Multiuser Applikation

Damien Flury

04. November 2019

Inhaltsverzeichnis

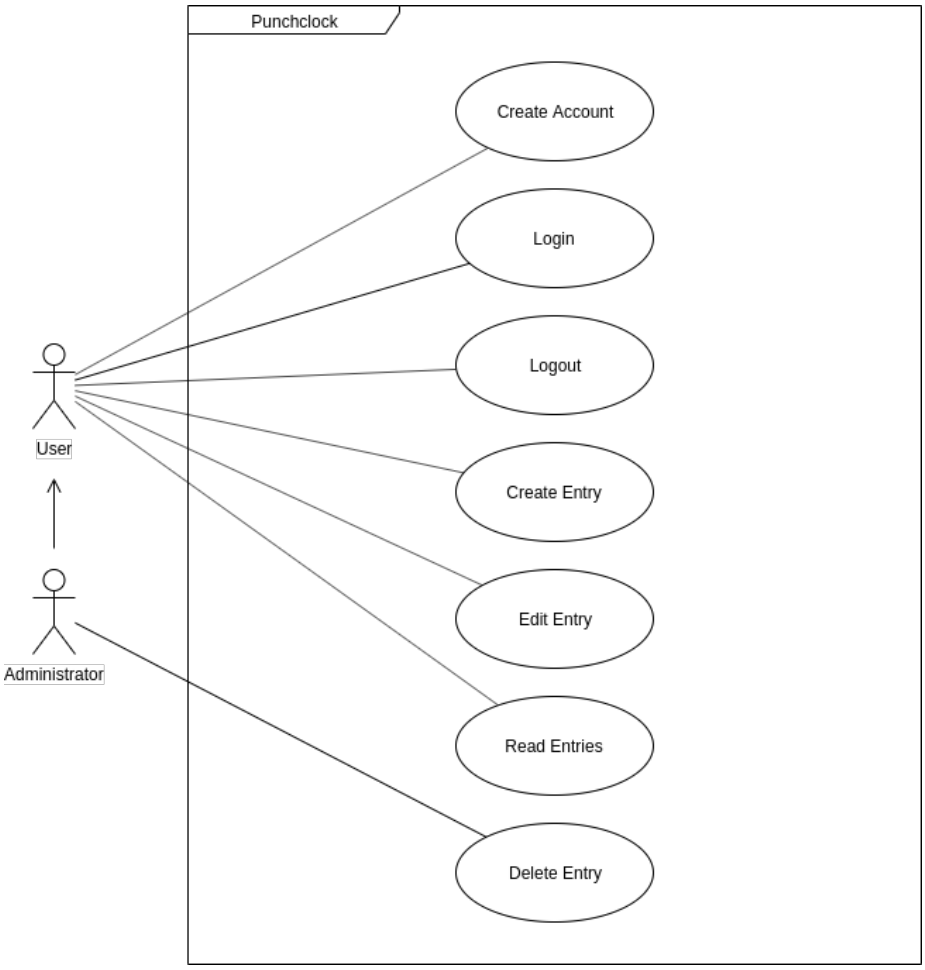

Abbildung 1: Use Case-Diagramm (Entries und Autorisierung)

1 Anwendungsfälle

Ein Anwendungsfalldiagramm finden Sie in Abbildung 1.

1.1 Akteure

Benutzer können sich anmelden, abmelden und Entries verwalten.

1.2 Anforderungen

1.2.1 Create Account

Neue Benutzer können einen eigenen Account erstellen. Dazu brauchen sie eine Email-Adresse und ein Passwort, welches den Anforderungen entsprechen.

1.2.2 Login

Bestehende Benutzer können sich anmelden. Dazu werden wiederum die Email-Adresse, welche sie zur Erstellung verwendet wurde, und das dazugehörige Passwort benötigt. Ein Aktivitätsdiagram dazu finden Sie in Abbildung 2.

1.2.3 Logout

Eingeloggte Benutzer können sich wieder abmelden. Dazu müssen sie zunächst angemeldet sein.

1.2.4 Manage Entries

Eingeloggte Benutzer können neue Entries erstellen, lesen, bearbeiten und wieder löschen.

1.3 Nicht funktionale Anforderungen

1.3.1 Performance

Die Datenbankabfragen müssen möglichst performant ablaufen, um die Benutzerfreundlichkeit nicht einzuschränken.

1.3.2 Design

Einheitliches Design in der Webapplikation, um die Benutzerfreundlichkeit zu optimieren.

