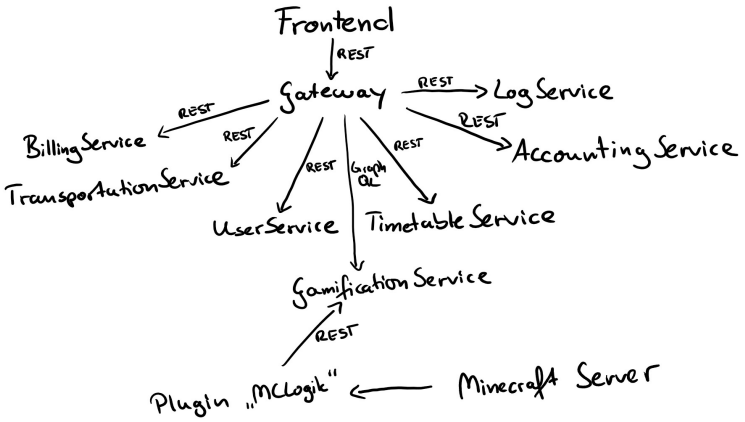
Nachhilfe-App

Soufiane Akarkach und Mathis Neunzig

Architektur der Anwendung

Die Anwendung besteht aus einem Frontend und verschiedenen Microservices, die gemeinsam das Backend bilden. Jeder Microservice kann in anderen Programmiersprachen geschrieben werden und andere Datenbanken nutzen, weshalb ein weites Spektrum an Programmiersprachen, Frameworks und Datenbanken ausprobiert werden kann. Wenn in dieser Microservice-Architektur ein Service nicht funktioniert, stürzt nicht das gesamte Backend zusammen, wie es bei einer monolithischen Architektur passieren würde.



Technischer Aufbau des Frontends

Das Frontend ist mit JavaScript geschrieben, wobei dabei TypeScript zur Typisierung und React Native als Framework verwendet wurde. Zusätzlich zu React Native wird Expo verwendet, um eine schnelle und angenehme Entwicklung für Web, Desktop und Mobilen Endgeräten zu ermöglichen. Das Frontend kommuniziert mit dem Gateway-Service, um Daten aus den verschiedenen Microservices zu bekommen.

Technischer Aufbau des Backends / Microservices

Gateway

Der Gateway-Service ist die API-Schnittstelle, mit der das Frontend kommuniziert und die alle ankommenden REST-Anfragen vereinfacht, loggt und zum Ziel weiterleitet. Der Gateway-Service wird ausschließlich per REST angesprochen, damit im Frontend nur eine Kommunikationsart benutzt werden muss. Für Services, die nicht REST benutzen, stellt der Gateway-Service eine Konvertierung zur Verfügung, sodass Services, die z.B. mit GraphQL angefragt werden, durch den Gateway-Service mit REST anzusprechen sind. Der Gateway-Service ist mit Java programmiert worden, wobei das SpringBoot Framework zusammen mit Maven genutzt wurde, um ein funktionierendes Backend zu erstellen. Da der Gateway-Service keine eigenen Daten hält, wird keine Datenbank benötigt.

Log-Service

Der Log-Service wird vom Gateway angesprochen und dient der automatischen Protokollierung von Statusinformationen und Ereignissen während des Betriebs der Applikation samt Zeitstempel und ID. Da sich die Datenbank Kibana mit ElasticSearch und Logstach (ELK-Stack) als teuer bzw. sehr speicheraufwendig herausgestellt hat, werden die Logs in einer Redis-Datenbank auf der Redis Cloud gespeichert. Der Service ist mit Java, sowie dem SpringBoot Framework in Verbindung mit Maven programmiert worden.

User-Service

Dieser Service wird vom Gateway gesteuert und dient der Verwaltung der User. So werden hier Nutzer angelegt und verwaltet, Skills angelegt und verwaltet, sowie der Login und die Registrierung abgewickelt. Die Programmiersprache des User-Services ist Go, wobei zum

Aufsetzen des REST-Servers Gin Gonic und eine MySQL-Datenbank mit XAMPP benutzt wurde.

