ITSAUR

SOFTWARE SOLUTIONS

Grigoriadis Grigoris

Co-Founder & Software Architect @ ITSAUR SOFTWARE SOLUTIONS

Microservices enthusiast !

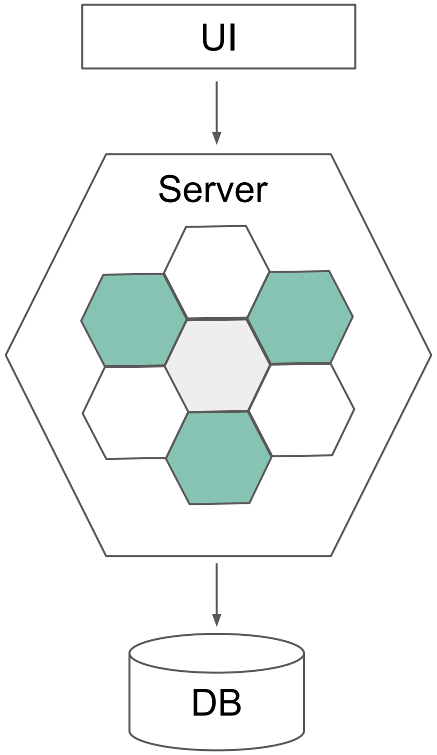
Event-Based Microservices

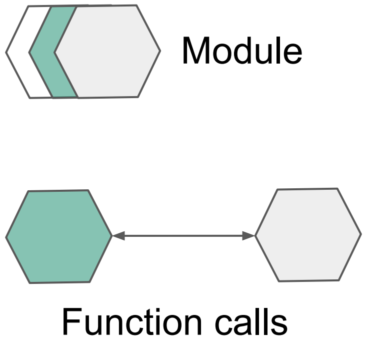
Introduction

What we will cover

- What is Monolithic Architecture
- What is Microservices Architecture
- Pros/Cons
- Moving to Event Based Microservices

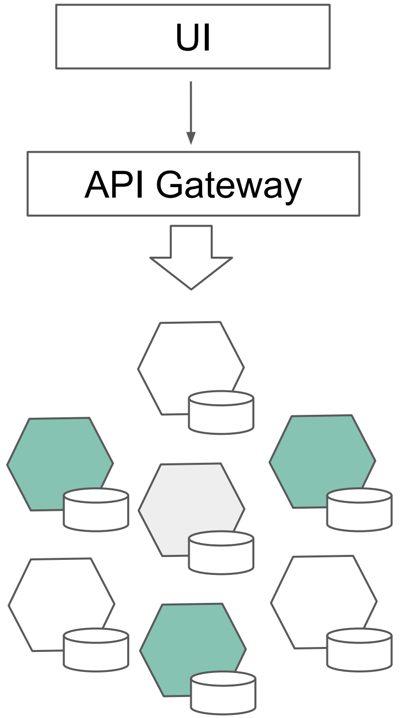
Monolithic Architecture

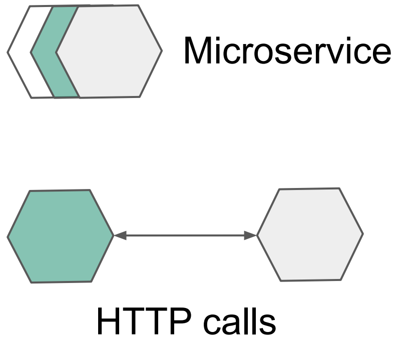




- Building block is the code modules
- Modules communicate with each other using function calls
- System is deployed and run as a single OS process
- A Database is used in order to store the system state, usually a relational database

Microservice Architecture - Bare Minimum



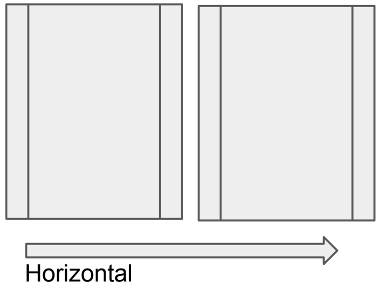


- Building block is the microservices
- Services communicate with each other via a lightweight interoperable communication protocol, usually HTTP
- Each service MUST have its own database
- An API Gateway provides a unified API for our system