1.3.3 Simple Authentifizierung

Die Applikation benötigt Authentifizierung. Die Tokens werden im Frontend gespeichert, sodass der Benutzer sich nicht jedesmal erneut einloggen muss. Für das Login werden lediglich Email und Passwort benötigt.

1.3.4 Sicherheit

Um die bestmögliche Sicherheit zu garantieren, wird HTTPS verwendet und ein ORM verwendet, um Database Injection zu vermeiden.

2 Datenhaltung

Die Applikation besteht aus vier Datenklassen (siehe Abbildung 3):

- ApplicationUser
- Entry
- Position
- Department

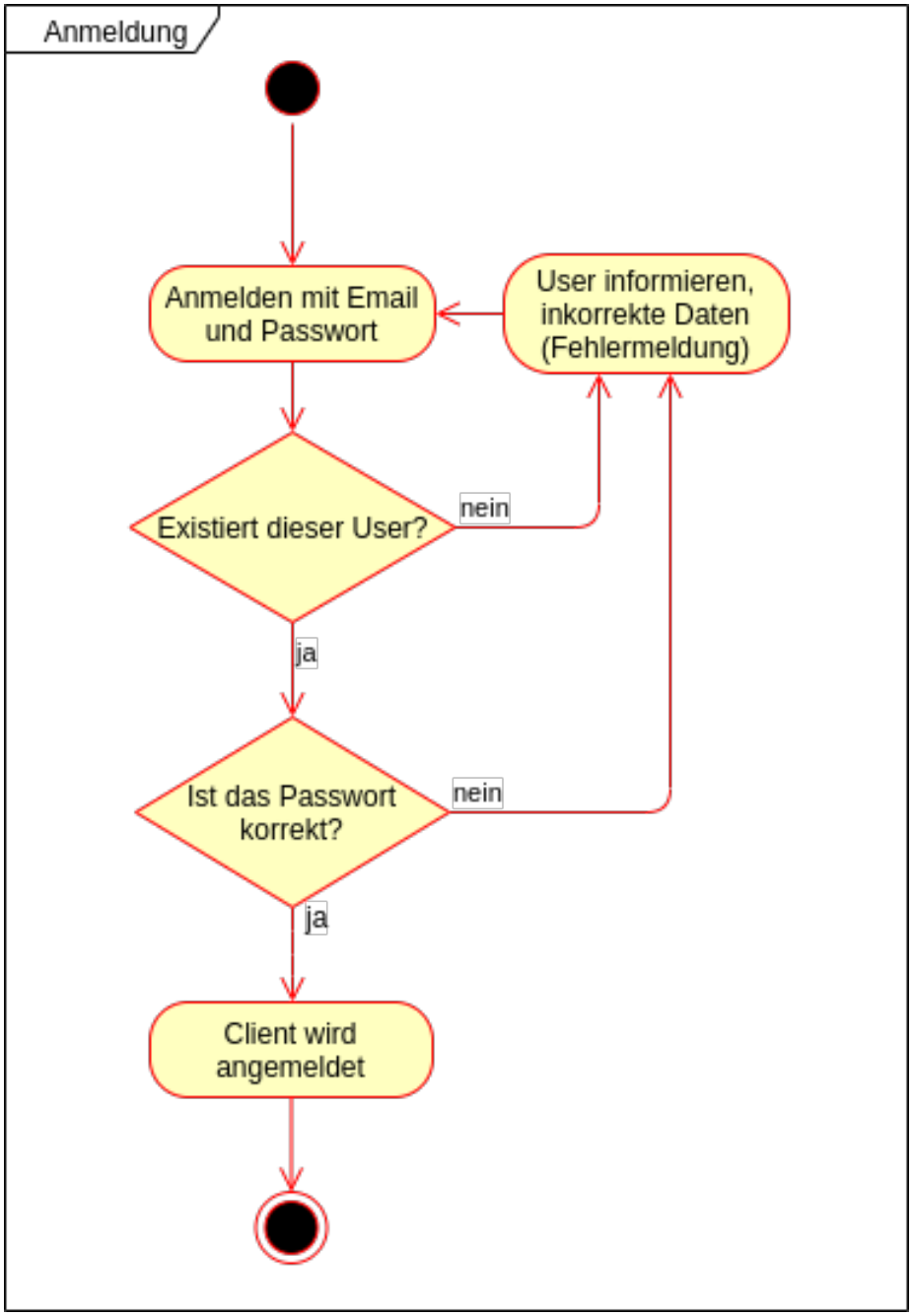

Abbildung 2: Aktivitätsdiagramm (Anmeldung)

