



**«Xung anstatt Pfung»**

**«mix your own meal»**

**Informationen für die Jury**

**Inhalt**

[1 Zugänge 3](#_Toc50250495)

[1.1 Aktueller Stand des Sourcecodes 3](#_Toc50250496)

[1.2 CI/CD Pipeline 3](#_Toc50250497)

[1.3 Production 3](#_Toc50250498)

[2 Fachliche Aspekte 4](#_Toc50250499)

[2.1 Ausgangslage 4](#_Toc50250500)

[2.2 Lösungsansatz 4](#_Toc50250501)

[2.3 Umgesetzte Use Cases 5](#_Toc50250502)

[2.4 Abgrenzungen 5](#_Toc50250503)

[3 Technische Aspekte 6](#_Toc50250504)

[3.1 Architekturentscheide 6](#_Toc50250505)

[3.2 Bausteinsicht 7](#_Toc50250506)

[3.2.1 Frontend 7](#_Toc50250507)

[3.2.2 Backend 7](#_Toc50250508)

[3.2.3 Recommender 7](#_Toc50250509)

[3.2.4 Persistenz 7](#_Toc50250510)

[3.2.1 EventHub 8](#_Toc50250511)

[3.3 Verteilungssicht 9](#_Toc50250512)

[3.3.1 Toolchain 9](#_Toc50250513)

[3.3.2 Frontend 9](#_Toc50250514)

[3.3.3 Backend 9](#_Toc50250515)

[3.3.4 Recommender 9](#_Toc50250516)

[3.3.5 Persistenz 9](#_Toc50250517)

[3.3.6 EventHub 9](#_Toc50250518)

[3.4 Implementation 10](#_Toc50250519)

[4 Ausblick 10](#_Toc50250520)

[5 Verzeichnisse 11](#_Toc50250521)

# Zugänge

## Aktueller Stand des Sourcecodes

* <https://github.com/baernhaeckt> [Organisation]
* <https://github.com/baernhaeckt/Backend2022> [Backend Source]
* <https://github.com/baernhaeckt/calorie_estimation2022> [AI Image Nutri-Score Source]
* <https://github.com/baernhaeckt/Frontend2022> [Frontend Source]
* <https://github.com/baernhaeckt/Misc2022> [Documentation, Presentation, Misc]

## CI/CD Pipeline

CI/CD ist mit GitHub Actions implementiert. Das Deployment wird von Azure selbst orchestriert.

* <https://github.com/baernhaeckt/Backend2022/actions> [Backend Source]
* <https://github.com/baernhaeckt/calorie_estimation2022/actions> [AI Image Nutri-Score Source]
* <https://github.com/baernhaeckt/Frontend2022/actions> [Frontend Source]

## Production

* <https://polite-water-0ef553c03.1.azurestaticapps.net/> [Frontend]
* <https://mixmeal-backend.azurewebsites.net/swagger/index.html> [Backend – Swagger UI]
* <https://mixmeal-estimator.azurewebsites.net/docs> [Estimator Swagger]

# Fachliche Aspekte

Dieses Kapitel gibt einen kurzen Überblick über die fachlichen Aspekte und den Umfang der Lösung.

## Ausgangslage

Das Interesse im Bereich health food steigt. Die Leute wollen qualitativ gute Ware essen. Diesem Bedürfnis will die feelfree GmbH mit health food take-away Filialen namens „freshfood-station“ gerecht werden. Die erste Filiale wird im Sommer 2023 eröffnet. Im Rahmen der Challenge soll für die feelfree GmbH ein App erstellt werden, mit welcher ein Benutzer sogenannte Menüs zusammenstellen, bestellen und später abholen kann.[[1]](#footnote-1)

## Lösungsansatz

Wir haben bei der Entwicklung der Lösung zwei Überlegungen ins Zentrum gestellt.

