MICROSERVICES

M321 Dokumentation

Ael Banyard, Evan Lüber, Yannick Schläpfer

Inhaltsverzeichnis

1 Projektdefinition

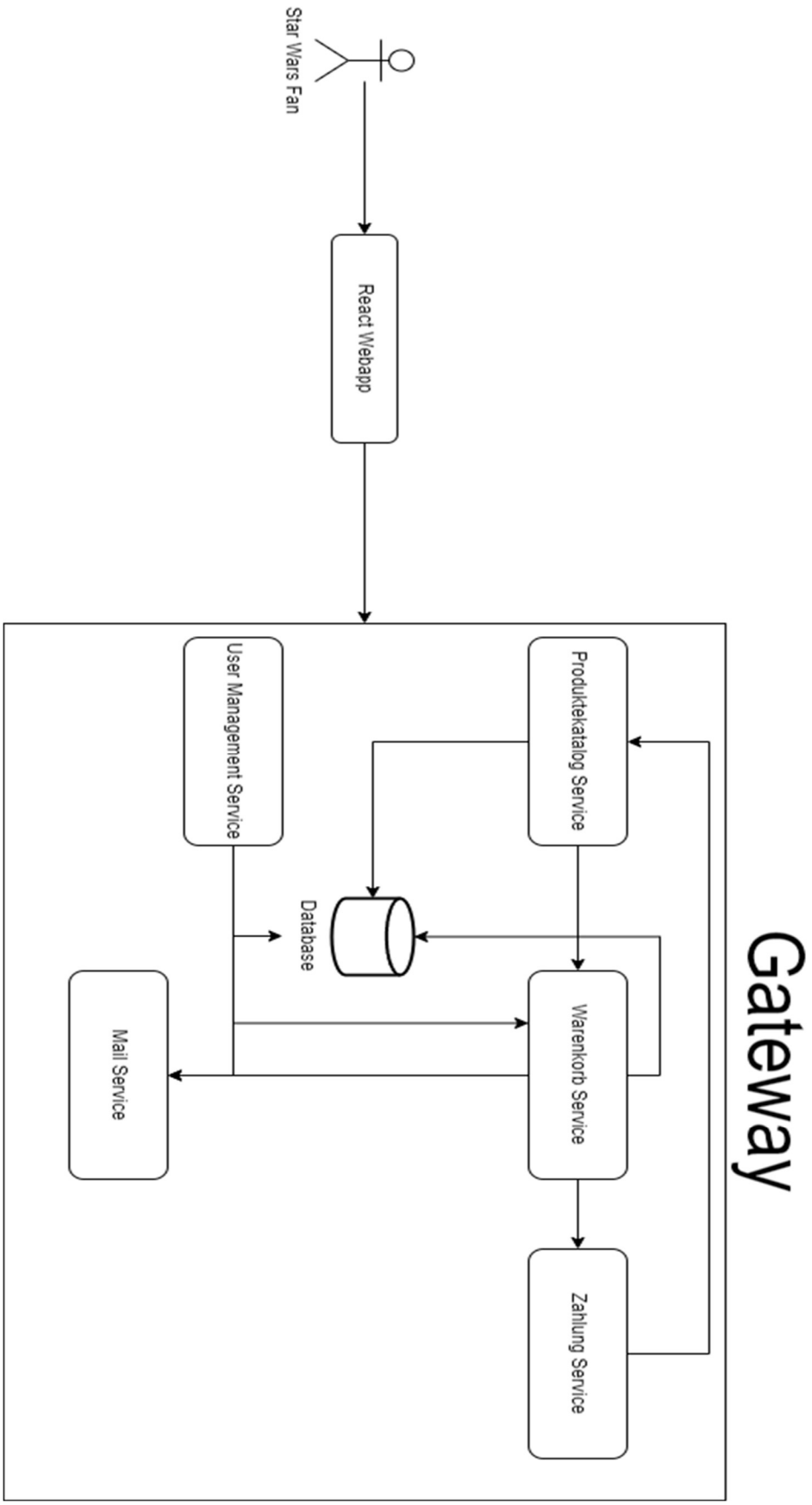
In diesem Projekt geht es darum, sich Gedanken über Mikroservices zu machen. Wir haben uns für den Auftrag für einen Onlineshop entschieden. In diesem Shop werden wir fiktionale Star Wars Objekte verkaufen, wie zum Beispiel ein T-Shirt mit einem Spruch oder eine Tasse.

