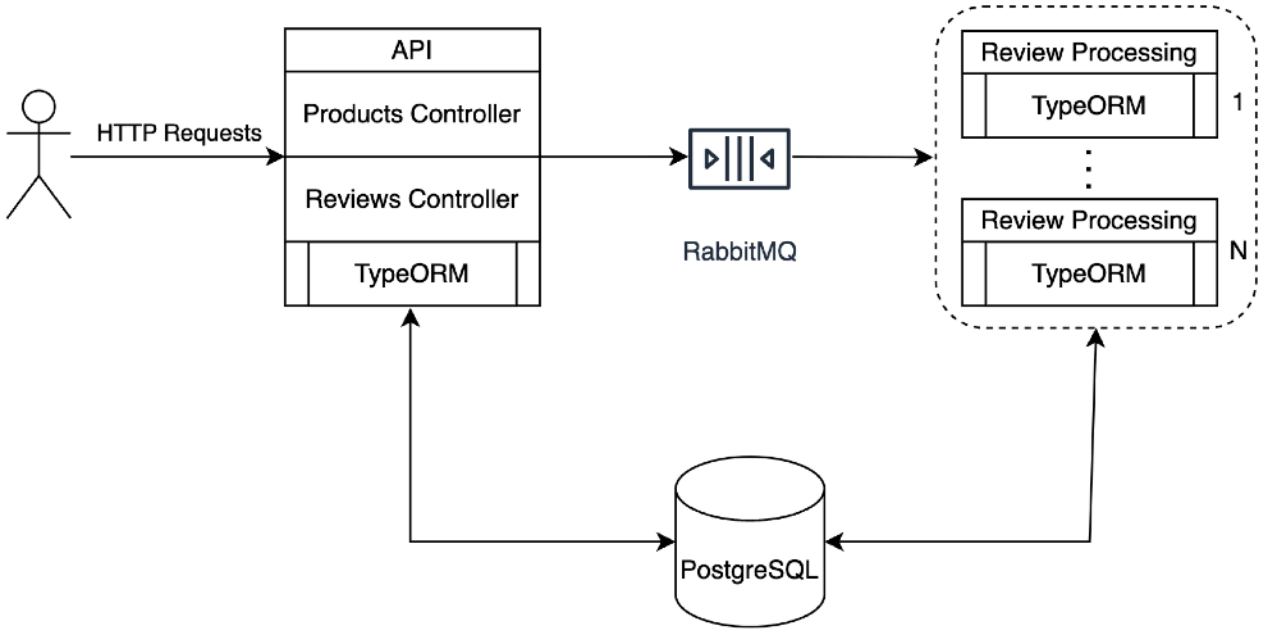
Design overview



Services based on NestJS
Persistent storage - PostgreSQL
ORM - TypeORM
Transport - RabbitMQ
Deployment and orchestration - Docker Compose

Alternatives:
- Kubernetes + KEDA to scale containers based on the number of events(reviews)
- Serverless: Amazon API Gateway + AWS Lambdas + SQS

Thoughts and trade-offs

1. Code organization - standard or monorepo
   Initially followed the standard code organization approach but later transitioned to a monorepo to prevent redundancy in artifacts, configurations, and models.
2. RESTful API design - nested or non-nested endpoints
   Requirements specify the need for two APIs—one for products and another for reviews. Maybe I've got it wrong and this is not strict one.
   A review can't exist without an associated product and in my opinion it looks like we need to use nested endpoints with the pattern **/products/{id}/reviews**. Moreover we need to know the product id every time we do something with the review to notify review processing service. On the other hand we can modify and delete only by review id.
3. Caching
   Used the simplest option - built-in auto-caching responses.
   Can be also used NestJS cache manager or external services like Redis.
   Didn’t get in which scenario product rating needs to be cached, rating is part of the product information being returned.

What is not covered

1. Application configurations and environment variables, especially resources credentials
2. Didn't manage to solve the NestJS ClientProxy injection issue in Testing Module(similar one)