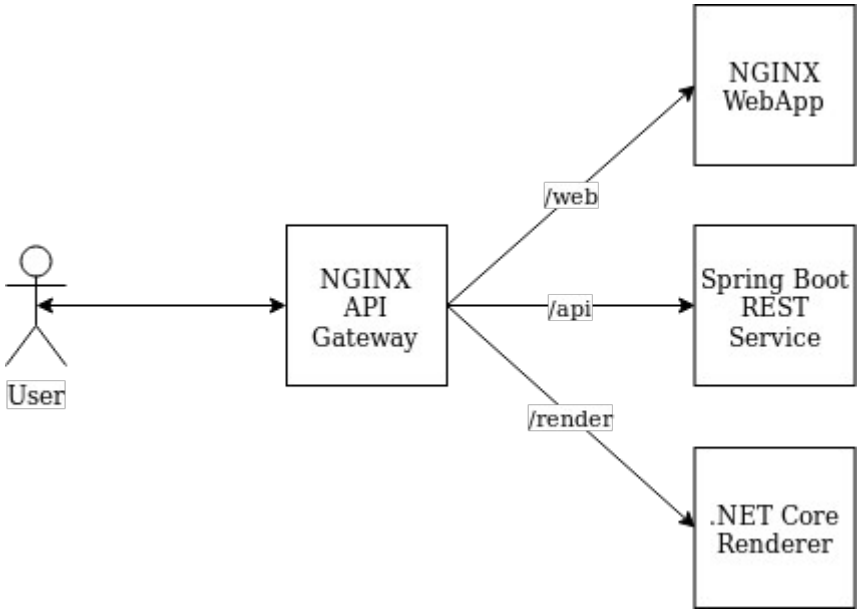
Übungsklausur - Beispiel

Web-based 3D modelling software (W3D)

W3D ist eine web-basierte Anwendung für die Modellierung von 3D Modellen. Ein Benutzer kann mit dieser Anwendung 3D Modelle als Wireframe (hier sind nur die Kanten sichtbar) zeichnen, sowie die Oberfläche des 3D Modells (Texturen, Farben) festlegen und diese dann in einem weiteren Schritt in 3D darstellen (rendern) lassen.

Die Architektur sieht wie folgt aus:



Requests vom Benutzer werden an einen NGINX Webserver geschickt, welcher als Reverse Proxy dient. Die Webseite ist unter /web aufrufbar und wird von einem NGINX Webserver bereitgestellt. Die Funktionalität für die Webseite wird von einer Spring Boot Anwendung über REST zur Verfügung gestellt und ist unter /api abrufbar. Für das Rendern der 3D Modelle gibt es einen eigenen Service unter der URL /render. Dieser wird auch von der WebApp verwendet. Der Renderer ist eine .NET Core Anwendung.

Das Arbeitsverzeichnis auf Ihrem Rechner sieht wie folgt aus:
`/home/vir/W3D/`

`/webapp`: Webseite, HTML, JS und CSS Dateien
`/service`: REST Service, enthält Java Dateien und ein Gradle Build Skript
`/render`: Renderer, enthält C# Dateien und `renderer.csproj`

Aufgaben

a) Schreiben Sie ein Dockerfile welches das API Gateway erstellt. Konfigurieren Sie NGINX damit die Requests richtig an die entsprechenden Container/Services weitergeleitet werden.

b) Schreiben Sie ein Dockerfile für die WebApp. Das Image soll alle benötigten Dateien für die Webseite enthalten.

c) Schreiben Sie ein Dockerfile für das REST Service. Bauen Sie dieses mit `gradle (build)` und erstellen Sie ein Image das die kompilierte Anwendung beim Starten des Containers startet. Die kompilierte Anwendung hat den Dateinamen `service.jar` und befindet sich im `service` Ordner. Sie bindet sich an Port 2351. Wenn Benutzer Modelle speichern möchten, werden diese im Dateisystem unter `/var/opt/models` gespeichert.

d) Schreiben Sie ein Dockerfile für den Renderer. Es soll der Quellcode mit `dotnet` kompiliert werden und die Anwendung mit dem Dateinamen `renderer.dll` beim Starten des Containers gestartet werden. Die Anwendung bindet sich an Port 5523.

e) Schreiben Sie eine docker-compose Datei, welche alle vier Dockerfiles verwendet um die Services zu bauen und zu starten. Nehmen Sie darauf Rücksicht, dass die richtigen Ports freigegeben sind, die Services kommunizieren können und, wenn benötigt, Daten dauerhaft gespeichert werden können.