1. Viele Personen haben grundsätzlich ein Interesse an einem gesunden und nachhaltigen Lebensstil, und somit an gesunder Ernährung. Sich gesund zu ernähren ist jedoch aufwändig, es erfordert Wissen um den Metabolismus, Produkte und deren Inhaltsstoffe. Auch wenn dieses Wissen vorhanden ist, ist bleibt optimieren der Ernährung und erfordert viel Aufwand. Es muss mühsam Buch geführt werden, die wenigsten tuen das.
2. Es existieren viele Apps und Plattformen, welche es den Benutzern ermöglichen Daten zu sammeln. Der «tracking-trend» bringt jedoch Daten in Hülle und Fülle. Es lässt sich auch ein Trend zu mehr «gesunden» Restaurants und Take-aways feststellen. Eine zielgerichtete und personalisierte gesunde Ernährung bleibt aber schwierig. Ein Gastrounternehmen kann diese Daten nutzen, um dem Kunden ohne Aufwand Empfehlungen zu unterbreiten und ihn bei seinen Zielen in Punkto Ernährung zu unterstützen.

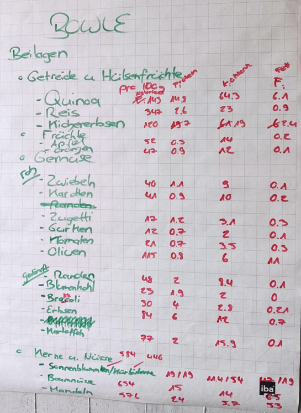
Wir wollen es möglich machen, sich ohne viel Aufwand, nicht nur gesund, sondern optimal auf die Person und die Lebenssituation zugeschnitten zu ernähren. Dafür schaffen wir eine App, welche es ermöglicht die nötigen Informationen zur Person sowie zum Lebensstil zu sammeln. Wir verwenden diese Informationen, um dem Benutzer gesunde und für seine persönlichen Bedürfnisse optimale Gerichte vorzuschlagen. Um diese Gerichte muss sich der Benutzer aber nicht selbst kümmern, sondern kann diese gleich bestellen und abholen – *personalisierter healthy fast-food*!

Die Benutzer erhalten eine auf *Web-Technologien* basierenden *App (PWA),* die sich in Zukunft auf Plattformen von bekannten Anbietern wie Garmin, Suunto oder Nike verbindet und die sowieso bereits vorhandenen Daten nutzt, um den Benutzer ohne weiteren Aufwand eine gesunde Ernährung ermöglicht. Die *App* kombiniert die erhaltenen Daten wie Aktivitäten, tägliche Bewegungsmuster, Puls und Gewicht, um den Bedarf an verschiedenen Nährstoffen zu bestimmen. Mit *künstlicher Intelligenz* bestimmen wir für den Benutzer Nährstoffe basierend auf Fotos und machen so das Tracking kinderleicht. Im Zentrum steht die freshfood-station, denn aufgrund der gesammelten Informationen wird dem Benutzer nun das jeweils optimale Menü vorgeschlagen. Egal ob abnehmen oder neue sportliche Höchstleistungen vollbringen, wir machen es möglich. Ä guete!

## Umgesetzte Use Cases

* Registrierung (für Kunden).
* Sicheres Login und Logout (für Kunden).
* Eigenes Profil anzeigen und editieren.
* Extrahieren und Speichern von Nährwerten aus Bildern mit künstlicher Intelligenz.
* Berechnen des Tagesbedarfs an Nährwerten basierend auf Profil\*.
* Hinterlegen von Zutaten, Gerichten und Menüs inkl. Nährwerten (Backend-only).
* Dynamisches und flexibles berechnen von Menü Nährwerten anhand Zutaten.
* Berechnen des aktuellen Bedarfs an Nährwerten und unterbreiten von Menü-Vorschlag\*\*.
* Grafische Anzeige, wie sich die Menü-Vorschläge zum errechneten Bedarf verhalten.
* Möglichkeit zur Editierung der Vorschläge.
* Bestellungen mit Details speichern und zur Abwicklung weiterreichen.
* Simulation Bezahlen & Bestellabwicklung.

