Installationsanleitung

Erstellt von: Miguel Friesen, Noah Schwenk	**Überprüft von:** Miguel Friesen, Noah Schwenk, Stephan Braun

Version	Effektiv ab	Beschreibung / Änderungen
1.0	16.05.26	Backend installationsanleitung erstellt
2.0	17.05.26	Frontend Installationsanleitung erstellt
3.0	17.05.26	Finale überarbeitung

1. Allgemeines

Dieses Dokument dient als technische Installations- und Hosting-Anleitung für die Applikation **TaskFlow Engineering**. Es richtet sich in erster Linie an Entwickler, Administratoren und Prüfer, die das System lokal aufsetzen, konfigurieren und ausführen möchten.

Die Anleitung führt Sie Schritt für Schritt durch den gesamten Setup-Prozess. Dies umfasst im Detail:

- Die Installation der zwingend notwendigen Systemvoraussetzungen.
- Das Klonen der jeweiligen Code-Repositories (Frontend und Backend).
- Die lokale Authentifizierung und Rechtevergabe für die angebundenen Google-Cloud-Dienste (Secret Manager, Firestore, Firebase Auth).
- Die korrekte Konfiguration der Umgebungsvariablen.
- Den erfolgreichen lokalen Start des .NET-Backends sowie des React-Frontends.

2. Systemvoraussetzungen

Bevor mit der Einrichtung und dem lokalen Start der Applikation begonnen wird, müssen die folgenden Softwarekomponenten auf dem lokalen Rechner installiert sein:

1. **.NET 9.x SDK** – Erforderlich für das Kompilieren und Ausführen des Backends.
2. **Node.js** (Version 18.x oder höher) & **npm** (Version 9.x oder höher) – Erforderlich für die Ausführung und das Paketmanagement des React-Frontends.
3. **Google Cloud CLI (gcloud)** – Erforderlich für die Authentifizierung und den sicheren Zugriff auf Cloud-Ressourcen.
4. **Git** – Für das Klonen der Quellcode-Repositories.

3. Backend-Installation & Cloud-Konfiguration

Das Backend von TaskFlow Engineering basiert auf .NET 9 und kommuniziert mit der Google Cloud, um sensible Daten sicher abzurufen.

3.1 Voraussetzungen & benötigte Werkzeuge

1. **Repository klonen:** Klonen Sie zunächst den Quellcode des Backends aus dem Git-Repository und wechseln Sie in das Hauptverzeichnis des Projekts.

2. **Google Cloud CLI:** Diese wird zwingend benötigt, da sie den Zugriff auf die im Google Cloud Secret Manager gespeicherten Secrets ermöglicht (insbesondere auf den Brevo API Key).

3. **Firebase Tools (Optional):** Diese Werkzeuge werden ausschließlich für den Firebase Emulator benötigt, welcher in den automatisierten Tests zum Einsatz kommt. Für den normalen lokalen Betrieb des Backends sind sie nicht erforderlich.

3.2 Authentifizierung und IAM-Rollen

Damit das Backend erfolgreich mit den Cloud-Diensten kommunizieren kann, muss eine lokale Authentifizierung durchgeführt werden. Dadurch werden die sogenannten *Application-Default-Credentials* erzeugt, die das Backend automatisch verwendet.

1. Öffnen Sie Ihr Terminal und führen Sie folgenden Befehl aus:

```
gcloud auth application-default login
```

2. Es öffnet sich automatisch ein Browserfenster. Melden Sie sich mit dem Google-Account an, um die Berechtigung zu erteilen.

3. **Wichtig (IAM-Rollen):** Der verwendete Google-Account muss in der Google Cloud Konsole über die notwendigen Berechtigungen verfügen. Ohne diese Rollen kann das Backend zwar lokal gestartet werden, jedoch nicht erfolgreich mit den Cloud-Diensten kommunizieren:

    a. **Secret Manager:** Mindestens die Rolle Zugriffsperson für Secret Manager-Secret.

    b. **Firestore:** Entsprechende Firestore-Rollen für den Datenbankzugriff.

    c. **Firebase Authentication:** Die Berechtigung, um auf den Authentifizierungsdienst zugreifen zu dürfen.

3.3 Konfiguration der Umgebung (appsettings.Development.json)

Für den Zugriff auf den Brevo API Key stellt die Klasse SecretLoader (Implementierung des Interfaces IEnvVariableLoader) die zentrale Komponente dar. Der BrevoEmailService nutzt dieses Interface und erwartet, dass der API Key im Google Secret Manager exakt unter dem Namen BREVO_API_KEY hinterlegt ist.

Nehmen Sie in der lokalen Konfigurationsdatei appsettings.Development.json folgende Anpassungen vor:

1. **ProjectId:** Tragen Sie im SecretLoader bzw. in den Appsettings die korrekte ProjectId des Google-Cloud-Projekts ein, in dem das Secret gespeichert ist.

2. **Firebase Anbindung:** Hinterlegen Sie die richtige Firebase ProjectId, damit das Backend auf Firestore und Firebase Authentication zugreifen kann.

3. **CORS-Freigabe (AllowedOrigins):** Tragen Sie unter AllowedOrigins exakt die URLs ein, unter denen Ihr Frontend lokal (z. B. http://localhost:5173) oder produktiv ausgeführt wird, damit Cross-Origin-Anfragen korrekt verarbeitet werden können.