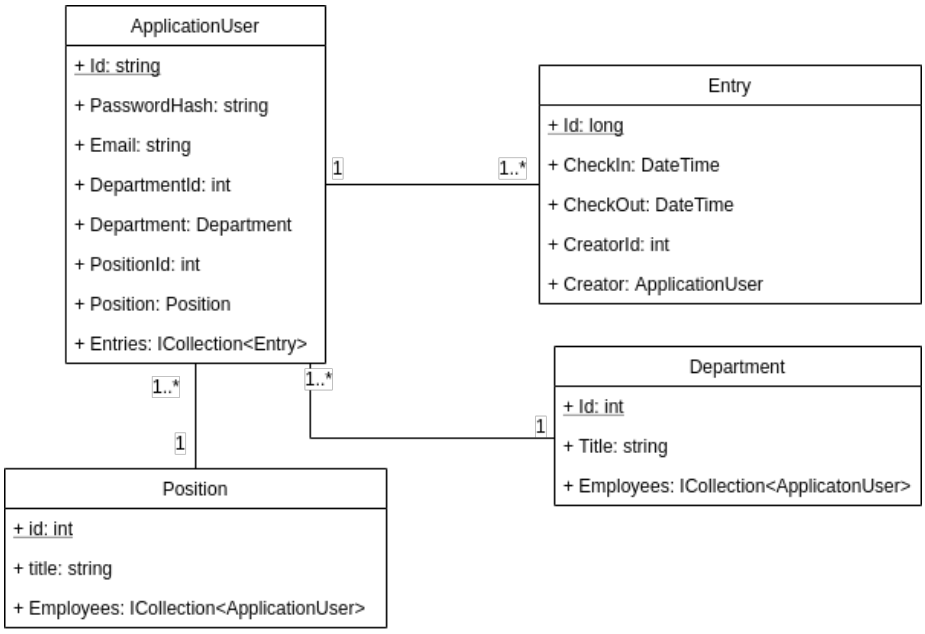

Abbildung 3: Fachklassendiagram

3 Architektur

3.1 Packagediagramm

Ich verwende sechs Namespaces, zwei für die Datenbank und vier für die GraphQL API (siehe Abbildung 4).

3.2 Klassendiagramm

Ich verwende Klassen für die Datenbank-Entitäten und für die GraphQL Types. Ausserdem gibt es eine Startup-Klasse für die Projektkonfiguration (siehe Abbildung 5).

3.3 Deploymentdiagramm

Da wir lediglich auf localhost deployen, entstehen zwei Ports und eine Datenbank, welche als einfaches File eingesetzt wird. Auf Port 5001 wird eine GraphQL Api mit .NET Core eingesetzt, auf Port 3000 entsteht eine React Applikation (siehe Abbildung 6).

4 Testfälle

4.1 Funktionale Testfälle

Eine Übersicht zu den funktionalen Testfällen finden Sie in Tabelle 1.