Q&A

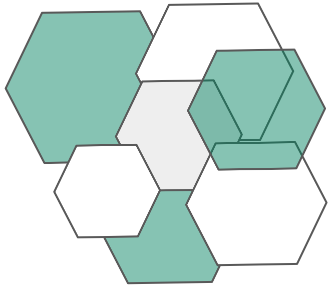
Architectural Factors

Scalability



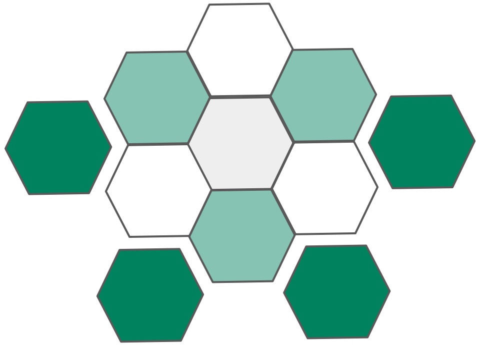
How well the system handles a growing amount of data/requests?

Maintainability



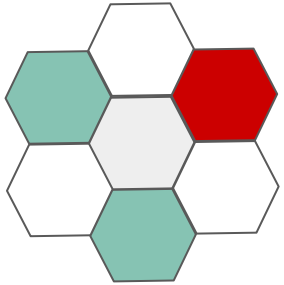
Is the system simple and easy to understand?

Evolvability



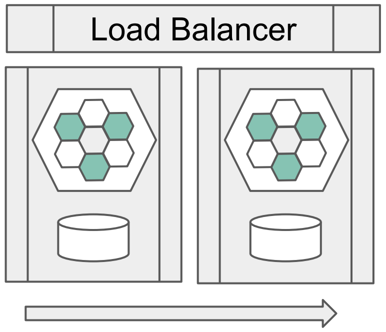
Is the system design in such a way that can evolve along with the product requirements and the changing ecosystem?

Fault Tolerance



When part of the system fails does it fail completely?

Scalability



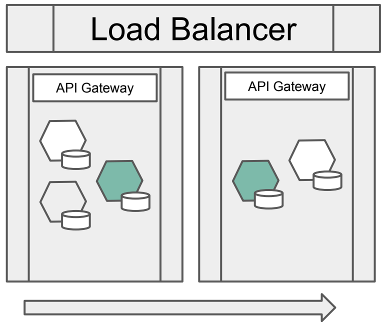
Monolithic

👍

- Easier to scale

👎

- The entire system must be scaled
- Different modules might have different resource requirements



Microservices

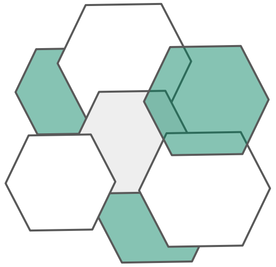
👍

- Services can be deployed on different machines
- Services can be scaled independently

👎

- Databases must be scaled along with microservices
- Service discovery

Maintainability



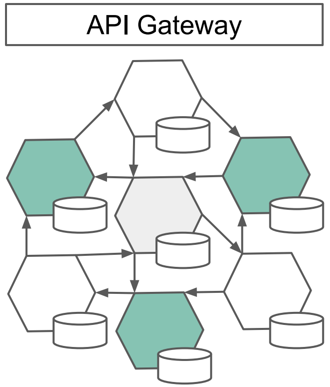
Monolithic

👍

- Easier to debug

👎

- Boundaries between modules tend to break
- Code size becomes intimidating
- A change can impact the whole system



Microservices

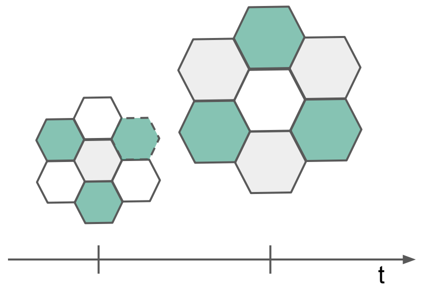
👍

- Each service is small in code size and easier to understand
- Easier to work with different teams