3.4 Lokaler Start & Testausführung

1. **Backend starten:** Wenn alle Konfigurationen abgeschlossen sind, öffnen Sie Ihr Terminal, navigieren Sie in den spezifischen Projektordner des Backends (in dem die .csproj-Datei und die appsettings.Development.json liegen) und starten Sie den Server mit folgendem Befehl im Verzeichnis [BACKEND_ORDNERNAME]/Task-Weaver-Backend:

```
dotnet run
```

2. **Automatisierte Tests ausführen:** Um die Tests innerhalb einer vollständig isolierten Umgebung auszuführen (in der Firestore und Firebase Authentication simuliert werden, ohne produktive Datenbanken anzusprechen), nutzen Sie folgenden Befehl im Verzeichnis [BACKEND_ORDNERNAME]:

```
firebase emulators:exec 'dotnet test Task-Weaver-Backend.sln --configuration Release --logger "junit;LogFileName=TestResults.xml"'
```

3.5 Optionale Anpassungen (Architektur & alternative Provider)

Das Backend ist so modular aufgebaut, dass Kernkomponenten ohne Änderung der Kernlogik ausgetauscht werden können. Alle Bindungen werden über das Singleton-Pattern in der Datei ServiceCollectionExtensions.cs verwaltet:

- **Alternative Authentifizierung & Datenbank:** Falls Sie nicht Firebase/Firestore nutzen möchten, können Sie die bestehenden Repository-Interfaces beibehalten, neue Klassen für Ihre alternative Datenquelle schreiben und diese in der ServiceCollectionExtensions.cs binden.

- **Alternativer Secret-Loader:** Wenn Sie keine Google-Secrets verwenden möchten, erstellen Sie eine eigene Klasse, die das Interface IEnvVariableLoader implementiert (z. B. zum Auslesen lokaler Dateien), und registrieren Sie diese.

- **Alternativer E-Mail-Dienst:** Möchten Sie nicht Brevo nutzen, erstellen Sie eine neue Klasse basierend auf dem Interface IEmailService für einen beliebigen anderen Provider und binden diese in den Extensions.

4. Frontend-Installation & Start

Das Frontend ist ein React-Projekt. Zum lokalen Aufsetzen werden **Node.js** und **npm** benötigt.

4.1 Voraussetzungen & Installation

1. Klonen Sie das Frontend-Repository über Ihr Terminal.
2. Navigieren Sie in den erstellten Frontend-Projektordner:

cd [FRONTEND_ORDNERNAME] aktuell

3. Installieren Sie alle im Projekt definierten npm-Pakete und Abhängigkeiten mit dem Befehl:

npm install

4.2 Konfiguration der Umgebungsvariablen (.env)

Das Frontend benötigt zur Laufzeit verschiedene Konfigurationswerte, um mit Firebase und dem Backend kommunizieren zu können. Diese Werte müssen manuell in .env-Dateien im Hauptverzeichnis des Frontends gespeichert werden.

Hinweis: Falls das Projekt mit einer völlig neuen Firebase-Instanz verknüpft werden soll, öffnen Sie die Firebase Console, wählen Sie das Projekt aus und kopieren Sie die Keys unter „Settings“ -> „General“.

Lokale Entwicklung

Erstellen Sie für die lokale Entwicklung manuell eine Datei namens .env (oder .env.development) im Hauptverzeichnis mit folgendem Inhalt:

Code-Snippet
```
VITE_FIREBASE_API_KEY="AIzaSyCiKmYhNMWkceEN1SSuKt8Ho5xD7TukOrM"
VITE_FIREBASE_AUTH_DOMAIN="task-weaver-eba38.firebaseapp.com"
VITE_FIREBASE_PROJECT_ID="task-weaver-eba38"
VITE_FIREBASE_STORAGE_BUCKET="task-weaver-eba38.firebasestorage.app"
VITE_FIREBASE_MESSAGING_SENDER_ID="638307930135"
VITE_FIREBASE_APP_ID="1:638307930135:web:50ea8791eff5e4ae920c52"
VITE_BACKEND_URL=http://localhost:5270
```

Produktionsumgebung

Für den finalen Produktionsbuild wird stattdessen eine Datei namens .env.production verwendet:

Code-Snippet
VITE_FIREBASE_API_KEY="AIzaSyCiKmYhNMWkceEN1SSuKt8Ho5xD7TukOrM"
VITE_FIREBASE_AUTH_DOMAIN="task-weaver-eba38.firebaseapp.com"
VITE_FIREBASE_PROJECT_ID="task-weaver-eba38"
VITE_FIREBASE_STORAGE_BUCKET="task-weaver-eba38.firebasestorage.app"
VITE_FIREBASE_MESSAGING_SENDER_ID="638307930135"
VITE_FIREBASE_APP_ID="1:638307930135:web:50ea8791eff5e4ae920c52"
VITE_BACKEND_URL=https://backend-git-861098134684.europe-west3.run.app

4.3 Ausführungs- und Testbefehle

Sobald die Umgebungsvariablen definiert sind, stehen Ihnen im Terminal folgende npm-Befehle zur Verfügung:

- **Entwicklungsmodus starten:** npm run dev
- **Produktionsbuild erzeugen:** npm run build
- **Automatische Frontend-Tests ausführen:** npm run test