NEXTGEN
- DEVELOPMENT -

Designbeschreibung

NextGen Development

Version 1.3

05. April 2025

Teammitglieder:

Julian Lachenmaier
Ayomide Adeniyi
Din Alomerovic
Cedric Balzer
Rebecca Niklaus

Verantwortlich für dieses Dokument:

Din Alomerovic

Allgemeines

Die Anwendung, die in diesem Dokument beschrieben wird, ist der Prototyp eines Event-Management-Systems. Die Software soll anderen Unternehmen das Erstellen und Verwalten von Events und Teilnehmern vereinfachen.

Produktübersicht

In diesem Abschnitt wird der Funktionsumfang des Prototyps kurz vorgestellt.

- Event konfigurieren: Der Organisator ist dazu in der Lage Events innerhalb seiner Organisation zu gestalten. Dazu kann er sich Vorlagen bedienen und eigene Flows erstellen.
- Auf der Plattform registrieren: Ein Benutzer kann sich mit seiner E-Mail-Adresse der Organisation auf der Plattform registrieren. Nach seiner Registrierung kann der Benutzer seiner Organisation beitreten und alle Events einsehen.
- Organisatoren einladen: Ein Organisator kann innerhalb seiner Organisation weitere Organisatoren einladen.
- Zu einem Event an-/abmelden: Ein Benutzer kann sich selbstständig zu einem Event innerhalb seiner Organisationen an- und abmelden.
- Flows konfigurieren: Ein Organisator kann Flows für die Organisation oder eventspezifisch erstellen.

Grundsätzliche Struktur- und Entwurfsentscheidungen

In diesem Abschnitt wird eine Beschreibung des Gesamtsystems erstellt, wobei zunächst auf die gewählte Architektur und anschließend auf den gewählten Technologie Stack eingegangen wird.

Für die Entwicklung des Prototyps haben wir uns für eine schichtbasierte Architektur entschieden. Der Hauptgrund dafür war die schnelle und unkomplizierte Umsetzbarkeit. Da uns nur begrenzt Zeit zur Verfügung stand, erwies sich dieser Ansatz als ideal. Die Architektur gliedert sich in die folgenden Ebenen: Frontend, Backend, Datenbank und Filestore.

Frontend

Für das Frontend haben wir uns für die Entwicklung mit der JavaScript-Bibliothek React.js entschieden. Diese Entscheidung basiert auf drei Faktoren: Unser Team verfügt über React-Expertise, mehrere Mitglieder haben bereits JavaScript- und React-Erfahrung, und React ist die am weitesten verbreitete Frontend-Technologie.

Backend

Für das Backend haben wir uns für die Entwicklung mit dem Web-Application-Framework ASP.NET entschieden. Diese Entscheidung beruht darauf, dass alle Mitglieder unseres Teams bereits über C#-Kenntnisse verfügen und einzelne bereits mit ASP.NET gearbeitet haben. Zu dem ist ASP.NET Platz drei der serverseitigen Technologien [1].

API

Als Schnittstelle zwischen dem React.js-Frontend und dem ASP.NET-Backend nutzen wir eine RESTful-API, welche als OpenAPI Spezifikation vorliegt. Ausschlaggebend für diese Entscheidung

waren die vorhandenen Kenntnisse im Team zu RESTful APIs, deren große Verbreitung sowie die zu hohe Komplexität alternativer Lösungen für unseren speziellen Anwendungsfall.