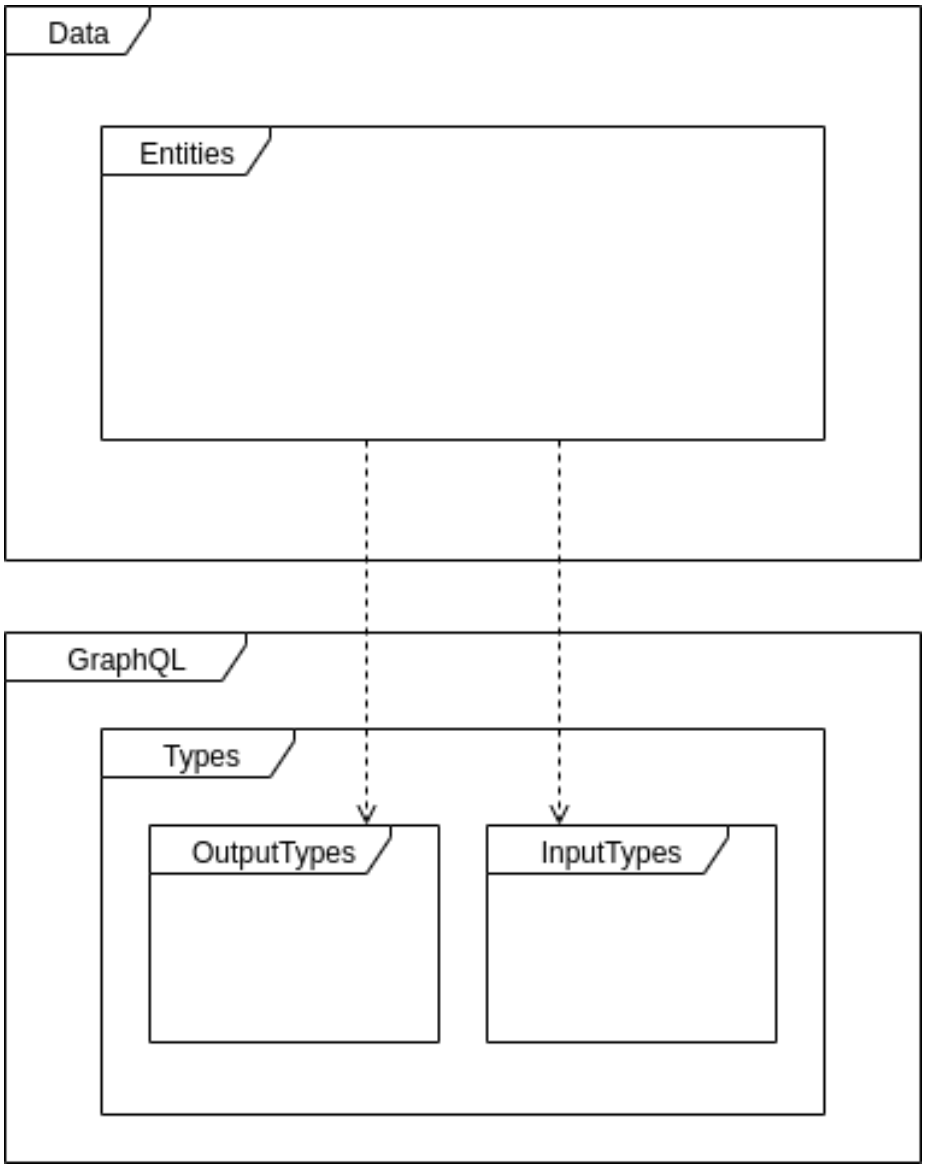

Abbildung 4: Packagediagramm

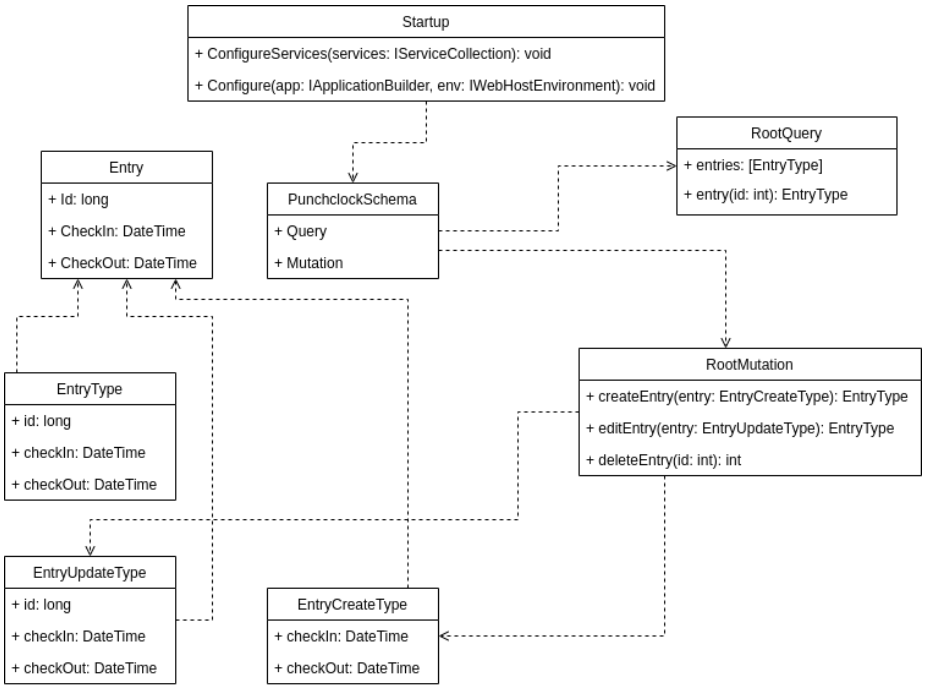

Abbildung 5: Klassendiagramm

Testfall	**Input**	**Output**
User erstellt Account	Email Passwort	User wird angemeldet
User gibt falsche Credentials ein	Email Passwort	Fehlermeldung wird angezeigt User bleibt auf Anmeldepage
User gibt korrekte Credentials ein	Email Passwort	User wird auf Home umgeleitet.
User checkt ein	Current DateTime	User sieht visuelles Feedback
User checkt out	Current DateTime	User sieht visuelles Feedback
User meldet sich ab	-	User wird zurück auf Welcome-Page umgeleitet.

