Microservices

Tejas Parikh (t.parikh@northeastern.edu)

CSYE 6225
Northeastern University

What Are Microservices?

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of loosely coupled services.

Microservices are…

- Small, and focused on doing one thing well
- autonomous

Microservices are small, and focused on doing one thing well

- Codebases grow as we write code to add new features.
- Over time, it can be difficult to know where a change needs to be made because the codebase is so large.
- Code related to similar functions starts to become spread all over, making fixing bugs or implementations more difficult.
- Microservices focus our service boundaries on business boundaries, making it obvious where code lives for a given piece of functionality.

Microservices are autonomous

- Microservices are a separate entity.
- Microservices are deployed as isolated service.
- All communication between the services themselves are via network calls, to enforce separation between the services and avoid the perils of tight coupling.
- Microservices need to be able to change independently of each other, and be deployed by themselves without requiring consumers to change.
- Services expose an API and other services communicate with it via those APIs.

Benefits of Microservices

- Technology Heterogeneity – Pick right tool (programming language, framework, operating system, database, etc.) for the job.
- Resiliency – A service failure does not cascade into total system failure.
- Scaling – Scale on the services that need to be scaled.
- Ease of Deployment – Service deployments are independent of the rest of the components.
- Composability – Microservices can be designed to allow its functionality be consumed in many different ways for different purposes.
- Optimizing for Replaceability – Easy to kill a service when not needed.

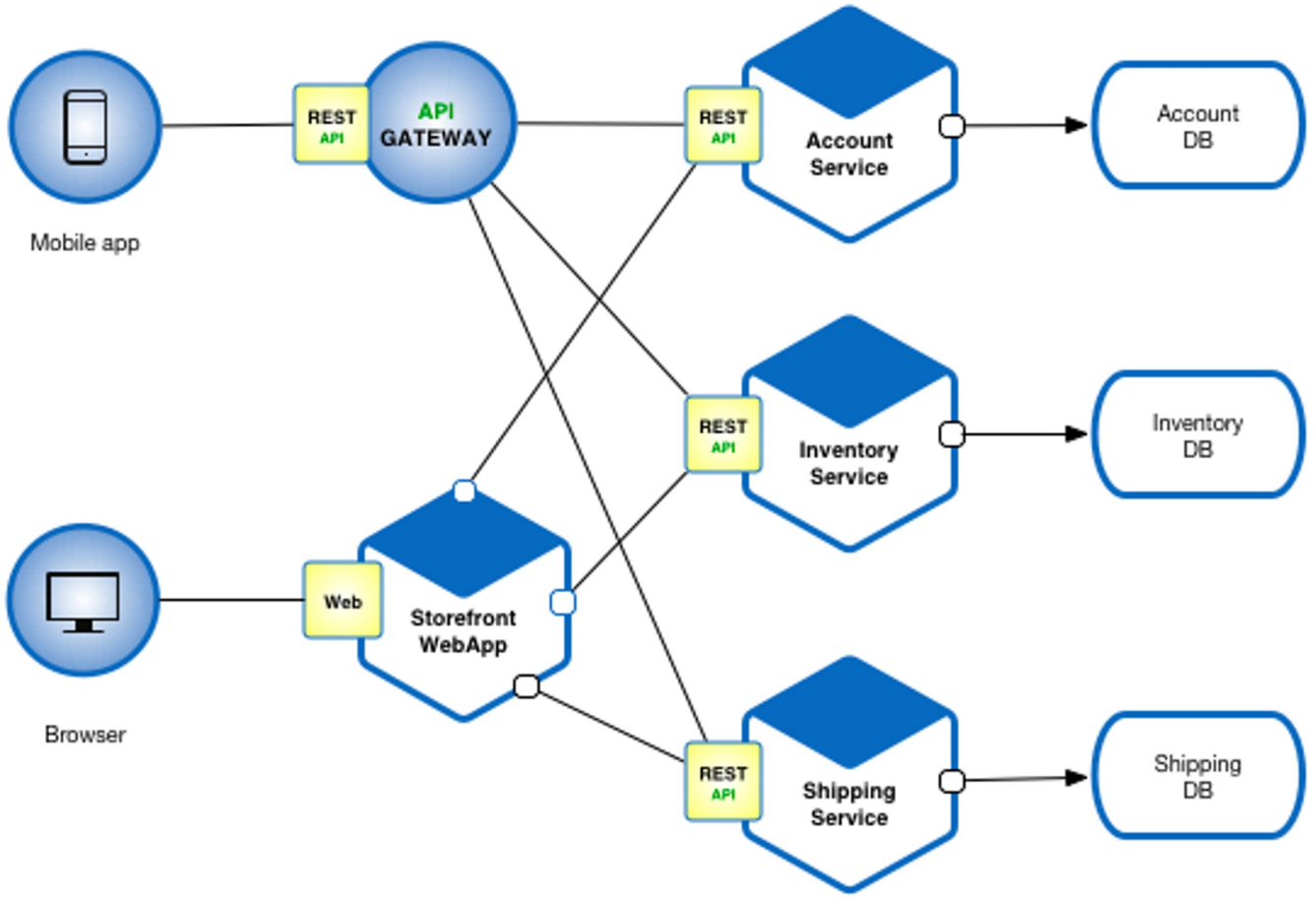
What Makes a Good Service?

- Loose Coupling – When services are loosely coupled, a change to one service should not require change to another.
- High Cohesion – Related behavior should sit together and unrelated behavior to sit elsewhere.

Integration

- Shared Database?
- Synchronous Versus Asynchronous
- Remote Procedure Calls
- REST
- Message Queues
- Reactive Design

Fictitious e-commerce application



Drawbacks of Microservices

- Services can be too small.
- Microservices architecture introduces additional complexity and new problems to deal with such as network latency, message formats, load balancing and fault tolerance.
- Calls between services over a network have higher cost in term of network latency and processing time compared to in-process calls in monolithic services.

Additional Resources

See Lecture Page