Datenbank

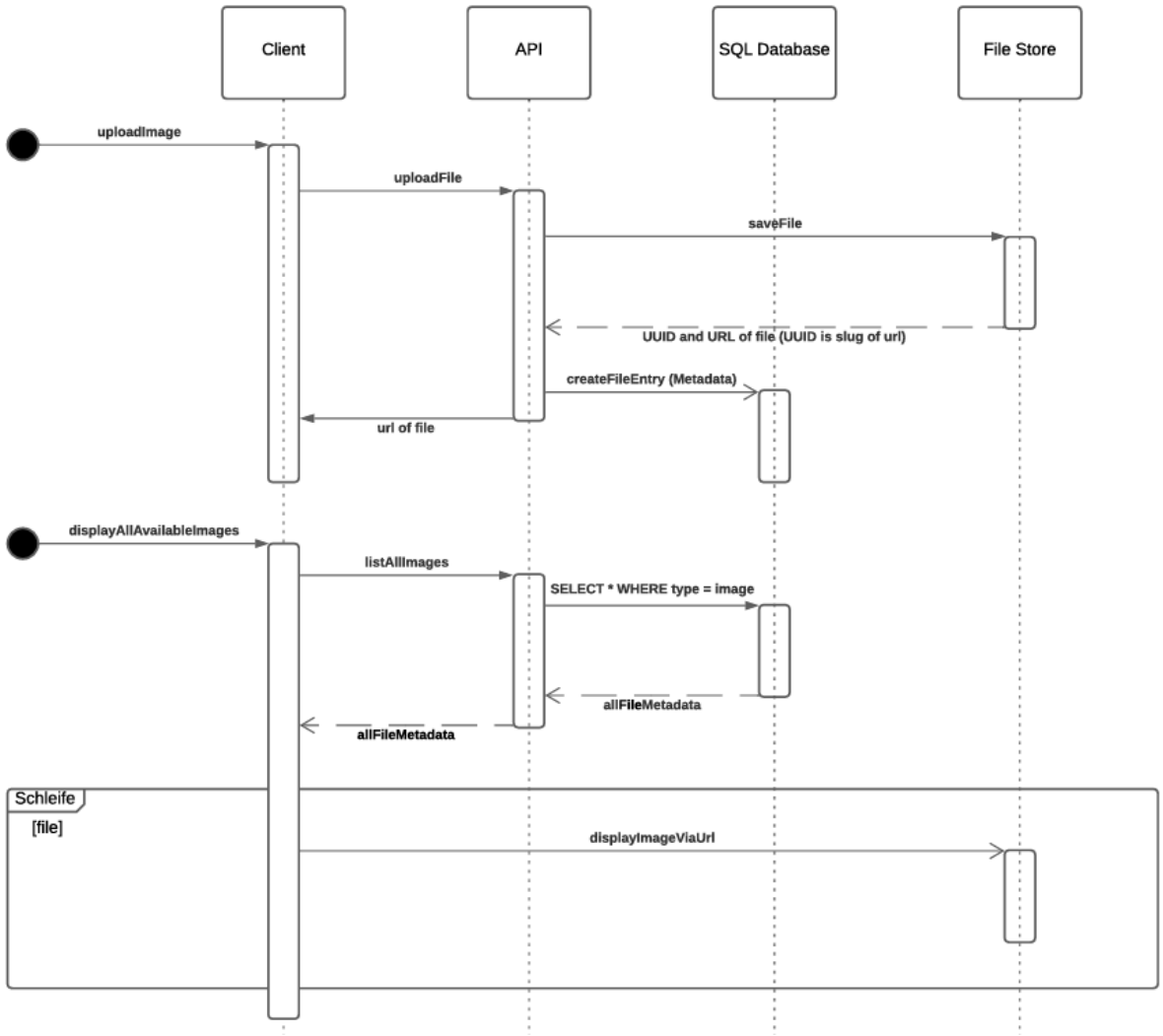
Zur Speicherung benötigter Daten wie Benutzer- oder Eventinformationen verwenden wir eine PostgreSQL-Datenbank. Die Entscheidung fiel auf PostgreSQL, da Teammitglieder bereits Erfahrung damit hatten und die Datenbank durch Flexibilität, Kostenfreiheit und ihren Open-Source-Charakter überzeugt.