Das Frontend gestalten wir mit ReactJS und das Backend wird aus mehreren Microservices bestehen. Diese werden ein Produktekatalog-Service, Warenkorb-Service, Zahlungs-Service, Mail-Service und Usermanagement-Service sein.

In dieser Dokumentation geht es hauptsächlich um die Planung des Onlineshops. Die Implementierung uns hierbei nebensächlich. Zum Speichern unserer Dokumentation haben wir ein GitHub Repository in unserem GitHub-Team erstellt. Somit hatte unser ganzes Team schnellstmöglichen Zugriff auf den gesamten Umfang des Auftrags und war in der Lage, falls nötig, diesen weiterzureichen.

GitHub (https://github.com/BLS-YEA/M321_Microservices)

2 Architekturskizze



3 Erklärung zur Architektur

3.1.1 Produktekatalog-Service

Zweck/Aufgabe

- Verwaltung des Lagerbestands
- Speicherung und Verteilung der von uns verkauften Produkte

Technologien

- Spring Boot für den Service, Spring Data JPA für den Datenbankzugriff, und eine Verbindung zu MongoDB

Endpoints

- /products GET all products
- /products/{id} GET product by id
- /products POST new product
- /products/{id} PATCH update a product
- /products/{id} DELETE delete a product by id

3.1.2 Warenkorb-Service

Zweck/Aufgabe

- Ermöglicht Kunden das Hinzufügen und Entfernen von Produkten in einem virtuellen Warenkorb.

Technologien

- Spring Boot und eine Verbindung zur MongoDB

Endpoints

- /cart GET alle Produkte zurückgeben
- /cart/{id} GET ein Produkt anhand id zurückgeben
- /cart POST ein Produkt zum Warenkorb hinzufügen
- /cart DELETE Warenkorb leeren
- /cart/{id} DELETE ein bestimmtes Produkt entfernen
- /cart/{id} PATCH ein Produkt aktualisieren (z.B Anzahl)

3.1.3 Zahlungs-Service

Zweck/Aufgabe

- Abwicklung der Zahlungen über PayPal

Technologien

- Spring Boot für den Service
- PayPal für die Zahlungsabwicklung

Endpoints

- /capture POST Abschluss der Zahlung
- /init POST Beginn der Zahlung
- /cancel POST Abbruch der Zahlung

3.1.4 Mail-Service

Zweck/Aufgabe

- Verschicken von Bestätigungsmails an Kunden
- Verschicken von Bestellinformationen an Kunden
- Verschicken von Verifizierungsmails an Kunden
- Verschicken von Newsletter an Kunden

Technologien

- Spring Boot für den Service
- Resend für das Verschicken von Mails

Endpoints

- /sendVerify POST Verify-Mail Senden
- /sendPasswordForgotten POST Passwort-Mail senden
- /sendOrdered POST Bestellbestätigung-Mail senden
- /sendNews POST Newsletter senden

3.1.5 Usermanagement -Service:

Zweck/Aufgabe:

- Zuständig für die Verwaltung von Benutzerinformationen (Name, Adresse etc.)

Technologien:

- Spring Boot, Spring Data JPA, und zugriff auf die MongoDB

Endpoints:

- GET /user Accountdaten abrufen
- GET /user/{id} Abrufen von Informationen pro Kunde
- POST /user Hinzufügen eines neuen Kunden
- PUT /user/{id} Aktualisieren von Kundendaten eines Kunden.
- POST /user/register als Neukunde registrieren
- POST /user/login Anmelden

4 Sicherheitsaspekte

Für unsere Microservices haben wir ein umfassendes Sicherheitskonzept entwickelt. Um maximale Sicherheit zu gewährleisten, übergeben wir viele Sicherheitsaspekte an Drittanbieter, da diese meist sicherere Implementierungen anbieten.