\* Aktuell Kohlenhydraten, Protein, Fett und Kalorien. Wir haben dazu die aktuell in der Wissenschaft am geeignetsten angesehenen Formeln verwendet.

\*\* Die Menüs wurden durch die Challenge Sponsors während dem Hackaton zusammengestellt und mit uns nach unseren technischen Bedürfnissen abgeglichen. Wir verwenden reale und aufwändig erstellte Daten.

## Abgrenzungen

Die aufgeführten Features sind komplett umgesetzt und können verwendet werden. Dies gilt auch für die Sicherheitsmassnehmen (Passwort-Hashing, starke Authentisierung, sichere Verbindung) Auf folgende Dinge haben wir jedoch bewusst verzichtet.

* Es gibt keine grafische Benutzeroberfläche für Stammdaten steht nicht zur Verfügung (Zutaten, Menüs, Gerichte)
* Es wurde kein Activity-Tracker angebunden.
* Die künstliche Intelligenz zum Extrahieren von Nährwerten müsste mit mehr Daten trainiert werden (wir haben während des Hackaton 323 annotiert)
* Self-service Funktionalitäten für Benutzer (editieren der persönlichen Daten, Passwort ändern und/oder zurücksetzen) sind nicht umgesetzt.
* Bei der Empfehlung handelt es sich um ein Optimierungsproblem. Wir haben keine Metaheuristiken angewandt, sondern einen trivialen, aber nicht optimalen Ansatz verwendet (kleinste Distanz).
* Die PWA ist nur rudimentär implementiert und nutzt aktuell keine weiteren PWA Features ausser der lokalen Installation.
* Die Notifikationen in der App funktionieren nur, wenn der Benutzer die App geöffnet hat.
* Es sind keine echten Zahlungsprovider eingebunden.

# Technische Aspekte

Das folgende Kapitel zeigt den technischen Aufbau des «MixMeal» Software-Systems. Zuerst werden Architekturentscheide erläutert, welche signifikanten Einfluss auf die Lösung haben. Danach wird die Baustein- und Verteilungssicht der Lösung aufgezeigt.

## Architekturentscheide

1. Realisierung als Webapplikation bzw. als PWA: Die inhärente Plattformunabhängigkeit, sowie die Möglichkeit für kurze Releasezyklen führten zu diesem Entscheid.
2. Realisierung als SPA: Durch den gewählten PWA-Ansatz, muss die Applikation sinnvollerweise eine PWA sein.
3. Backend bietet eine REST-Schnittstelle an: Da der Client mittels JavaScript realisiert wird, ist dieser Ansatz am besten geeignet.
4. Das Frontend wird mit VUE.JS realisiert: VUE.JS ermöglicht eine «progressive» Entwicklung. Das Framework lässt sich am Anfang leichtgewichtig einsetzen und lässt ein rasches Prototyping zu. Wird die Lösung grösser, skaliert das Framework mit und ermöglicht weiterhin die Umsetzung von wartbaren Lösungen. Aktuell wird JavaScript eingesetzt. Wächst die Lösung, ist der Umstieg auf TypeScript möglich.
5. ASP.NET Core & C# im Backend: Bei ASP.NET Core handelt es sich um ein Plattformunabhängiges «cloud native» Web Framework. Das Framework lässt sich mit seinem Middleware-Konzept sehr leicht an diverse Bedürfnisse anpassen. Standardszenarien sind ohne grossen Aufwand abgedeckt. Es ist zudem auf Durchsatz optimiert.
6. Nutrition-Estimator als eigenständiger Service: Da der Estimator mittels künstlicher Intelligenz umgesetzt werden soll, haben wir dafür eine passende Technologie gewählt und deshalb wurde dieser Baustein als eigenständiger Service implementiert.
7. Die Persistenz mittels einer PostgreSQL: Die Applikation enthält einige relationale Beziehungen und in Zukunft könnte auch Transaktionalität nützlich sein. Aufgrund des Preis-Leistungs-Verhältnisses von RDMBS auf Cloud-Plattformen fiel der Entscheid auf PostgreSQL.
8. Hosting auf Microsoft Cloud Azure: Hosting auf einem dem höchstmöglichen Abstraktionlayer (z.B. SaaS) wurde gewählt, um initialaufwand zu vermeiden. Alle benötigten Dienste stehen auf Azure zur Verfügung und das Team verfügt über entsprechende Erfahrungen mit der Plattform. Um die Lösung später skalieren zu können verwenden wir jedoch Container für das Deployment (single container deployment).
9. Folgende Entscheide sind aufgrund der vorherrschenden Verbreitung der jeweiligen Technologien und dem Vorwissen des Teams gefallen:
   1. API Dokumentation: Swagger & Swagger UI
   2. Authentication: Bearer Token & JWT
   3. Automatisierte Tests: xUnit (Komponenten und Integrationstests)
   4. Styling: Bootstrap v5 mittels Sass

