BÄRNHÄCKT

«Tuuri’s»
«Digitale Heimat- und Gästekarte für die Ferienregion Bern»

Informationen für die Jury

Inhalt

1 Zugänge

1.1 Aktueller Stand des Sourcecodes

- https://github.com/baernhaeckt [Organisation]
- https://github.com/baernhaeckt/Backend2020 [Backend Source]
- https://github.com/baernhaeckt/Frontend2020 [Frontend Source]
- https://github.com/baernhaeckt/Recommender2020 [Recommender Source]
- https://github.com/baernhaeckt/Misc2020 [Documentation, Presentation, Misc]

1.2 CI/CD Pipeline

- https://dev.azure.com/baernhaeckt/BaernHaeckt2020/_dashboards/dashboard/7c97efa6-fb51-49ad-95d1-76affb7b42ea [Dashboard]
    - Frontend Build & Release
    - Backend Build & Release
    - Recommender Build & Release

1.3 Production

- https://baernhaeckt2020.z19.web.core.windows.net/ [Frontend]
- https://baernhaeckt2020.azurewebsites.net/swagger/index.html [Backend – Swagger UI]
- https://baernhaecktrecommender.azurewebsites.net/offers?text=stadt [Recommender output offers]
- https://baernhaecktrecommender.azurewebsites.net/paidoffers?text=stadt [Recommender output paid offers]

2 Fachliche Aspekte

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

2.1 Ausgangslage

Im Rahmen der Challange soll für die BE! Tourismus AG ein überregionaler digitaler Visitor Pass geschaffen werden. Er soll sich von anderen bestehenden Produkten abheben und einen Mehrwert und praktischen Nutzen bieten. Der Ansatz «Mobile First» und die Selbstfinanzierung des Passes ist dabei zentral. [1]

2.2 Lösungsansatz

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

1. Viele Reisende sind auf der Suche nach Erlebnissen im fremden Land, die üblichen Sehenswürdigkeiten sind rasch abgeklappert und langweilige kehrt ein. Der Kanton Bern hat jedoch viel zu bieten. Was es alles zu erleben gibt wissen am besten die Einheimischen. Kontakt mit der einheimischen Bevölkerung ist zudem oftmals gewünscht und gibt dem erlebten eine besondere sowie persönliche Note.

2. Der Pass soll sich Selbstfinanzieren. Viele Partnerschaften aufzubauen und Verträge auszuhandeln ist entsprechend teuer (Personalaufwand, Logistik für den Pass etc). Zudem sollen die Reisenden nicht zu tief in die Tasche greifen müssen.

Wir wollen einheimische als lokale «Guides» gewinnen. Sie stellen sich zur Verfügung, um mit den Reisenden eine Aktivität zu unternehmen. Die Reisenden entscheiden sich gemäss ihren Interessen für eine Aktivität und einen «Guide». Mit dem Einsatz von einheimischen «Guides» muss nicht mit jedem Lokal ein Vertrag bestehen – der «Guide» übernimmt vorerst die Unkosten für die im Package beinhaltenen Leistungen. Nebst dem Austausch mit den spannenden internationalen Gästen, erhält der «Guide» natürlich auch eine kleine Entschädigung. Mit Werbung, ausgehandelten Vergünstigungen und einer kleinen Marge können die Ausgaben gedeckt werden.

Die Reisenden und die «Guides» finden sich spielend mit der auf *Web-Technologien* basierenden *App (PWA)*. Die mit *künstlicher Intelligenz* gefundenen Empfehlungen für Aktivitäten, basierend auf Interesse garantieren Spass für Reisenden und Guide. Die *App* bietet dem Reisenden alles was er braucht. Über eine Partnerschaft wird automatisch ein ÖV-Ticket gekauft, welches den Reisenden zur Aktivität bringt. Der «Guide» scannt beim Empfang seines Gastes den QR-Code, welcher bestätigt, dass er den richtigen Gast vor sich hat und dieser im Besitz eines gültigen Passes ist. Das Abenteuer kann nun beginnen.

[1] https://www.bernhackt.ch/be-tourismus/ [05.09.20]

2.3 Umgesetzte Use Cases

- Registrierung (für Guides und Touristen)
- Login und Logout (für Guides und Touristen)
- Eigenes Profil anzeigen und editieren.
- Interessen mit «Swipe» Ansatz definieren
- Empfehlung von passenden Aktivitäten basierend auf den Interessen*
- Vorschlag von passenden beworbenen Angeboten basierend auf den Aktivitäten*
- Details zu empfohlenen Aktivitäten anzeigen
- Buchen einer Aktivität bzw. eines Packages bei einem lokalen «Guide»
- Aktivitätspass in Form eines QR-Codes
- Automatisches kaufen eines ÖV-Tickets für die Aktivität in Form eines QR-Codes**
- Erworbenen Pass ansehen
- Verantwortlicher Guide kann Gutschein über QR-Code Scan ansehen.
- Verantwortlicher Guide kann Gutschein abbuchen.
- Echtzeit Benachrichtigung, wenn der Gutschein von «Guide» abgebucht wird.

* Die Empfehlungen werden mittels künstlicher Intelligenz generiert. Das System kommt so auch mit neuen Daten d.h. Interessen und Aktivitäten zurecht. Anpassungen an Angeboten werden asynchron an das Recommender-System übermittelt, damit dieses seine Empfehlungen berechnen kann.

** Anbindung ist nur vorgesehen und nicht umgesetzt.

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.