Sicherheitsmassnahme	Beschreibung
Authentifizierung und Autorisierung	Nutzung von Diensten wie OAuth 2.0, um nur berechtigten Nutzern den Zugriff zu erlauben
Datenverschlüsselung	Sensible Daten wie Passwörter und Tokens werden verschlüsselt übertragen, um Sicherheit während der Übertragung zu gewährleisten.
API-Endpunkte	Implementierung von Rate Limiting, um Missbrauch zu verhindern. Dies schützt vor Brute-Force- und DDoS-Angriffen.
Monitoring	Einrichtung von Monitoring- und Logging-Systemen, um problematische Aktivitäten zu erkennen und auszugrenzen.
Datensicherung und Wiederherstellung	Regelmäßige Backups der Datenbank, um im Falle eines Datenverlusts eine Wiederherstellung zu ermöglichen.
Sicherheit der Anbieter	Auswahl von Drittanbietern, die keinen Sicherheitsvorfall hatten und ihre Sicherheitssoftware auf dem neusten Stand halten.

Wie schon oben bereits erwähnt, würden wir PayPal für die Zahlungsabwicklung verwenden.

4.1 Error Handling

Jede Applikation benötigt effektive Fehlerbehandlungsmechanismen, insbesondere bei Mikroservices. Es ist wichtig, frühzeitig über Probleme informiert zu werden, um die Ursache schnell zu erkennen und zu beheben. Hier sind einige wichtige Massnahmen:

Handling-Massnahme	Beschreibung
Fehlerbehandlung	- **Fehlerseiten im Frontend:** Fehler werden im Frontend benutzerfreundlich angezeigt - **Fehlerprotokollierung:** Fehler werden für die Weiterentwicklung protokolliert
Logging und Monitoring	- **Logging:** Fehlerprotokollierung in Log-Dateien für spezifischere Fehlerbeschreibung - **Monitoring:** Somit kann der Zustand der Services überwacht werden
Automatisierte Tests	- **Unit-Tests:** Zur kontinuierlichen Prüfung der Funktionalität - **Integrationstests:** Zur Prüfung der Zusammenarbeit der Services
Skalierung und Backups	- **Automatische Skalierung:** Somit werden Überlastungen vermieden und eine anhaltende Performance gewährleistet - **Automatische Backups:** Bei Datenverlust ist es somit möglich, die meisten Daten wieder zu erhalten
Kontinuierliche Verbesserung	- **Regelmässige Updates:** Dadurch ist es möglich die Software auf dem neusten Stand zu halten und Sicherheitslücken zu vermeiden

Durch diese Massnahmen wird die Robustheit und Zuverlässigkeit der Applikation deutlich erhöht, wodurch eine stabile und sichere Nutzung gewährleistet wird.

4.2 Reflexion

Das Projekt verlief insgesamt gut, obwohl wir zu Beginn sehr unorganisiert waren. Wir wussten nicht genau, was zu tun war und welche Anforderungen erfüllt werden mussten, da uns nur zwei Wochen zur Verfügung standen. Anfangs war es schwierig, einen Ansatzpunkt zu finden. Nachdem wir jedoch das Dokument gelesen hatten, wussten wir, wie wir vorgehen sollten. Nach einer Besprechung entschieden wir uns, einen Star Wars Fan Shop zu erstellen. Anschließend erstellten wir die Projektdefinition und führten ein Brainstorming durch, um die benötigten Services zu identifizieren.

Durch das Brainstorming konnten wir schnell alle erforderlichen Services ermitteln. Damit hatten wir eine solide Grundlage für die Architekturskizze, die wir mit Draw.io erstellten. Wir überprüften die Verbindungen mehrfach, um ihre Richtigkeit sicherzustellen. Dies erwies sich als sehr hilfreich, da wir dadurch viele Fehler korrigieren konnten, die wir allein möglicherweise übersehen hätten. Zudem ersparten wir uns später die Mühe, die Verbindungen erneut durchdenken zu müssen, wenn wir die Services beschrieben.

Die Beschreibung der Services fiel uns recht leicht und wir konnten sie schnell verfassen. Durch einige Recherchen im Internet erkannten wir, dass es notwendig war, Sicherheitsaspekte und Fehlerbehandlung in die Dokumentation aufzunehmen. Mithilfe von Vorlagen konnten wir diese Aspekte problemlos umsetzen. Trotz des holprigen Starts kamen wir insgesamt gut voran, verteilten die Aufgaben effizient und kommunizierten einwandfrei miteinander. Für das nächste Mal würden wir das Aufgabendokument vollständig durchlesen, um ein besseres Verständnis zu erlangen.