Drink Domain

M321 – API 21b

Von Gian-Luca Zwahlen, Kian Ganz, Quentin Herren und Sky Bollin

Inhaltsverzeichnis

Contents

Einleitung

Unsere Webseite wurde mit dem Ziel gemacht eine digitale Bar mit allen zur Verfügung gestellten Cocktails und Flaschen einfach darzustellen für alle Gäste. Auch sollten die Gäste die Flaschen bewerten können und einen kleinen Eintrag in unserem Gästebuch erstellen. Dies sollte für alle eine übersichtliche Darstellung für alle zeigen genauso wie direkt zeigen, wenn Nachschub gebraucht ist.

Vorgehen

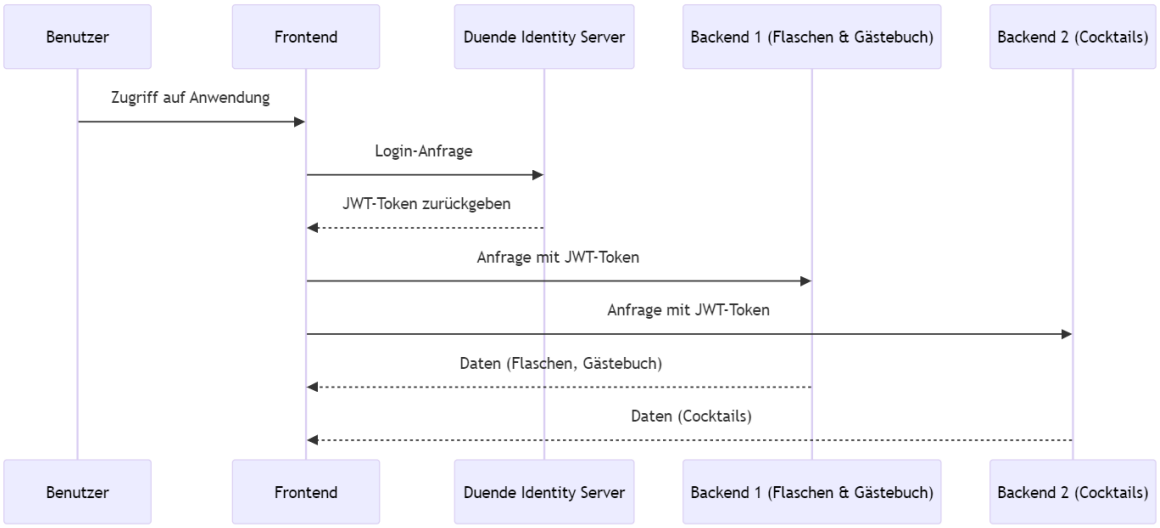
Wir starteten mit einem bereits bestehenden Projekt, das jedoch nicht funktionierte. Gemeinsam analysierten und besprachen wir dessen Aufbau und legten fest, welche Funktionen noch implementiert werden mussten. Danach teilten wir die Arbeit in vier Bereiche auf und begannen, auf Basis der Datenobjekte die benötigten Endpunkte im Backend zu bestimmen. Die bestehende Vorlage erleichterte uns das Einrichten der Grundstruktur erheblich, wodurch wir zügig mit der Planung beginnen konnten und die Endpunkte nahezu in Echtzeit umsetzten.

Überblick des Verteilten Systemes

Unsere Applikation besteht aus einem verteilten System mit zwei spezialisierten Backends, die auf unterschiedliche Funktionsbereiche fokussiert sind: Ein Flaschen-Backend verwaltet die Daten der Bar-Flaschen und Gästebucheinträge, während ein separates Cocktail-Backend die Datenbank der Cocktails bereitstellt. Diese Aufteilung ermöglicht eine effiziente Verarbeitung und flexible Skalierbarkeit, da jedes Backend auf seine spezifische Funktion optimiert ist und unabhängig voneinander weiterentwickelt werden kann.

Im Frontend greift die Anwendung auf die verschiedenen Endpunkte der beiden Backends zu, um Daten abzurufen und anzuzeigen. Zur Absicherung sensibler Daten und Ressourcen wird ein Duende Identity Server genutzt. Der Benutzer

loggt sich zunächst über diesen Identity Server ein und erhält anschließend ein Authentifizierungstoken, das ihm authentifizierte Zugriffe auf die Backends ermöglicht.



