friendifyer – Technische Umsetzung

BackEnd

Das friendifyer Backend soll aus einer containerisierten Java Applikation und einer Datenbank und einem Apache Kafka System (Azure Event Hubs) bestehen. Die Java Applikation wird mit dem jdk17 programmiert um die neusten Features, wie z.B. das Pattern Matching nutzen zu können. Als Basis Framework für die Applikation dient das Microservice Framework quarkus.io. Quarkus.io erlaubt das builden nativer Images und deren direktes Dockerisieren mit Hilfe der Graal VM. Quarkus.io ist ein reaktives framework für Micro Services, welches z.B. das reaktive Arbeiten auf SQL Datenbanken erlaubt und es ermöglicht blocking IO Zugriffe auf einen zweiten Threadpool auszulagern. Die Applikation soll zum einen eine REST Schnittstelle für das Abfragen von Daten zur Verfügung stellen und zum andern eine Kafka Schnittstelle für die Echtzeitdatenverarbeitung. Das Backend wird dann in einer Azure Container Instanz betrieben, um eine hohe Verfügbarkeit zu erreichen. Auch der Chat und die allgemeine Verbindung der aktiven Nutzer*innen mit dem System wird von dieser Applikation mittels WebSockets gehandelt. So werden dann auch die Notifications etc. gesendet.

Die Datenbank soll eine SQL Datenbank sein (Azure Elastic SQL Pool), um die einfache Umsetzbarkeit zu gewährleisten. Die Applikation greift mittels eines Java Konnektors den quarkus liefert auf die Datenbank zu. Der Konnektor erlaubt sehr effektive anfragen, da er reaktiv gestaltet ist.

Die Applikation liefert einen internen WebServer mit, mit dem man die Desktop Applikation (WebSite) gleich mit deployen kann.

Durch das Kafka Cluster und die Kafka Kommunikation wird zum einen eine Schnittstelle zur Echtzeitdatenverarbeitung geliefert und zum anderen eine Kommunikation zwischen mehreren Instanzen derselben Applikation, um so die Skalierbarkeit des Systems sicherzustellen. Dies ermöglicht zum Beispiel das Ausrollen in einem Kubernetes Cluster.

FrontEnd

Das FrontEnd soll mittels vue.js erstellt werden. Im Prototypen existiert nur die Website, welche in plain HTML mit css und javascript erstellt wurde. Die Applikation soll später mit Kotlin entwickelt werden, um den eigentlichen Android Standard zu verwenden. Als konnektoren werden im FrontEnd eine WebSocket Verbindung und ein REST Client benötigt.

Authentifizierung

Die Authentifizierung soll mittels Zertifikat über SSL (TLS) laufen. Dies ist im Prototyp nicht umgesetzt, aufgrund der mangelnden Zeit und der Mangelnden Manpower an Entwicklern.