Projekt Spezifikation

Projektname: FridgeBuddy

1. Thema

Plattform für den Austausch von Lebensmitteln zur Reduzierung von Food Waste und Förderung von günstigen Lebensmitteln

2. Umfeld, Ausgangslage

Gemäss Bundesamt für Umwelt BAFU werden in Schweizer Haushalten pro Jahr 778‘000 Tonnen Lebensmittel verschwendet. Dies entspricht ungefähr einem Drittel des gesamten Food Waste in der Schweiz.

Die Schweizer Haushalte verschwenden dadurch pro Jahr über 5 Milliarden Franken. Pro Kopf werfen wir jährlich ungefähr 100kg Lebensmittel im Wert von 620.- in die Tonne.

Diverse Projekte fördern bereits den Austausch von Lebensmitteln, beispielsweise in öffentlich zugänglichen Kühlschränken (https://www.madamefrigo.ch/).

FridgeBuddy setzt sich nun zum Ziel, den Austausch von Lebensmittel auf einer Online-Plattform zu unterstützen.

3. Aufgabenstellung & Ziel des Projektes

Das primäre Ziel von FridgeBuddy ist, Lebensmittelspender und Konsumenten zusammen zu bringen. Spender, welche ein Überangebot an Lebensmittel haben, können Lebensmittel als Spende in FridgeBuddy publizieren. Interessierte können die Lebensmittel online reservieren und die Kontaktdaten austauschen.

Sowohl Spender als auch Konsumenten profitieren von der Plattform. Die Abnehmer der Lebensmittel profitieren finanziell von günstigen Lebensmitteln, wobei die Spender den eigenen Fussabdruck optimieren können.

4. Scope des Projektes

Lebensmittel spenden
Die Spender können Lebensmittel anbieten. Hierfür können diverse Eigenschaften erfasst werden, wie beispielsweise die folgenden:

- Produkt
- Lebensmittelart (Gemüse/ Früchte, Fleisch, Milchprodukte, Getreide, etc.)
- Beschreibung
- mindestens haltbar bis oder zu verbrauchen bis
- Label (Bio, IP-Suisse, Demeter, etc.)
- Herkunftsland
- Foto
- Abholort
- Menge
- Essgewohnheit: vegetarisch, vegan, laktosefrei, etc.

Lebensmittel suchen
Die Konsumenten können Lebensmittel nach diversen Kriterien suchen.
Filterkriterien können beispielsweise die Art der Lebensmittel oder das Herkunftsland sein.

Lebensmittel reservieren
Die Konsumenten können Lebensmittel reservieren und bekommen Kontaktdaten angezeigt.

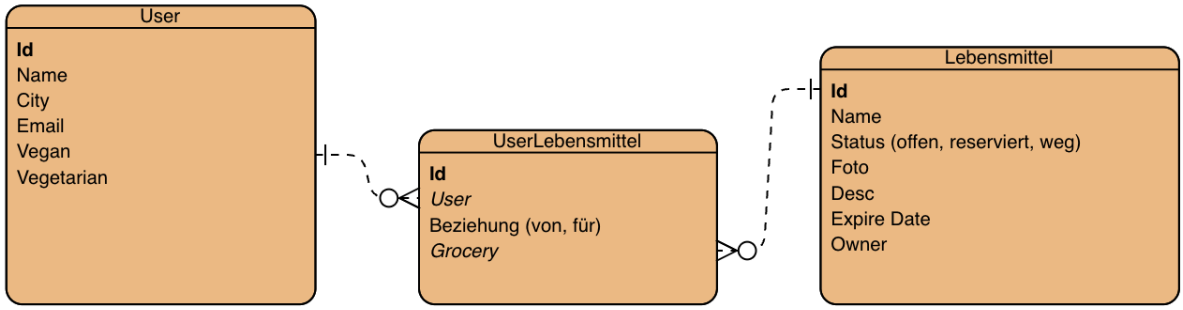
Login / Registrierung
Das Backend kann zwar User registrieren aber wir werden im Frontend nur vorgefertigte User verwenden und keinen echten Login-Mechanismus implementieren.

Usecase-Diagramm

5. Grob-Architektur und Design

5.1 Technischer System-Kontext

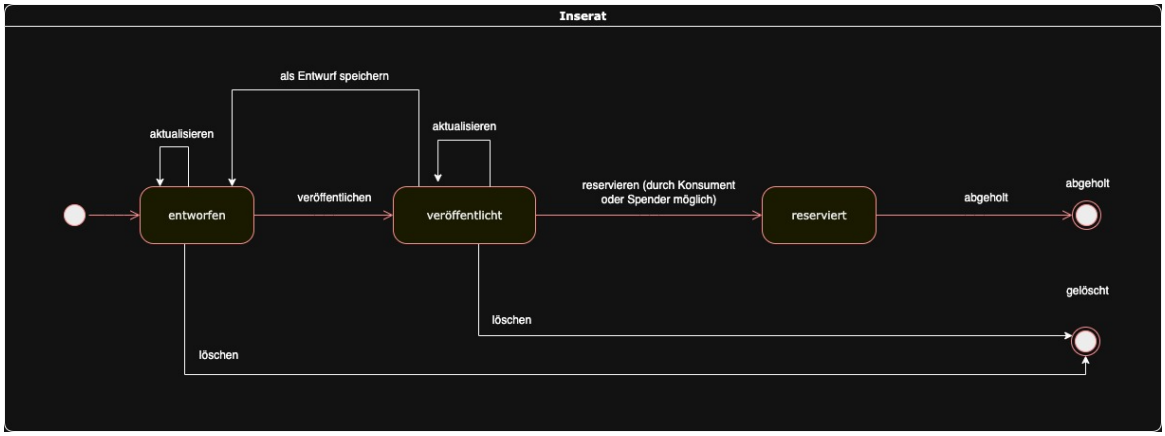
Datenmodell



Ein User kann mehrere Lebensmittel haben und ein Lebensmittel kann zu mehreren Usern gehören. Damit kann ein User ein Lebensmittel erstellen und ein anderer kann es reservieren. Der aktuelle Status wird in der Zwischentabelle gespeichert.