👎

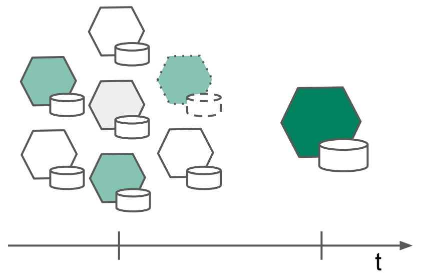
- Communication between services might become complicated and difficult to follow
- Business logic might leak to API Gateway and can become complex

Evolvability



Monolithic

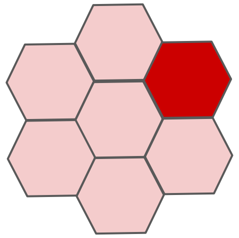
- Any change, such language change, db change etc. must be applied to the whole application
- Refactoring database requires coordination between multiple teams



Microservices

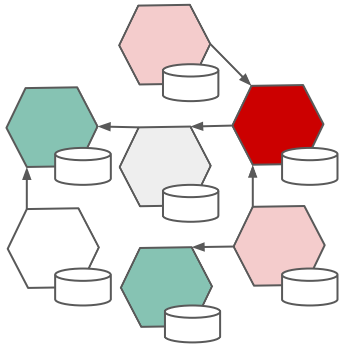
- Microservices can be rewritten individually in different language using different database
- Changes in infrastructure can be applied gradually to each microservice
- Refactoring in Database is easier since each database is governed by 1 team

Fault Tolerance



Monolithic

- If a module has a memory leak or malfunctions the whole system might stop working



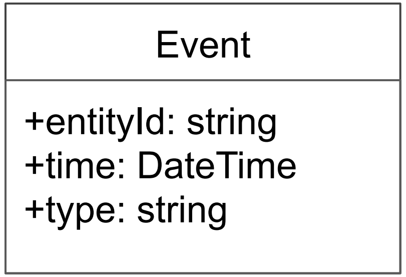
Microservices

- Part of the system will continue to work even if some services are not working

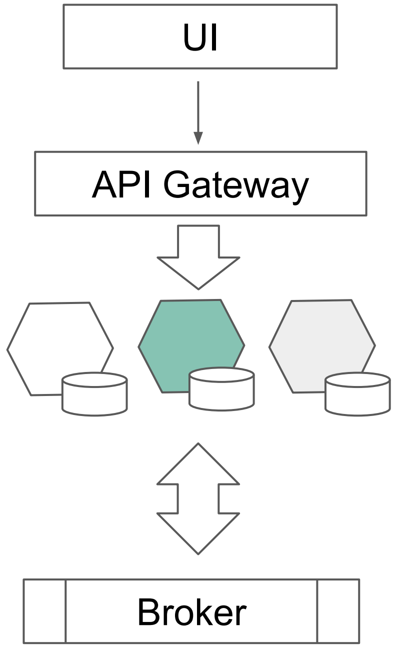
Q&A

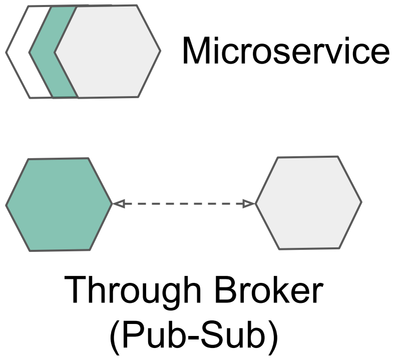
Microservice Architecture - Event Based

- An event is a fact, something that has already happened and cannot change
- Every state change must generate an event
- Events should NOT be overly generic (e.g. OrderUpdated), prefer more specific events like OrderStatusChanged or even better OrderDispatched etc.
- Events should contain only data relevant to the event.



