# Backend: Testbetrieb

## Voraussetzungen

Für den Betrieb des Backend ist die Installation des .NET 5.0 SDKs notwendig. Die Installationsdateien sowie Anleitungen für die Installation via Package-Manager sind unter der URL <https://dotnet.microsoft.com/download/dotnet/5.0> verfügbar.

Zusätzlich ist ein relationales DBMS notwendig, idealerweise PostgreSQL, welches ich in meiner Arbeitsumgebung verwendet habe. Eine Liste der verfügbaren Datenbanken und der dafür notwendigen Packages ist unter folgender URL verfügbar:

<https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli>

Eine erfolgreiche Installation kann in der Shell über den Befehl «dotnet» überprüft werden.

Wie im nachfolgenden Punkt erklärt wird, muss das bestehende Datenbankpaket für den PostgreSQL Server ausgetauscht, sowie ein passender Connection-String hinterlegt werden. Am einfachsten ist es dabei, wie beschrieben die SQLite Variante zu wählen, da hierbei auch keine weitere Installation notwendig wird.

## Optional: Anpassungen für SQLite

Sollte kein passender Datenbankserver vorhanden sein, ist als Alternative das Verwenden von SQLite im Vergleich zu anderen Datenbanken mit wenig Aufwand verbunden. Dazu müssen folgende Änderungen am Projekt durchgeführt werden:

**Compmanager.csproj**: Diese Datei ist im Stammverzeichnis zu finden und beinhält die verwendeten Packages, welche bei der Installation der Applikation mit installiert werden. Hier muss lediglich die Zeile…

<PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL"... Version="5.0.10"/>

… mit folgender Zeile ausgetauscht werden:

<PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite" Version="5.0.10" />

**Appsettings.json:** Diese Datei findet man ebenso im Stammverzeichnis. Hier muss der Knoten «ConnectionStrings 🡪 CompManagerBackend» angepasst werden, beispielsweise ersetzt man…

"Host=localhost;Database=compManager;Username=username;Password=password"

…mit:

"Data Source=CompManagerDatabase.db"

Wobei der Dateiname frei gewählt werden kann. Die Datei wird während der Installation automatisch angelegt.

## Konfiguration

Alle notwendigen Konfigurationen werden in der Datei **Appsettings\_example.json** respektive **Appsettings.json** vorgenommen, welche man im Stammverzeichnis des Backend findet. Es wird angenommen, dass man mit JSON-Dateien vertraut ist, entsprechend findet sich hier lediglich eine Auflistung der einzelnen Schlüssel und passender Werte. Zudem ist es wichtig, dass die vorhandene Datei **Appsettings\_example.json** zum Schluss in **Appsettings.json** umbenannt wird.

|  |  |
| --- | --- |
| **Schlüssel** | **Erklärung** |
| Comment | Enthält Instruktionen zum weiteren Vorgehen, Schlüssel kann im Anschluss gelöscht werden. |
| AppSettings | Alle Einstellungen welche direkt in der Applikation verwendet werden. |
| Secret | Ein möglichst langer String, welcher verwendet wird, um die JWT Token zu signieren, kann frei gewählt werden, je länger, umso sicherer sind die jeweiligen Token. |
| RefreshTokenTTL | Gibt an, wie lange gültig die für die Token-Rotation notwendigen RefreshToken sind, der Wert wird als Ganzzahl angegeben und wird in Tagen interpretiert. |
| JWTTokenTTL | Gibt an, wie lange gültig die JWTToken sind, welche für die Authentifizierung bei API-Anfragen verwendet werden. Wert wird als Ganzzahl angegeben und in Minuten interpretiert. |
| EmailFrom | E-Mailadresse, über welche beim Registrationsprozess und anderen Prozessen entsprechende Nachrichten verschickt werden. |
| SmtpHost | Name des zu verwendenden E-Mail-Servers. |
| SmtpPort | Port, welcher für die Verbindung zum E-Mailserver verwendet wird, Provider und Verbindungsabhängig. Mögliche Werte: 25, 587, 465 und 2525. |
| SmtpUser | Benutzername für die Verbindung zum E-Mail-Server. Oftmals identisch mit «EmailFrom». |

|  |  |
| --- | --- |
| SmtpPass | Für die Verbindung zum E-Mail-Server notwendiges Passwort, eventuell nicht notwendig. |
| FrontEndBaseUrl | URL des Frontend, inklusive verwendetem Protokoll (http/https). Falls Port vom Standard (443 oder 80) abweicht, ebenso anzugeben. Format: protokoll://domain.com:port |
| ConnectionStrings | Übergeordneter Schlüssel, kann so belassen werden. |
| CompManagerBackend | ConnectionString für die Verbindung zur Datenbank. Vorhandener Wert für die Verwendung mit PostgreSQL vorgesehen. Die Werte hinter dem = müssen jeweils angepasst werden. |
| Logging, etc. | Einstellungen fürs Loglevel der Applikation, müssen für den Testbetrieb nicht angepasst werden, für produktiven Betrieb kann man den Wert «Microsoft» auf «Warning» einstellen. |

## Installation

Sofern die vorherigen Schritte erfolgreich durchgeführt werden konnten, sind nun nur noch einige Befehle notwendig, um das Backend zu installieren und zu starten. Die nachfolgenden Befehle müssen im Stammverzeichnis «Backend» in der korrekten Reihenfolge eingegeben werden:

|  |  |
| --- | --- |
| **dotnet ef add migrations initial** | Erstellt aus den vorhandenen Datenbankmodellen die jeweiligen Migrationsdateien, aus welchen Später die Datenbank erstellt wird. Da es beim Einsatz anderer SQL-Varianten hier Unterschiede im Code geben kann, habe ich die Migrationsdateien vorhergehend aus dem Projekt entfernt. |
| **dotnet run** | Startet den Applikationsserver, und erstellt die Datenbank. Ein erfolgreicher Start wird mit einer Ausgabe bestätigt, die ähnlich dem unten zu sehenden Screenshot sein sollte. Die angegebene Adresse, standardmässig <http://localhost:4000> wird später für die Konfiguration des Frontend benötigt. Der angegebene Port 4000 kann eventuell abweichen. |

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Prüfen der Installation

Da die meisten Funktionen des Backend nicht direkt einsehbar sind, kann man über die URL <http://localhost:4000/swagger> die automatisch generierte API-Dokumentation einsehen. Diese gibt Auskunft über die verwendeten Endpoints und Schemen. Eventuell muss der Port der URL auch hier angepasst werden.