Tabelle 1: Funktionale Testfälle

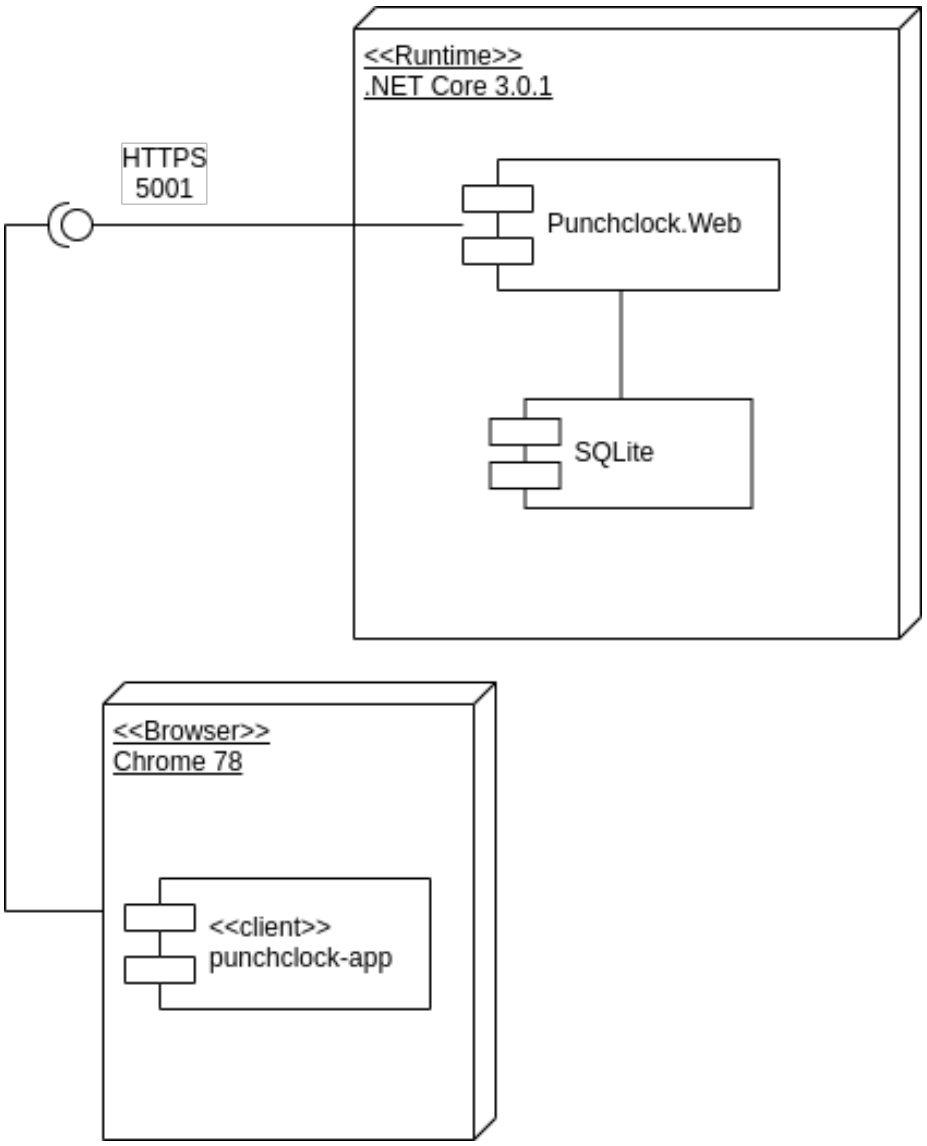

Abbildung 6: Deploymentdiagramm

Testfall	**Result**
Einheitliches Design	Der Benutzer findet sich zurecht.
Einfache Authentifizierung	Der Benutzer braucht nur Email und Passwort, um sich anzumelden.
Performance	Die Entries werden schnell zurückgegeben, Wartezeiten müssen <1 Sekunde betragen.

Tabelle 2: Nichtfunktionale Testfälle

4.2 Nichtfunktionale Anforderungen

Eine Übersicht zu den nichtfunktionalen Anforderungen finden Sie in Tabelle 2.