Frontend

Unsere Features beinhalten unter anderem: Ein Gästebuch, eine Liste mit verfügbaren Flaschen und eine Liste mit Cocktails.

Das Gästebuch wurde mit der Intension erstellt, eine Möglichkeit für andere Gäste zu bieten Empfehlungen(oder Warnungen) an die nächsten Gäste weiterzugeben. Es werden die letzten zehn Einträge auf unserer Hauptseite angezeigt. Auch kann so eine Bewertung für die Flaschen abgegeben werden.

Bei der Liste der verfügbaren Flaschen kann man verschiedene Daten sowie eine Beschreibung erfassen. Diese werden dann mit der verfügbaren Anzahl an Flaschen angezeigt.

Die Liste der Cocktails wird aus den verfügbaren Flaschen zusammengestellt und wird dafür benutzt übersichtlich Ideen darzustellen. Man kann direkt sehen, was zur Verfügung steht und was nicht.

Flaschen Backend

Das erste Backend verwaltet alle Daten zu Flaschen und zum Gästebuch. Es ist als REST-API in Spring Boot implementiert und enthält Endpunkte zur Durchführung von CRUD-Operationen auf den Flaschen- und Gästebuchdaten. Dieses Backend ist speziell für die Interaktionen der Nutzer mit den physischen Flaschen und den Einträgen im Gästebuch zuständig und bietet eine flexible und skalierbare Schnittstelle.

Architektur

Das Flaschen-Backend ist darauf ausgelegt, flexibel auf Nutzeranfragen zu reagieren und sämtliche Daten zu den Flaschen und Gästebucheinträgen bereitzustellen. Dies erfolgt über einen zentralen Endpunkt, der alle Anfragen entgegennimmt und diese durch eigens entwickelten Code auswertet. Erweiterungen am Backend lassen sich dadurch schnell umsetzen, und der Frontend-Entwickler kann bereits mit Testdaten arbeiten, ohne dass das Backend zunächst vollständig entworfen, entwickelt und deployed werden muss.

File Handling

Das Backend unterstützt die Verwaltung von Bildern und Anhängen, die verschlüsselt gespeichert werden. Mithilfe eines einzigartigen, dateispezifischen Schlüssels wird jede Datei verschlüsselt, bevor sie im Speicher abgelegt wird. Dies gewährleistet, dass Dateien nur durch autorisierte Anfragen abgerufen und entschlüsselt werden können.

Sicherheit

Zur Authentifizierung und Autorisierung der Anfragen nutzt das Backend einen Identity Server. Sensible Endpunkte sind durch Authentifizierung geschützt, sodass unberechtigte Zugriffe verhindert werden.

Cocktail Backend

Das zweite Backend ist speziell für die Verwaltung von Cocktails zuständig und ist in seinem Aufbau ähnlich zur CocktailDB. Es ist als RESTful-API implementiert und ermöglicht den Abruf und die Verwaltung von Cocktaildaten sowie von Zutaten und Zubereitungsanweisungen.

Filtermöglichkeiten

Das Backend bietet eine API-Funktionalität zur Filterung von Cocktails nach Zutaten und Alkoholgehalt. Durch diese Filteroption kann das Frontend gezielt nach Cocktails suchen, die nur mit den verfügbaren Zutaten in der Bar zubereitet werden können.

Datenquelle

Die CocktailDB ist eine öffentliche Datenquelle mit über 600 Cocktails, die von privaten Personen beigesteuert wurden. Diese Website diente uns als Ausgangspunkt und Datenquelle für unser Cocktail-Backend.

Filtermöglichkeiten

Das Backend bietet eine API-Funktionalität zur Filterung von Cocktails nach Zutaten und Alkoholgehalt. Durch diese Filteroption kann das Frontend gezielt nach Cocktails suchen, die nur mit den verfügbaren Zutaten in der Bar zubereitet werden können.