FileStore

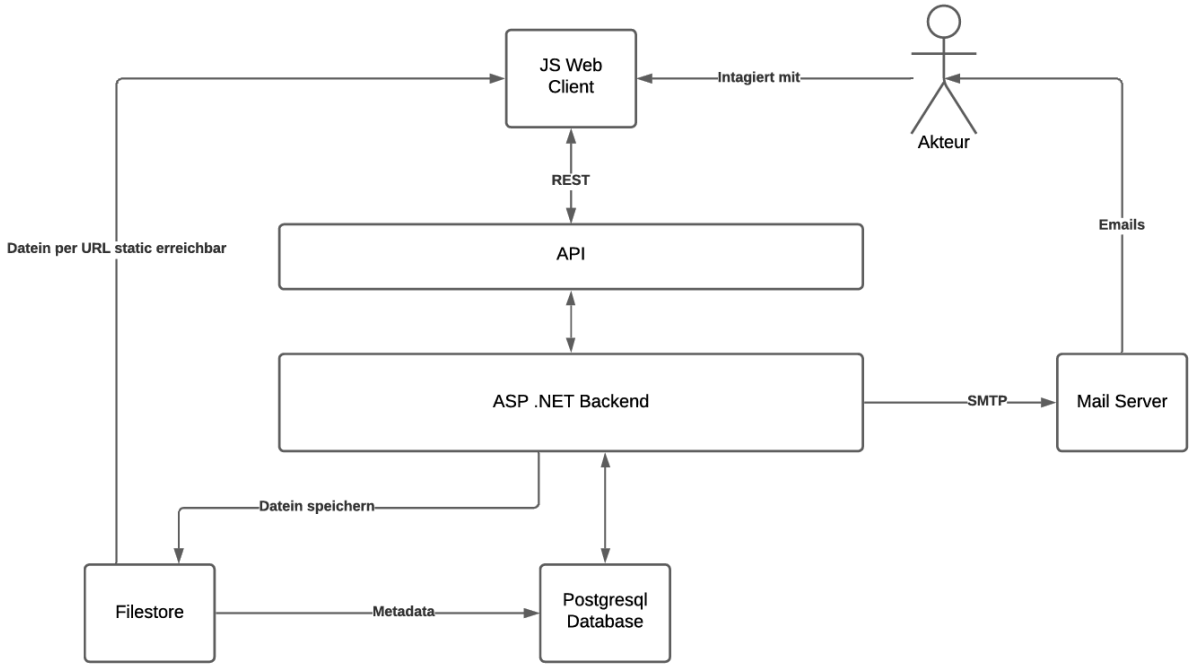
Um die von den Usern hochgeladenen Dateien zu speichern, verwenden wir einen File Store Service, auf dem die Daten gespeichert werden. In unserer Datenbank wird dann nur die URL bzw. der Pfad gespeichert.

Diese Art der Speicherung bietet uns eine skalierbare, sichere und wartungsarme Lösung für nutzergenerierte Dateien. Sie entlastet unsere Infrastruktur, ermöglicht den effizienten Zugriff über temporäre Links und trennt sauber Anwendungs- und Dateiverwaltung.

Folgende Abbildung soll den Speichervorgang einer Datei visualisieren.



Eine Übersicht über die beschriebenen Komponenten soll folgende Ansicht bieten.



Grundsätzliche Struktur- und Entwurfsentscheidungen der einzelnen Pakete/Komponenten

Frontend:

- React Context Provides für User und selected org
- Aufsplittung in möglichst viele reuseable components
- Verwendung von populärer shadcn compnents library, da diese die ownership of code uns gewährleistet und dardurch, dass sie auf radix basiert auch solide ist

Backend:

Program.cs: Ist der Einstiegspunkt der Anwendung. Hier wird die ASP.NET App konfiguriert.

ApplicationDbContext.cs: Dient als zentrale Schnittstelle zwischen der Anwendung und der Datenbank. Sie ermöglicht es, Datenbankabfragen und -änderungen direkt in C# durchzuführen, ohne SQL schreiben zu müssen.

Models: Ordner in dem die Model-Klassen gespeichert sind. Ein Model repräsentiert ein Datenmodell in der Datenbank.

DTOs: Die Klassen, die in diesem Ordner enthalten sind, dienen zum Austausch von Daten. Man kann so Daten definieren, die eine Teilmenge der Model-Daten enthalten. So kann man bestimmte Felder, welche Daten enthalten, die nicht weitergeleitet werden sollen, entfernen.