Statusdiagramm

Ein Lebensmittel kann fachlich betrachtet im Lebenszyklus die untenstehenden Stati durchlaufen. Wichtig ist, dass eine Reservation nicht mehr rückgängig gemacht werden kann.



Statemanagement

Die Daten für “Lebensmittel suchen” aus dem Backend geholt. Das funktioniert mit Pagination und Filtern. Wird ein Lebensmittel angeklickt, wird mit der ID dieses Lebensmittels eine neue Abfrage gemacht, um alle Daten zu bekommen. Diese werden auf der Detail-Seite angezeigt. Reserviert der Benutzer das Lebensmittel, dann wird eine weitere Abfrage gesendet, um den Status zu aktualisieren.

Von “Meine Produkte bearbeiten” wird erst eine Abfrage ans Backend gesendet, um alle eigenen Lebensmittel zu erhalten. Bei einer Änderung wird eine Abfrage ans Backend gesendet, um dieses eine Lebensmittel zu aktualisieren.

Wir werden wahrscheinlich überall Optimistic Updates machen. Der Grund ist, dass die Daten nicht sehr kritisch sind und es dem User ein angenehmeres Erlebnis bietet. Aufgrund der Grösse des Projektes verzichten wir zudem auf eine Statemanagement-Library wie ngrx oder Akita und es wird keine client-seitige Persistenz und Offline-Fähigkeit implementiert.

5.2 Grob-Architektur der Lösung

Deployment Backend

Das Deployment des Django Rest Frameworks für unser Projekt erfolgt über die Plattform Render, wobei eine persistente Festplatte sowie eine PostgreSQL-Datenbank genutzt werden. Zunächst wurde das Projekt lokal entwickelt und alle Abhängigkeiten in der requirements.txt dokumentiert. Für das Deployment wird die Datei “backend.py” entsprechend angepasst.

Die Konfiguration der Umgebungsvariablen für die Datenbankverbindung und andere sensible Informationen erfolgte über die Render-Einstellungen, um die Sicherheit zu gewährleisten. Die persistente Festplatte wurde eingerichtet, um statische und Medien-Dateien zu speichern, wobei die entsprechenden Pfade in den Django-Einstellungen angegeben wurden.

Dieser Prozess stellt sicher, dass unsere Anwendung effizient und zuverlässig in einer Produktionsumgebung läuft. Um Änderungen zu deployen, wird der lokale Code zu GitHub “gepushed” was automatisch zu einem neuen Build und Migration in Render führt.

Deployment Frontend

Das Frontend wird wahrscheinlich auf Firebase oder Netlify gehostet. Das Frontend sollte kein großer Aufwand sein zu deployen.

5.3 API-Endpoints

API	Endpoint	Beschreibung
Obtain JWT Token	POST /api/token/	Eingabe von Username & Passwort, Erhalt von access & refresh token
Refresh JWT Token	POST /api/token/refresh/	Erhalt von neuem Access Token (Refresh token senden)
User registrieren	POST /users/v1/register/	Neuen User Registrieren
Userprofil bearbeiten	POST /users/v1/buddy-profile/	Profil updaten von eingeloggtem User
Lebensmittel suchen	GET groceries/api/v1/groceries/	Gibt alle Items zurück: Filterbar nach Name, Status und Auslaufdatum
Lebensmittel erstellen	POST groceries/api/v1/groceries/	Neues Lebensmittel erstellen
Managen von eigenen Lebensmitteln	GET, PUT/PATCH, DELETE /groceries/v1/my-groceries/<int:pk>/	Bearbeiten von eigenen Lebensmitteln (Erlaubt nur bei Lebensmitteln, die einem “gehören!)
Lebesmittel anfragen	POST /groceries/v1/groceries/<int:pk>/request/	Status von Lebensmittel ändern (nur logged in Users)
Eigene Lebensmittel anzeigen	GET groceries/v1/my-requests/	Eingeloggter User bekommt alle eigenen Lebensmittel angezeigt

5.4 Übersicht Screens

https://miro.com/app/board/uXjVli8QpK0=/

5.5 Wahl der Technologie

Das Frontend wird mit Angular umgesetzt. Angular bietet eine solide Grundlage für strukturierte Webanwendungen mit klarer Trennung von Komponenten, Datenlogik und UI. Für FridgeBuddy ist Angular besonders geeignet, weil es skalierbar ist. Falls wir in Zukunft fridgeBuddy um weitere Funktionen ergänzen möchten, dann können diese Features schrittweise einbauen, ohne dass die Technologie grundlegend angepasst werden muss. Zudem unterstützt Angular Routing und Typensicherheit direkt. Schlussendlich profitieren wir von einer grossen Community im Angular Ökosystem.

Das Backend wird mit Python umgesetzt, weil im Team bereits “Django Rest Framework” – Wissen vorhanden ist und weil Django “out of the box” sehr viel bietet. Von Django kommt ein Admin-Panel, eine Browsable API fürs Entwicklen und Testen und ein Datenmodell, welches z.B. User bereits beinhaltet.

6. Studierende

Luca Brandt
Miro Rutscho
Dominique Müller