Identity Server

Hier beschreiben wir die Implementierung und Funktionalität eines Identitätsservers, der mithilfe von Duende IdentityServer auf Basis von .NET Core entwickelt wurde. Dieser Server bietet OpenID Connect und OAuth2 für sichere Authentifizierung und Autorisierung. Er authentifiziert Benutzer und gibt ihnen Zugriff auf verschiedene Anwendungen und Services, einschließlich unserer Frontend- und Backend-Services.

Implementation

Der Identitätsserver ist konfiguriert, um Benutzeridentitäten zu verwalten und den Zugriff auf bestimmte Ressourcen zu steuern. Hierfür haben wir Identitätsressourcen, API-Scopes und Clients definiert, die im Code konfiguriert sind.

Identitätsressourcen und API-Scopes

In der Config-Klasse definieren wir Ressourcen, die die Identität der Benutzer verwalten und den Zugriff auf die verschiedenen APIs erlauben. Die Identitätsressourcen bestehen aus den Standard-Scopes für Benutzeridentifikation (openid), Profilinformationen (profile) sowie einer benutzerdefinierten Verification-Ressource zur Überprüfung der E-Mail-Bestätigung. Für den Zugriff auf unsere Backend-Services haben wir API-Scopes definiert, die den Zugriff auf das Cocktail-Backend und das Bottle-Backend regeln.

Clients

Ein Client ist in diesem Fall unser React-Frontend, das mit dem Authorization Code Flow arbeitet und zur zusätzlichen Sicherheit den PKCE-Mechanismus nutzt. Der Client erlaubt Redirect-URLs für Login und Logout, die die Benutzer nach der Anmeldung oder Abmeldung zum Frontend zurückleiten. Zu den zulässigen Scopes gehören openid, profile, cocktail und bottle, damit der Client auf die Identität des Benutzers und die Backend-APIs zugreifen kann.

Benutzer- und Account-Management

Die Benutzerverwaltung wird über ASP.NET Identity in einer SQLite-Datenbank realisiert. Wir haben eine SeedData-Klasse implementiert, die beim ersten Start des Servers die Datenbank initialisiert und Benutzer (alice und bob) mit grundlegenden Profilinformationen anlegt. Der Identitätsserver bietet Anmelde- und Registrierungsfunktionen sowie Token-Management für den sicheren Zugriff auf Ressourcen. Jeder Benutzer verfügt über individuelle Profilansprüche wie name, given_name und family_name.

Login und Logout

Der Server unterstützt den Authorization Code Flow für Anmeldungen und verwendet Redirect-URLs, um den Benutzer nach erfolgreicher Anmeldung oder Abmeldung zurück zur React-Anwendung zu führen.

Ereignisprotokollierung

Um eine genaue Überwachung zu gewährleisten, protokolliert Serilog alle wichtigen Ereignisse, wie Anmeldeversuche oder Fehler, im Terminal. Dies hilft beim Debuggen und ermöglicht es, Fehler schnell zu identifizieren.

Hosting

Für das Hosting unserer Anwendung haben wir auf unser LernMAAS mithilfe von Docker Compose die jeweiligen Komponenten als Container bereitgestellt. Danach haben wir diese direkt mit den IP's verbunden.

Fazit

Unser Projekt zur Entwicklung einer verteilten Systemarchitektur für eine Webanwendung hat gezeigt, wie sich verschiedene Komponenten eines modernen IT-Systems erfolgreich kombinieren lassen, um ein leistungsfähiges und flexibles Nutzererlebnis zu schaffen. Die Entscheidung für eine modulare Struktur mit separaten Backends für Flaschenverwaltung, Gästebucheinträge und Cocktails hat die Verantwortlichkeiten klar aufgeteilt und eine spätere Wartung und Erweiterung vereinfacht.

Durch die Implementierung eines zentralen Duende Identity Servers für die Authentifizierung und Autorisierung konnten wir die Sicherheit unserer Anwendung verbessern und sensible Daten effektiv schützen. Gleichzeitig ermöglicht die Verwendung von JWT-Tokens eine bereits uns bekannte Kommunikation zwischen Frontend und Backends, was die implementation Vereinfachte.

Die agile Zusammenarbeit im Team, unterstützt durch die initiale Planung und eine frühzeitige Einteilung der Aufgabenbereiche, war ein Schlüsselfaktor der Umsetzung. Zudem war die Implementation