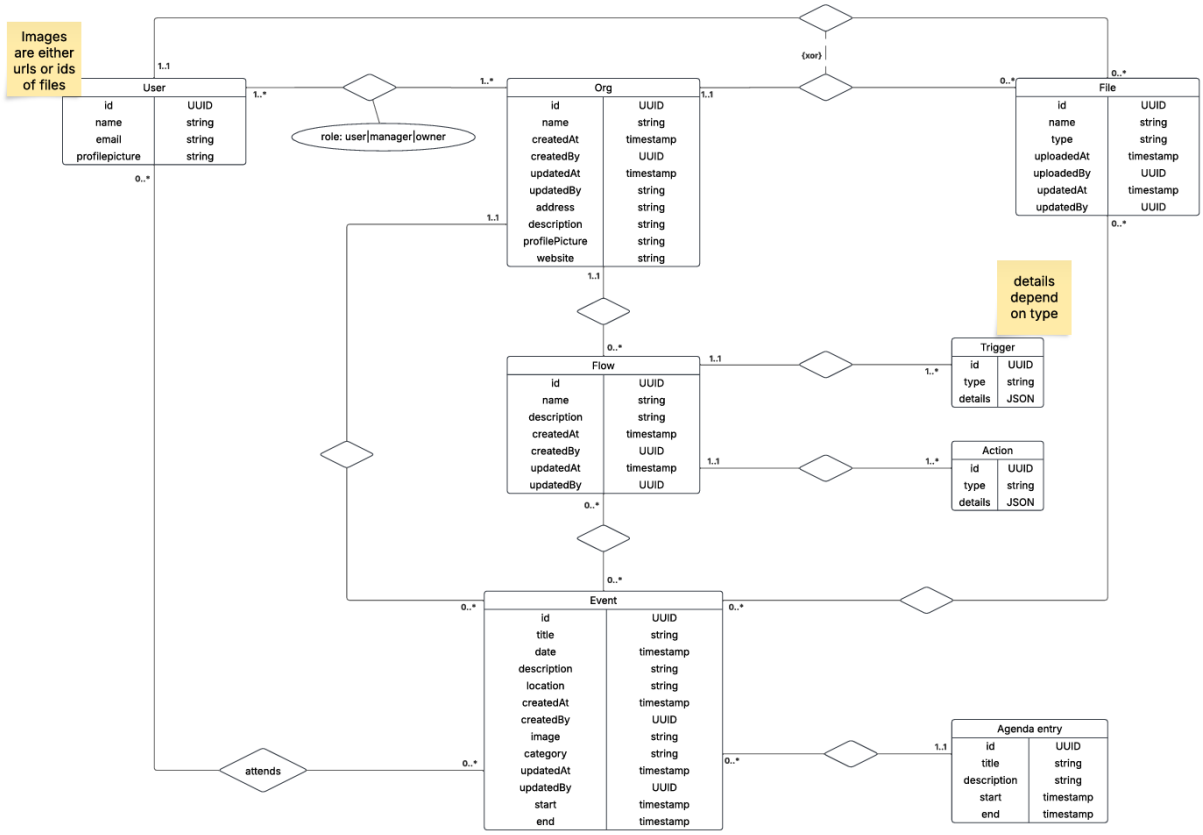
Controllers: Ornder in dem die einzelnen Controller enthalten sind. Ein Controller kümmert sich in ASP.NET um die http Anfragen. Hier werden die einzelnen http Endpoints und die jeweilige Aktion, die man daraufhin anwenden soll, definiert.

Repository Layer: Anstatt die Logik innerhalb der einzelnen Controller zu definieren benutzen wir Interfaces, die die jeweiligen Methoden beinhalten. Der Controller ruft damit nur noch Methoden auf, die innerhalb der Interfaces definiert werden. Die eigentliche Logik befindet sich dann innerhalb des Repository Ordners. Die Klassen, die darin enthalten sind, implementieren die jeweiligen Interfaces.

Das sorgt für die Einhaltung des „Separation of Concerns“ Prinzips, da sich die Controller nur noch um die Handhabung von http-Anfragen kümmern müssen. Ein Repository kann dabei von mehreren Services genutzt werden was Redundanzen vermeidet.

Datenbank:

Folgendes E/R-Modell soll einen Überblick über die in der Datenbank verwendeten Tabellen und Attribute geben.



Quellen

[1] Usage Statistics and Market Share of Server-side Programming Languages for Websites, April 2025 Abgerufen am 05.04.2025