## Bausteinsicht

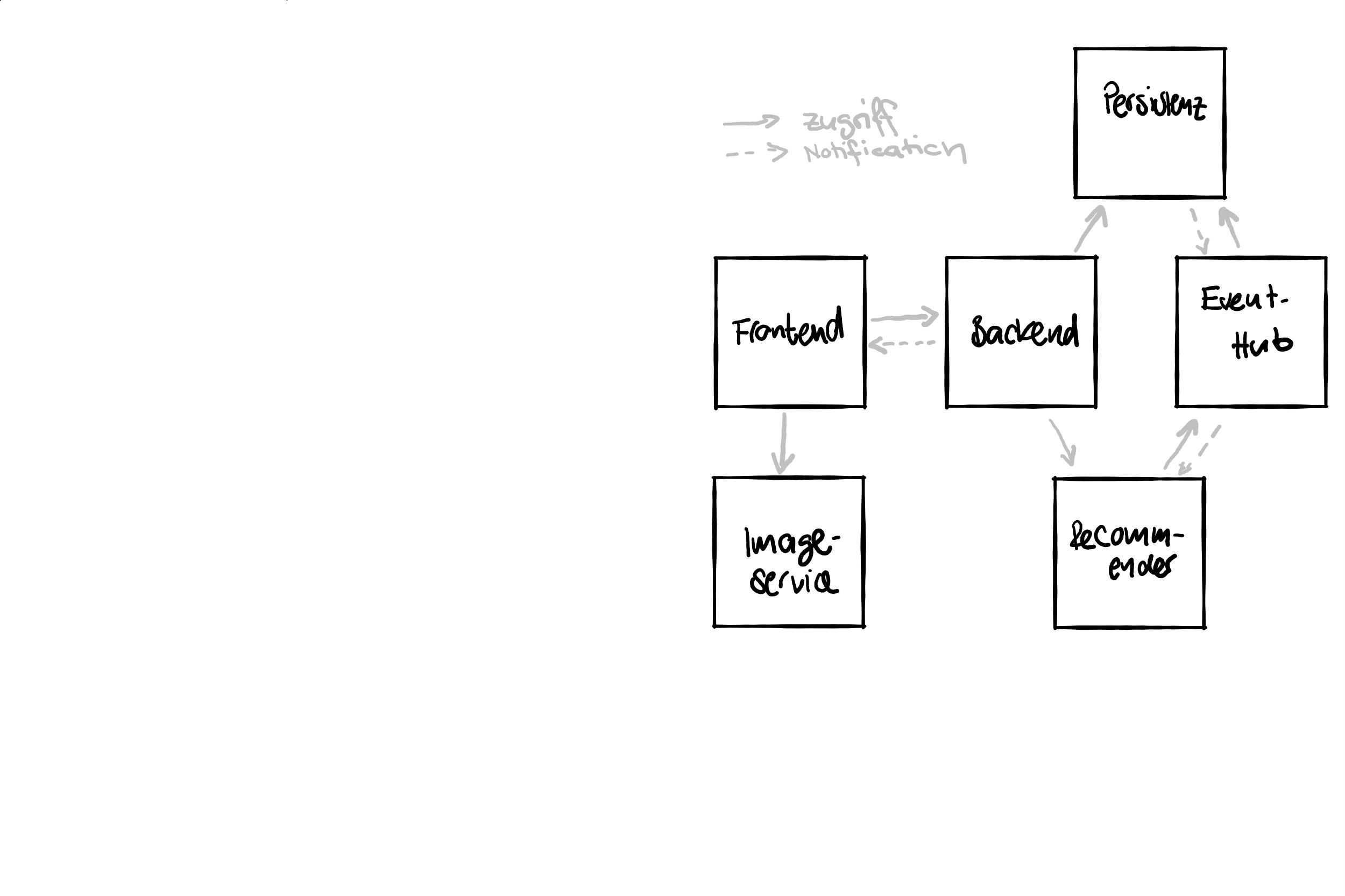
Die Bausteine Frontend, Recommender und backend sind Eigenentwicklungen, während die Baustein Persistenz, EventHub und Image Service als SaaS bezogen wird.

