BÄRNHÄCKT

freshfood STATION

«Xung anstatt Pfung»
«mix your own meal»

Informationen für die Jury

Inhalt

1 Zugänge

1.1 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]

1.2 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]

1.3 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]

2 Fachliche Aspekte

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

2.1 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 Menüs zusammenstellen, bestellen und später abholen kann.[1]

2.2 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 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*!

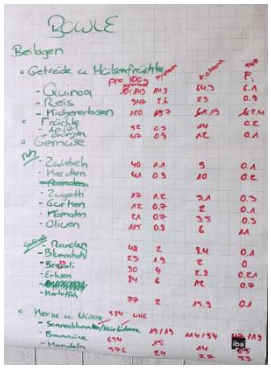
Die Benutzer erhalten eine auf *Web-Technologien* basierenden *App (PWA),* die sich in Zukunft auf Plattformen von bekannten Anbietern wie Garmin, Suunto oder Nike verbindet und die sowieso bereits vorhandenen Daten nutzt, um den Benutzer ohne weiteren Aufwand eine gesunde Ernährung ermöglicht. Die *App* 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!

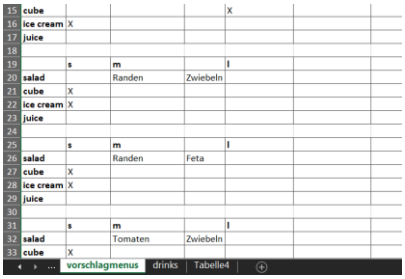
[1] https://www.bernhackt.ch/challenge-feelfree-mix-your-own-meal/ [27.09.22]

2.3 Umgesetzte Use Cases

- Registrierung (für Kunden).
- Sicheres Login und Logout (für Kunden).
- Eigenes Profil anzeigen.
- 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 Anzeige von Details zu den Menü-Vorschläge (inkl. Filterung).
- 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.

2.4 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.
- Die Bestellung wird gespeichert, jedoch nicht weitergereicht und die Notifikation über die gestartete Verarbeitung ist simuliert.

3 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.

3.1 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:
    a. API-Dokumentation: Swagger & Swagger UI
    b. Authentication: Bearer Token & JWT
    c. Automatisierte Tests: xUnit (Komponenten und Integrationstests)
    d. Styling: Bootstrap v5 mittels Sass

3.2 Bausteinsicht

Die Bausteine Frontend, Estimator und Backend sind Eigenentwicklungen, während der Baustein Persistenz als SaaS bezogen wird.

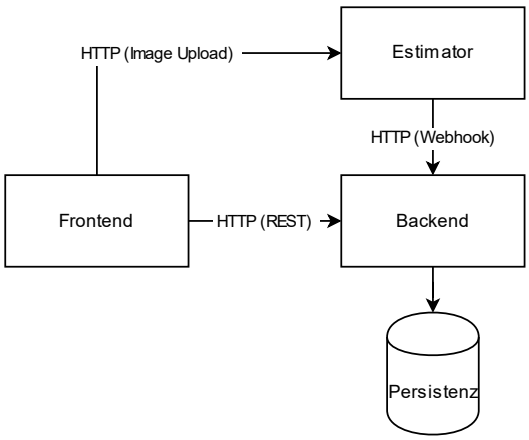


Abbildung 1 Bausteinsicht

3.2.1 Frontend

Das Frontend wurde mittels VUE.JS v3 umgesetzt. Der Aufbau folgt dem Vue.js Standard. Es werden folgende Konzepte eingesetzt: Single-file-Components, vue-js-router, pinia. Für das Styling kommt Bootstrap v5 und Sass zum Einsatz. Als Builds werden mittels vite gemacht.

3.2.2 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. Für den Zugriff auf die Datenbank wird Entity Framework Core verwendet.

