8. نمای کلی جریان همگام‌سازی

فرآیند همگام‌سازی داده‌ها بین فروشگاه آنلاین و سیستم حسابداری محک، ستون فقرات ارتباطی این دو سیستم است. این جریان به صورت ناهمگام و با هدف حفظ یکپارچگی داده‌های مالی و عملیاتی بین پلتفرم‌ها طراحی شده است.

جریان اصلی:

1. زمان‌بندی (Scheduling):

- یک سیستم زمان‌بندی (Scheduler) داخلی در فروشگاه آنلاین، بر اساس یک برنامه از پیش تعیین شده (مثلاً هر 15 دقیقه، یا هر ساعت) اجرای فرآیند همگام‌سازی را آغاز می‌کند.
- هدف از این زمان‌بندی، اطمینان از به‌روز بودن داده‌ها در هر دو سیستم بدون بارگذاری بیش از حد بر روی APIهای خارجی یا پایگاه داده‌ها است.

2. درخواست داده از Mahak API:

- Scheduler، سرویس `Mahak Sync Worker` را فعال می‌کند.
- `Mahak Sync Worker` با استفاده از آخرین وضعیت ثبت شده (مانند آخرین RowVersion یا Timestamp دریافتی) در پایگاه داده داخلی خود، درخواستی را به Mahak API (از طریق `Mahak API Adapter`) ارسال می‌کند.
- این درخواست برای دریافت داده‌هایی است که از زمان آخرین همگام‌سازی تغییر کرده‌اند یا جدیداً ایجاد شده‌اند (مثلاً محصولات جدید، سفارشات تکمیل شده، تغییرات انبار).
- در این مرحله، مکانیزم‌های مدیریت Rate Limit و Timeout در `Mahak API Adapter` فعال هستند تا از بروز خطا به دلیل محدودیت‌های سیستم Mahak جلوگیری شود.

3. دریافت و پردازش اولیه داده‌ها:

- Mahak API پاسخ را ارسال می‌کند که شامل مجموعه‌ای از داده‌ها است. این داده‌ها ممکن است به صورت دسته‌ای (Batch) و صفحه‌بندی شده دریافت شوند.
- `Mahak Sync Worker` داده‌های دریافتی را ابتدا در Data Transformer پردازش می‌کند تا فرمت آن‌ها برای استفاده در سیستم داخلی مناسب شود. این شامل نگاشت فیلدها، تبدیل انواع داده و اطمینان از صحت داده‌ها است.

4. ارسال به صف پیام (RabbitMQ):

- داده‌های پردازش شده در قالب پیام‌هایی (هر پیام شامل یک یا چند رکورد) به صف پیام RabbitMQ ارسال می‌شوند.
- ارسال به صف پیام، به سیستم اجازه می‌دهد تا پردازش اصلی را به صورت ناهمگام و با قابلیت اطمینان بالاتری انجام دهد. همچنین امکان مدیریت Retry و DLQ را فراهم می‌کند.

5. پردازش توسط Mahak Sync Worker (Worker Pool):

- چندین نمونه از `Mahak Sync Worker` به صورت موازی به صف پیام گوش می‌دهند و پیام‌ها را دریافت می‌کنند.
- هر Worker، یک دسته از داده‌ها را پردازش می‌کند:
  - Upsert Engine: عملیات "Insert" یا "Update" را در پایگاه داده محلی فروشگاه آنلاین انجام می‌دهد. در صورت وجود رکورد مشابه، از منطق Conflict Resolution برای اطمینان از درستی داده‌ها استفاده می‌کند.
  - Logical Delete Handler: اگر داده‌های Mahak نشان‌دهنده حذف منطقی باشند، این مکانیزم در پایگاه داده داخلی نیز رکورد مربوطه را غیرفعال می‌کند.

6. مدیریت خطا و تلاش‌های مجدد:

- اگر در هنگام پردازش یک دسته از داده‌ها (مانند Upsert در پایگاه داده) خطایی رخ دهد، `Retry & Error Manager` فعال می‌شود.
- Worker سعی می‌کند عملیات را مجدداً انجام دهد (Retry). برای این کار، پیام به صف Retry در RabbitMQ بازگردانده می‌شود.
- اگر پس از چندین تلاش، عملیات همچنان ناموفق باشد، پیام به Dead-Letter Queue (DLQ) منتقل می‌شود. این پیام‌ها توسط تیم عملیات یا توسعه‌دهندگان برای بررسی علت اصلی خطا جمع‌آوری می‌شوند.

7. ثبت وضعیت و لاگینگ:

- تمام عملیات همگام‌سازی، موفقیت‌ها و شکست‌ها در لاگ‌های سیستم ثبت می‌شوند.
- آخرین RowVersion یا Timestamp پردازش شده برای هر نوع داده، در پایگاه داده داخلی ذخیره می‌شود تا در چرخه همگام‌سازی بعدی مورد استفاده قرار گیرد.

این فرآیند ناهمگام تضمین می‌کند که سیستم فروشگاه آنلاین همواره با داده‌های مالی و عملیاتی به روز از سیستم Mahak هماهنگ باشد، ضمن اینکه از قابلیت اطمینان بالا و مدیریت خطای مؤثر برخوردار است.