Abbildung Bausteinsicht

### Frontend

Das Frontend wurde mittels VUE.JS umgesetzt. Der Aufbau folgt dem Vue.js Standard. Es werden folgende Konzepte eingesetzt: Single-file-Components, vue-js-router, vuex. Für das Styling kommt Material Design Bootstrap und Sass zum Einsatz.

### Backend

Das Backend wird als eine einzelne Applikation umgesetzt. Es wird ASP.NET Core MVC eingesetzt. Die Lösung wird in fachlichen Modulen (sog. Features) gegliedert. Die fachliche Schichtung bietet die Möglichkeit später eine Aufsplittung in eine Microservice-Architektur vorzunehmen.

### Recommender

Die Recommender-Engine ist als «Content-Based-Filter-Recommender» in der Programmiersprache Python implementiert. Die Daten werden über den EventHub empfangen und aktuell gehalten, um sinnvolle Vorschläge unterbreiten zu können. Die Felder «Name», «Beschreibung» und «Kategorie» der Offers sowie Paid-Offers werden einem Pre-Processing unterzogen (d.h. Tokenization, Stopwords und Stemming/Lematization). Um die Relevanz von Dokumenten zu bestimmen wird die. TF-IF über den Text-Korpus gebildet. Über die Web-API kann mittels eines Suchtextes eine Empfehlung abgefragt werden. Um eine Empfehlung abzugeben wird die Kosinus-Distanz zum Suchtext berechnet. Das Resultat ist eine nach Distanz aufsteigend sortierte Liste der Resultate, welche oberhalb eines Thresholds liegen. Eingesetzte Libraries sind: pandas, scikit learn, flusk, nltk, pyMongo

### Persistenz

Die CosmosDb von Microsoft Azure unterstützt die MongoDb API und das MongoDb Protokoll. Sie kann als SaaS bezogen werden und wurde deshalb ausgewählt.

### EventHub

MongoDb exponiert Änderungen an Collections in Realtime als sogenannte Change Streams. Applikationen können einen Change Stream abonnieren und auf diese Ereignisse reagieren. Dieses Feature von MongoDb unterstützt auch die CosmosDb unter dem Namen «Change Feed». Dieses Feature benutzen wir, um das Recommender-System über Änderungen zu informieren.

## Verteilungssicht

Die gesamte Lösung wird auf Microsoft Azure gehostet. Grundsätzlich wurde darauf geachtet, dass kein Vendor-Lock-In entsteht. Alle Komponenten lassen sich auch On-Premise oder in einer anderen Cloud betreiben.