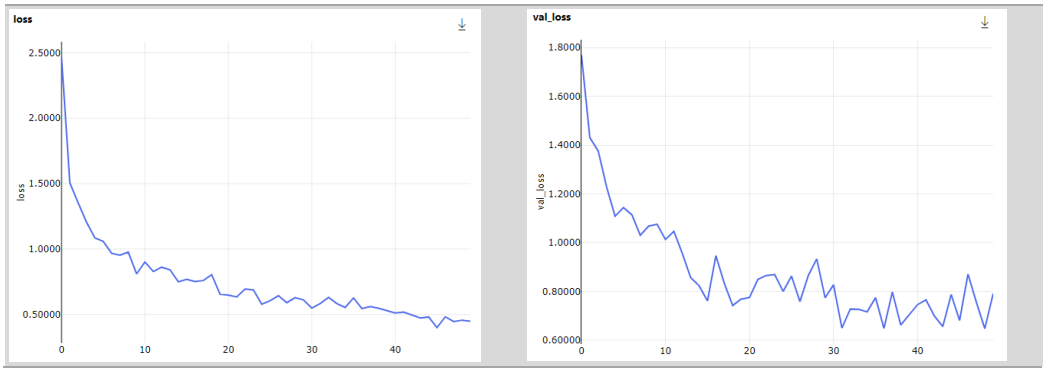
3.2.3 Nutrition-Estimator

Der Estimator für die Nährungswertschätzung von Gerichten aus Fotos ist als Image Segmentation und Classification umgesetzt. Der Service ist in Python implementiert. Es wurde ein Mask R-CNN[2] mit einem Resnet50 Backbone eingesetzt. Solche Modelle benötigen eine hohe Anzahl an Traingsdaten, aus diesem Grund haben wir uns für Transfer Learning entschieden. Unser Modell basiert auf den den Image Segmentation Weights des ImageNet Dataset. Im Train Prozess werden dann nur die letzten 3 Convolutation Layer mit den Bildern von Gerichten neu trainiert. Hierfür haben wir 323 Bildern manuell annotiert, diese Bilder wurden dann in einen Train / Validation Dataset aufgeteilt. Das Modell wurde mit folgenden Hyperparametern trainiert.

Parameter	Wert
Batch Size	10
Epochs	50
Learning Rate	0.0001
Optimizer	SGD (Gradient descent with Momentum)
Loss	Custom Implementation

Die Historie des Loss entwickelt sich innerhalb der 50 Epochen sehr gut. Man sieht das Modell lernt die Masken und die Nahrungsmittel Klassen, die definiert wurden.

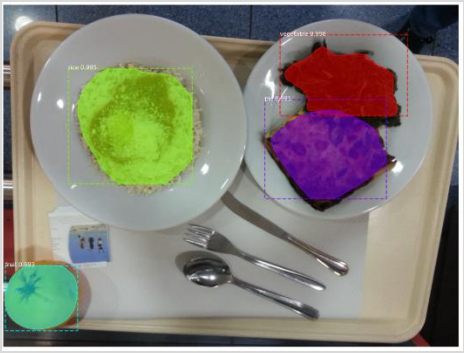
[2] https://arxiv.org/pdf/1703.06870.pdf [28.09.2022]



Der Vergleich des Loss mit dem Validation Loss sieht auch sehr gut aus. Es kann ausgeschlossen werden das, dass Modell overfittet und somit gut generalisiert.

Nach dem Train Step wird ein zusätzlicher Evaluation Step durchgeführt, hier wird das Modell mit Bildern getestet, die weder im Train noch im Validation Dataset vorhanden sind. Hier ein paar Beispiele der Evaluierung.

Beide Samples funktionieren gut, es werden alle Nahrungsmittel erkannt und korrekt klassifiziert.



In diesen Beispielen wird ein Nahrungsmittel nicht erkannt oder ein der Kassenzettel wird als Brot erkannt.

Das Trainiert Model wird dann vom Nutrition-Estimator Service verwendet. Diese Service ist in mit FastAPI und Python umgesetzt und stellt eine API zur Verfügung, welche das Bild entgegennimmt und dann die Segmentation und die Klassifikation durchführt. Anhand der Maske und der Klasse wird dann die Nutrition geschätzt. Der Wert wird anhand der Fläche des Nahrungsmittels und des Nurtionwerts pro Quadratzentimeter berechnet. Das Resultat aller Werte wird zurückgeliefert und wird vom Tagesbedarf des Benutzers abgezogen.

Für das Training und das Tracking des Experiments, Dataset-Verwaltung und Modellverwaltung wurde Azure ML Workspace eingesetzt.

Eingesetzte Libraries:
Azure ML Workspace, Tensorflow,, scikit.image, imgaug, Open-CV, numpy, pandas, FastAPI

3.2.4 Persistenz

PostgreSQL oft kurz Postgres genannt, ist ein freies, objektrelationales Datenbankmanagementsystem (ORDBMS). Seine Entwicklung begann in den 1980er Jahren, seit 1997 wird die Software von einer Open-Source-Community weiterentwickelt. PostgreSQL ist weitgehend konform mit dem SQL-Standard SQL:2011. D.h. wenn keine spezifischen Features verwendet werden, kann das RDMBS auch ausgetauscht werden. Die höchste Abstraktionsebene für Postgres auf Azure ist «Azure Database for PostgreSQL». Es handelt sich um eine voll gemanageden database-as-a-service.

3.3 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.