Timetable-Service
Der Timetable-Service ist das Gegenstück zum User-Service und dient dem Anlegen und Verwalten der Timeslots - aka Nachhilfestunden. In diesem Service werden allgemeine Informationen zu den Unterrichtsstunden geregelt, sowie auch die Besetzung einer Stunde mit dem Lehrer und dem Schüler verwaltet. Der Service wurde mit JavaScript zusammen mit den Frameworks Express und Node.js entwickelt. Für die Speicherung der Daten wurde als Datenbank SQLite verwendet.

Gamification-Service
Der Gamification-Service beinhaltet Statistiken und Möglichkeiten, spielerisch Punkte zu sammeln oder Challenges abzuschließen, um langweilige und eintönige Aktionen wie das Lernen von Inhalten spannend zu gestalten. Programmiert wurde der Service mit Kotlin, dem Framework SpringBoot und Gradle. Als Kommunikation zwischen den Services als Alternative zu REST wurde GraphQL genutzt, um nur auf bestimmte Einzelstatistiken zugreifen zu können. Die Daten werden in der Datenbank von MongoDB gespeichert, auf die mit Mongo Compass bzw. Mongo Atlas zugegriffen werden kann.

Billing-Service
Der Billing-Service wird für die Erstellung der Rechnung und Buchungsbestätigung genutzt. Der Service verwaltet zudem die Versendung dieser PDF-Dateien per E-Mail. Für den Billing-Service wurden Python und das Framework Django genutzt. Eine Datenbankanbindung wird hier nicht benötigt.

Transportation-Service
Um das Feature mit der Distanz zwischen einem Schüler und einem Lehrer bei der Buchung zu erhalten, wurde der Transportation-Service programmiert, der zusammen mit der Google Maps API benötigte Daten berechnen kann. Eine Datenbank wurde hierfür nicht benötigt. Der Service wurde in der Programmiersprache Erlang mit dem Phoenix-Framework geschrieben, eine Datenbank ist hier nicht benötigt.

Accounting-Service
Um die Themen rund um die Buchhaltung zu verwalten, wird der Accounting-Service verwendet. Dieser Service wurde mit Java geschrieben, wobei die Frameworks SpringBoot und Maven genutzt worden sind. Als Datenbank wurde Redis mit der Redis Cloud verwendet.

Technischer Aufbau von externen Services
Chatbot
Der Chatbot wurde mit der Plattform Collect.chat erstellt. Die Plattform bietet eine intuitive Benutzeroberfläche, sowie ein integriertes Visualisierungstool. Durch die automatische Abspeicherung der Konversationen des Users lassen sich Schwachstellen analysieren und zukünftige Handlungsempfehlungen ableiten.

Minecraft
Der Minecraft-Server ist ein normaler Minecraft-Server mit Spigot auf der momentan neuesten, stabilen Version 1.19.3. Das Plugin, welches das Logik-Modul auf dem Server ermöglicht und die Ergebnisse mit dem Gamification-Service kommuniziert, wurde mit Java und Spigot 1.19.3 programmiert. Auf dem Server sind keine weiteren Plugins installiert.

Anhang: Verwendete Techniken

Programmier-sprache	Framework	Zusatz	Komm.	DB	Einsatzgebiet Service
Java	Spring Boot	Maven	REST		Gateway
			REST	Redis	Accounting
			REST	Redis	Log
Kotlin		Gradle	GraphQL	MongoDB	Gamification
Go	Gin Gonic		REST	MySQL	User
JavaScript	Express	npm, node.js	REST	SQLite	Timetable
Python	Django	pip	REST		Billing
Erlang, Elixir	Phoenix	Google API	REST		Transportation

TypeScript	React Native	Expo			Frontend

X	Collect.chat				Chatbot

Java	Spigot				MC-Server
					MC-Plugin