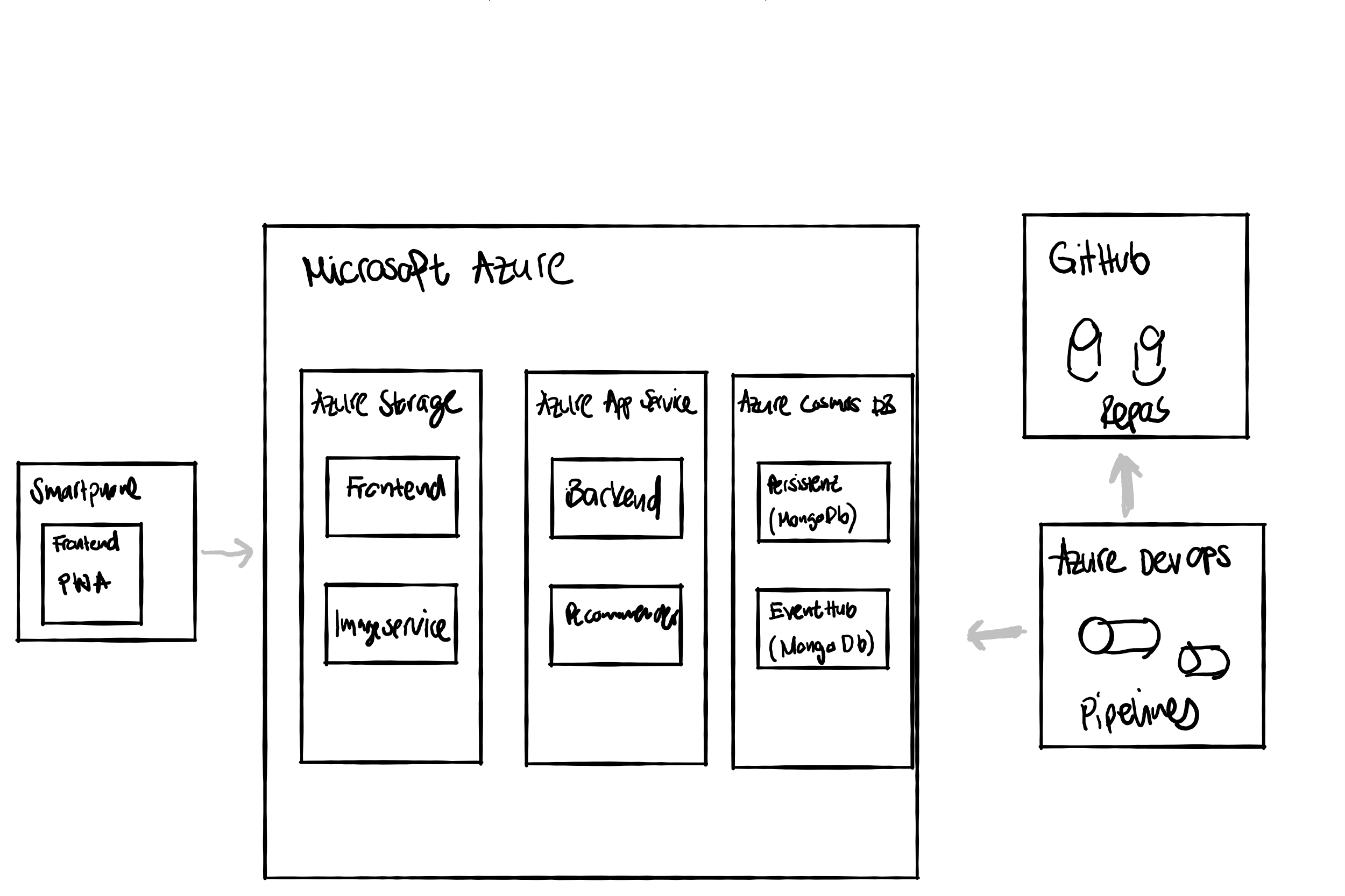


Abbildung Verteilungssicht

### Toolchain

Der Source Code wird innerhalb von mehreren Git-Repositories auf GitHub gehostet. Die CI/CD-Pipeline wurde mit Azure DevOps Pipelines implementiert.

### Frontend

Azure Storage stellt die Möglichkeit bereit, sehr einfach und günstig eine statische Website zu hosten. Es muss keine Infrastruktur betrieben werden. Das Frontend ist entweder im Browser direkt verwendbar oder kann als PWA auf dem Smartphone installiert werden.