- Die Administrationsoberfläche für Stammdaten steht nicht zur Verfügung (Interessen, Angebote, bezahlte Angebote)
- Es gibt keine grafische Benutzeroberfläche, um Bilder zu hinterlegen (Guides, Angebote & Interessen)
- Es wurde kein echter Partner angebunden (Angebote und ÖV-Ticket).
- Self-service Funktionalitäten für Benutzer (editieren der persönlichen Daten, Passwort ändern und/oder zurücksetzen) sind nicht umgesetzt.
- 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.

3 Technische Aspekte

Das folgende Kapitel zeigt den technischen Aufbau des «Tuuri» 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 Web Applikation bzw. als PWA: Die inhärente Plattformunabhängigkeit, sowie die Möglichkeit für kurze release Zyklen 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. Wo «push» vom Server nötig ist, werden Websockets eingesetzt.
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. Recommender als eigenständiger Service: Da der Recommender 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 DocumentDb: Da rasch eine funktionsfähige Lösung entstehen soll, ist der Entscheid zugunsten einer schemalosen DocumentDb gefallen. Aufgrund der vorhandenen Vorkenntnisse wurde der Entscheid zugunsten von MongoDb bzw. der MongoDb API von Azure CosmosDB getroffen.
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.
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: Material Design Bootstrap (MDB) mittels Sass
    e. Frontend Connectivity: Axios für http, SignalR für Websockets

3.2 Bausteinsicht

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

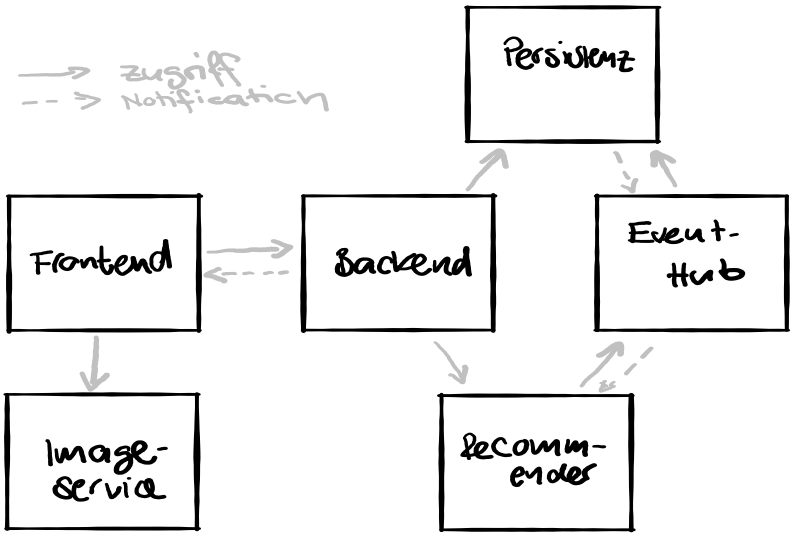

Abbildung 1 Bausteinsicht

3.2.1 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.

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.

3.2.3 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

3.2.4 Persistenz

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

3.2.1 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.

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.

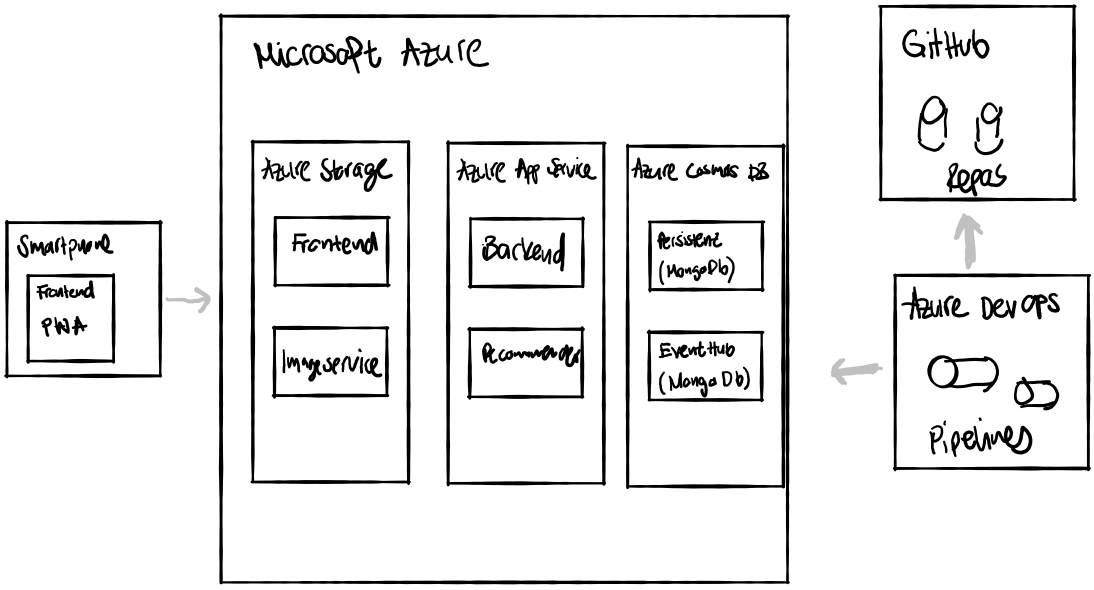

Abbildung 2 Verteilungssicht

3.3.1 Toolchain

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

3.3.2 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.

3.3.3 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).

3.3.4 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.

3.3.5 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.

3.3.6 EventHub

Siehe Persistenz.

3.4 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).

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
- «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

5 Verzeichnisse