

# Architecture Decision Records

Decisions made during the design process and the reasoning behind them.

## Table of Contents

ADR 001: Choose Microservices as Architecture.....	2
ADR 002: Choose Azure API Management as API Gateway .....	3
ADR 003: Choose React-TypeScript for Frontend .....	4
ADR 004: Choose C# for Backend.....	5
ADR 005: Choose Database per Service as Data Pattern .....	6
ADR 006: Choose PostgreSQL for Order Management .....	7
ADR 007: Choose MongoDB for Product Management.....	8
ADR 008: Choose Solr for Search Engine .....	9
ADR 009: Choose Azure for hosting containers .....	10

# ADR 001: Choose Microservices as Architecture

- **Context:**
  - My system requires a very scalable architecture since I must be able to scale to 1 million concurrent users.
- **Consequences:**
  - **Positive:** Efficient Scaling, very modular, very scalable.
  - **Negative:** Complex, high costs.
  - **Future Considerations:** As the number of microservices grows, managing dependencies and interactions between services can become more complex. Over time it may be necessary to implement a service registry and a centralized configuration management system to ensure reliable service discovery and configuration.
- **Alternatives Considered:**
  - **Monolithic, Two-Tier, Three-Tier:** Rejected because less efficient in scaling horizontally
  - **SaaS:** Rejected because limited control and lack of experience
  - **Headless:** Rejected because lack of experience
- **Links and References:**
  - **IP – Research:** There is a dedicated section in this research about the software architecture.

## ADR 002: Choose Azure API Management as API Gateway

- **Context:**
  - My system requires a free to use API gateway with built in features such as rate limiting, server discovery, authorization, authentication, auto-scaling, load balancing, low latency, DDoS protection, firewall and most importantly, free to use since I don't have a budget
- **Consequences:**
  - **Positive:** Free to use (100 free credits) and many built in features: rate limiting, server discovery, authorization, authentication, auto-scaling, load balancing, low latency, DDoS protection, firewall.
  - **Negative:** Not as many available regions as AWS.
  - **Future Considerations:** If I run into scalability issues, I might have to consider adding a load balancer in between the API Gateway and the services.
- **Alternatives Considered:**
  - **Google API Gateway, IBM API Connect, Kong Gateway, MuleSoft Anypoint Flex Gateway , Boomi API Management, WSO2 API Manager:** Rejected because not free to use.
  - **Cloudflare:** Rejected because not as many built in features as Azure.
  - **AWS:** Rejected because a credit card is required to use AWS and I don't have one.
- **Links and References:**
  - **IP - What API Gateway should I choose:** In this research I compare API Gateways with each other.

## ADR 003: Choose React-JS for Frontend

- **Context:**
  - My system requires a well performing frontend.
- **Consequences:**
  - **Positive:** High performance, high community support. Minor experience with React. Revolutionary live debugging.
  - **Negative:** Complex.
  - **Future Considerations:** Dedicated research on performance on frontends to decide if it's the best choice.
- **Alternatives Considered:**
  - **Vue.js:** Rejected because I don't have prior experience. I have tried it out briefly hoping it was easier than React but it wasn't.
  - **Flask:** Good Alternative, not as good performing as React but I have experience in flask and python.
  - **React-TypeScript:** I was initially going to choose React-TS, because I had very little experience with typescript. But the community support for React-TS is limited, and since I do have some react experience it would make sense to switch to React-JS since it does have a lot of community support.
- **Links and References:**
  - <https://www.simform.com/blog/best-frontend-frameworks/>

## ADR 004: Choose C# for Backend

- **Context:**
  - My system requires a well performing backend.
- **Consequences:**
  - **Positive:** High community support, I have years of experience in C#.
  - **Negative:** Possibly better solutions that I didn't consider.
  - **Future Considerations:** When the project is running and we run into performance issues it might be valuable to research if there are better performing backends.
- **Alternatives Considered:**
  - **Python:** I have a lot of experience in Python too. If for some reason C# won't work out, I can always switch to python without any hesitation.
- **Links and References:**
  - -