### Backend

Azure App Service stellt eine Hosting Umgebung für verschiedene Technologien zur Verfügung. Es musst keine Infrastruktur betrieben werden. Das Backend wird nicht containerisiert, sondern direkt auf einer Linux VM gehostet (Kestrel wird als Webserver verwendet).

### Recommender

Die Recommender Applikation wird als Single-Container als Azure App Service gehostet. Somit ist auch hier keine weitere Infrastruktur nötig. Das Image für den Container wird über die CI/CD-Pipeline erstellt und steht auf einer privaten Azure-Docker-Registry zur Verfügung.

### Persistenz

Bei CosmosDb handelt es sich um eine vielseitige DocumentDb, welche verschiedene Protokolle und APIs unterstützt. Es handelt sich um eine SaaS-Lösung und es muss keine Infrastruktur betrieben werden. Die CosmosDb bietet eine API welche Kompatibel mit der MongoDb-API ist.

### EventHub

Siehe Persistenz.

## Implementation

Nachfolgend einige Punkte, welche wir besonders hervorheben möchten.

* Wir sind überzeugt, dass die Idee mit den lokalen Guides ein Garant für gute Erlebnisse ist, welche sich positiv auf das Image des Kantons Bern bei Touristen auswirkt.
* Das UI leichtgewichtig und das Bedienen der Applikation fällt einfach und macht Spass.
* Alle Funktionalitäten, welche sichtbar sind, sind implementiert und voll funktionsfähig. Unser System verwendet mit wenigen Ausnahmen keine Mocks oder grafische Elemente ohne Funktionalität.
* Die gesamte Lösung ist live und für jedermann Verfügbar. Die Demos erfolgen auf Applikationen, die auf Azure laufen und nicht nur auf dem Entwicklungsrechner.
* Das System kann vollständig in der Cloud betrieben werden, es gibt jedoch trotzdem keinen Vendor-Lock-In.
* Die Lösung ist in der Entwicklung sowie auch bei der Verwendung Crossplattform tauglich.
* Grundlagen für Sicherheit sind gelegt (Verwendung von HTTPS, signierten und ablaufenden JWT, sicher gespeicherte Passwörter, CORS).
* Vollständige Automatisierung von Build und Deployment (CI/CD-Pipeline), für alle drei Teilkomponenten, welche ein Deployment benötigen.
* ist, in einem gut wartbaren Zustand und die Lösung kann weiterentwickelt werden.
* Der Technologie-Mix zur erstellen der Lösung (das beste Tool für eine Aufgabe).

# Ausblick

Die nachfolgenden Punkte zeigen auf was mit dieser Lösung in Zukunft noch gemacht werden kann.

* Ausbau der Standard Funktionalitäten einer Online Plattform (Password zurücksetzen usw).
* Stammdatenpflege über eine Benutzeroberfläche.
* Anbindung der Bezahlfunktionalität
* «Doodle»-Funktionalität (wann kann man eine Aktivität unternehmen etc)
* Erweitern für Gruppenaktivitäten
* Offline Notification (Erweiterung der PWA Funktionalität)
* Anbinden von anderen Plattformen (z.B. ÖV-Ticket)
* Gewinnen von Sponsoren, welche paid offers hinterlegen
* Implementieren einer Abrechnungsfunktionalität
* Ausbauen des Recommender-Systems mit Hilfe eines Kollaborativen-Ansatzes (was haben andere Leute gemacht)
* Möglichkeit Aktivitäten und Guides zu bewerten
* Prämiensystem für Guides
* Gamification aufgrund der unternommenen Aktivitäten

# Verzeichnisse

[Abbildung 1 Bausteinsicht 7](#_Toc50275270)

[Abbildung 2 Verteilungssicht 9](#_Toc50275271)

1. [https://www.bernhackt.ch/challenge-feelfree-mix-your-own-meal/](https://www.bernhackt.ch/be-tourismus/) [27.09.22] [↑](#footnote-ref-1)