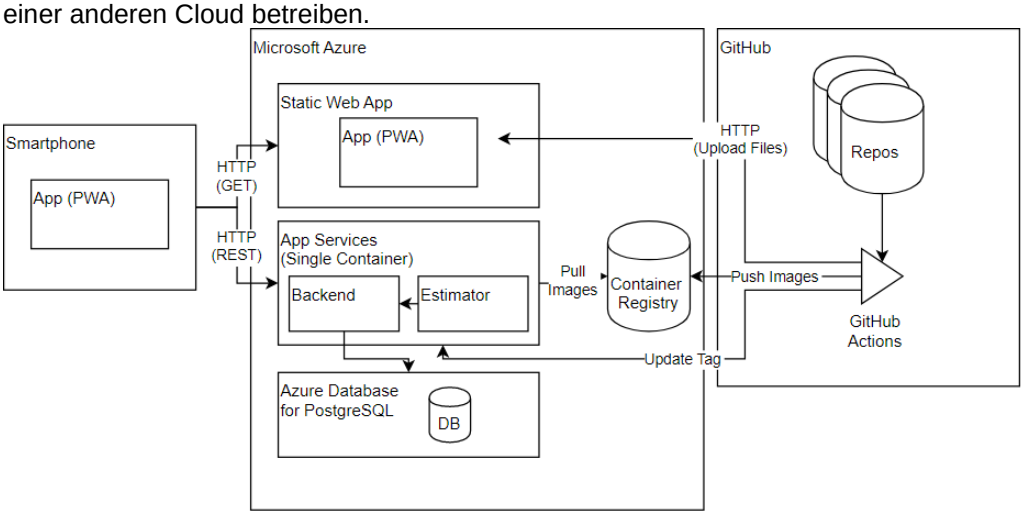


Abbildung 2 Verteilungssicht

3.3.1 Toolchain

Der Source Code wird innerhalb von mehreren Git-Repositories auf GitHub gehostet. Die CI/CD-Pipeline wurde mit GitHub Actions implementiert. Für das Backend und den Estimator werden Docker-Images generiert. Diese werden in eine lösungseigene Container Registry gepushed.

3.3.2 Frontend

Azure bietet mit «Static Web App» eine günstiger Alternative zum App Service. Dabei handelt es sich um einen einfachen Webserver, es braucht keine Runtime. Das Frontend ist entweder im Browser direkt verwendbar oder kann als PWA auf dem Smartphone installiert werden.

3.3.3 Backend

Azure App Service stellt eine Hosting Umgebung für verschiedene Technologien zur Verfügung. Es muss keine Infrastruktur betrieben werden. Eine der Möglichkeiten ist das Deployment eines einzelnen Docker-Container. Das Backend wird als Linux-Container gehostet und intern wird der mit .NET verbaute Webserver «Kestrel» verwendet. Das Image für den Container wird über die CI/CD-Pipeline erstellt und steht auf der Lösungseigenen privaten Azure-Docker-Registry zur Verfügung.

3.3.4 Estimator

Die Estimator Applikation wird ebenfalls als Single-Container mittels 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 der Lösungseigenen privaten Azure-Docker-Registry zur Verfügung.

3.3.5 Persistenz

Azure Database for PostgreSQL ist ein relationaler Datenbankdienst, der auf der Open-Source-Postgres-Datenbank-Engine basiert. Der vollständig verwaltete Dienst vom Typ „Database-as-a-Service“ kann geschäftskritische Workloads mit planbarer Leistung, Sicherheit, Hochverfügbarkeit und dynamischer Skalierung verarbeiten.

3.4 Implementation

Nachfolgend einige Punkte, welche wir besonders hervorheben möchten.

- Wir sind überzeugt, dass ein Gastronom mit IT-Kompetenz und der Digitalisierung sich auf dem Markt effektiv differenzieren kann, und dabei grosser Mehrwert für die Kunden entsteht.
- 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).

4 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.
- Anbindung von Activity-Tracker und andere IoT Devices (wie z.B. die digitale Waage)
- Soziale Elemente (beliebte Menüs, Freunde die bestellen)
- «Einfacher» und nicht personalisierter Online-Shop, wie ansonsten üblich ebenfalls anbieten
- Offline Notification (Erweiterung der PWA Funktionalität)
- Verbesserung der Algorithmen (Berechnung von Nährwerten, Menü-Vorschläge)
- Estimator durch mehr Trainingsdaten verbessern
- Menü Bewertungen einfügen
- Ausrichtung der Empfehlung auf Ziele der Benutzer (z.B. Abnehmen, Muskelaufbau etc.)
- Mit einbeziehen von geplanten Aktivitäten (z.B. Trainingsplan)

5 Verzeichnisse