Microservice Architecture - Event Based





- Microservices publish events in the broker and consume events that are of interest in order to update their databases
- Services do not communicate with each other to exchange information, they duplicate the data they need from events
- System has Eventual Consistency

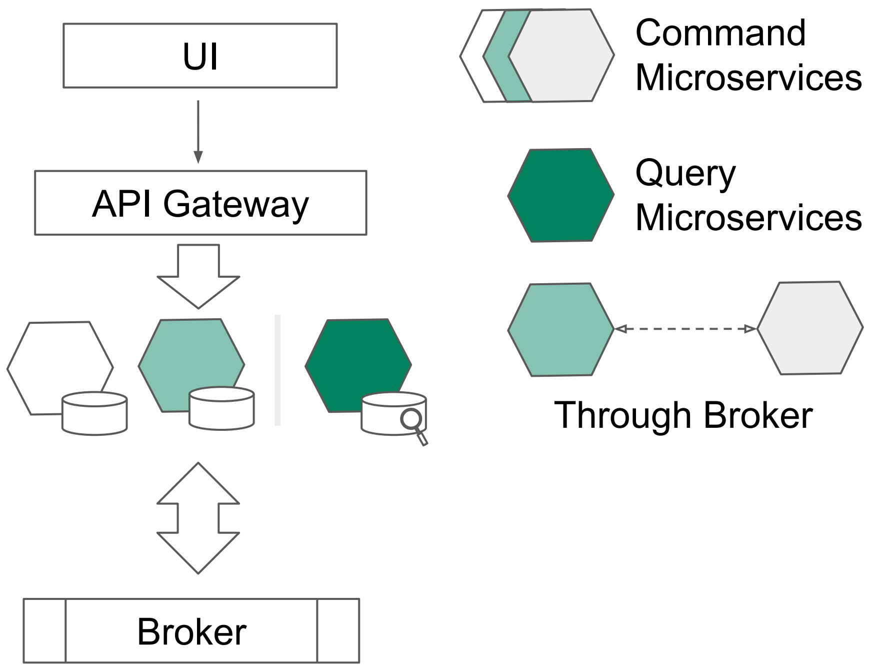
👍

- Better performance
- Fault tolerant
- Service discovery only used by API Gateway

👎

- Difficult to design
- API Gateway is still complex

Microservice Architecture - CQRS



- Command microservices update the state of our system and publish events
- Query microservices deal with keeping the data in a schema appropriate for the UI

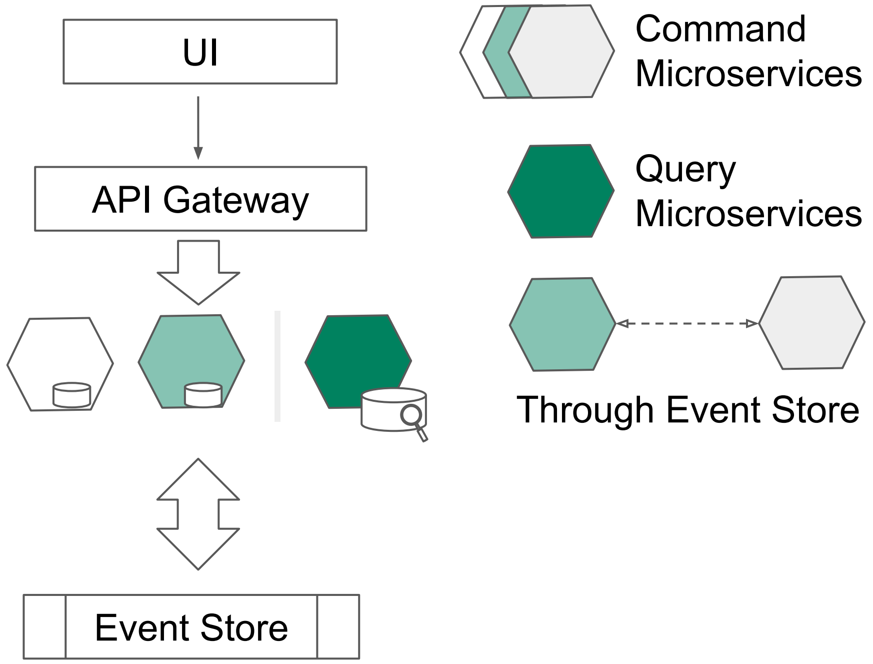
👍

- Better performance
- Decoupled read from write schema
- API Gateway does not need to compose calls

👎

- Still need to commit to multiple middleware

Microservice Architecture - Event Sourcing



- All state changes in our system are represented as events.
- All events are persisted in an event store.
- Each microservice creates its state from the events in the event store and keeps a snapshot for faster access.

- Better performance
- Each microservice can recreate its state

- Difficult to implement

Q&A

Thank you!