# ADR 005: Choose Database per Service as Data Pattern

- **Context:**
  - The system being developed is based on a microservice architecture.  
Database per service is not only a best practice in the microservice architecture, but it also satisfies my quality requirements.
- **Consequences:**
  - **Positive:** independently scalable, freedom in database choice per service which allows me to choose a fitting db per service, prevent concurrency conflicts
  - **Negative:** Challenging and higher costs.
  - **Future Considerations:** -
- **Alternatives Considered:**
  - -
- **Links and References:**
  - **IP - Microservice Research:** I briefly go over the pros and cons of db per service.

## ADR 006: Choose PostgreSQL for Order Management

- **Context:**
  - The Order Management service needs its own database. This database has to be ACID compliant and is preferably scalable horizontally.
- **Consequences:**
  - **Positive:** ACID compliant, providing very strong consistency. Also high performance.
  - **Negative:** Needs additional setup to scale horizontally.
  - **Future Considerations:** This is a free option, and there are most likely better options that aren't free such as CockroachDB, TiDB and YugabyteDB since these are ACID compliant and made to be scalable horizontally.
- **Alternatives Considered:**
  - **CockroachDB, TiDB, YugabyteDB:** Rejected because not free, or free trial isn't long enough
- **Links and References:**
  - **IP - SQL vs NoSQL Research:** Research about the difference between relational databases and document-oriented databases, also dive into ACID databases and BASE databases. This document further explains why I choose a database for a service

# ADR 007: Choose MongoDB for Product Management

- **Context:**
  - The Product Management service needs its own database. It is a best practice to use a document oriented database for product management
- **Consequences:**
  - **Positive:** Document oriented BASE database. Free to host.
  - **Negative:** -
  - **Future Considerations:** -
- **Alternatives Considered:**
  - -
- **Links and References:**
  - **IP - SQL vs NoSQL Research:** Research about the difference between relational databases and document-oriented databases, also dive into ACID databases and BASE databases. This document further explains why I choose a database for a service



## ADR 008: Choose Solr for Search Engine

- **Context:**
  - Usually a Search Engine service is used in combination with something like Elastic Search. However, Elastic Search isn't free to use. An alternative I managed to find was Solr.
- **Consequences:**
  - **Positive:** Free to use.
  - **Negative:** Not as much community support as Elastic Search.
  - **Future Considerations:** Consider if Elastic Search is a better choice, and research if it's worth changing to Elastic Search.
- **Alternatives Considered:**
  - **Elastic Search:** Rejected because not free to use.
- **Links and References:**
  - **IP - SQL vs NoSQL Research:** Research about the difference between relational databases and document-oriented databases, also dive into ACID databases and BASE databases. This document further explains why I choose a database for a service

## ADR 009: Choose Render for hosting containers

- **Context:**
  - The system needs a platform wherein I can host containers so that they can communicate with each other independently from my local machine.
- **Consequences:**
  - **Positive:** Free to use, easy to implement.
  - **Negative:** Very limited, but there for only 7\$ a month I can use more advanced features.
  - **Future Considerations:** Research if AWS is more fitting.
- **Alternatives Considered:**
  - **Heroku:** Rejected because not free anymore.
  - **AWS, Fly.io, Akamai:** Rejected because free tier requires a credit card.
  - **Render:** Free Alternative.
  - **Railway:** 5\$ worth of credit. Decent alternative.
  - **Azure:** Free 100\$ Credits, Hard learning curve and I wasted all my credits.
- **Links and References:**
  - **IP - What API Gateway should I choose:** In this research I compare API Gateways with each other.

# ADR 010: Choose Auth0 as Authentication Provider

- **Context:**
  - The system needs a way to authenticate.
- **Consequences:**
  - **Positive:** Free to use. Easy to implement. High community support.
  - **Negative:** -
  - **Future Considerations:** Research if there is a better alternative
- **Alternatives Considered:**
  - -
- **Links and References:**
  - -