Architecture Design TuneTurtle

Nieben van Sint Annaland
03-07-2024

Project Description

TuneTurtle is going to be the next Spotify. It's going to have most of Spotify's features such as playlists, liked songs, skipping songs, looping songs, changing volume, recommendations, and as many other features as possible. It's going to be a microservice structured project that is going to be hosted on the cloud.

Architectural Decisions

Cloud Platform

As cloud platform, I have decided to go with Azure. This is mainly because being a student at Fontys makes it very easy to get enough credits to do what I want on the cloud. I initially planned to go with AWS since I'm already going with Azure for my group project, and I wanted to explore the most options I could and learn the most, but AWS makes it hard to get started without adding a credit card, which is something I do not have. Using Azure is also a little more fitting in the whole scheme of things, considering we also use it within our group project.

API & Languages

For my backend I have decided to go with Java, mainly because that is what I am most familiar with, and I would like to focus on learning the cloud aspect of things, instead of figuring out the basics of a language. For my API I have decided to go with gRPC. There's a few reasons why, firstly it's because I'm already familiar with it since I've used it during my internship, secondly it's a lot faster and more performant than a standard REST. There's also more features in gRPC which REST struggle with such as bidirectional streaming, which will possibly be useful for my application. There's also a lot of automatic code generations using .proto files, which saves a lot of time in the end.

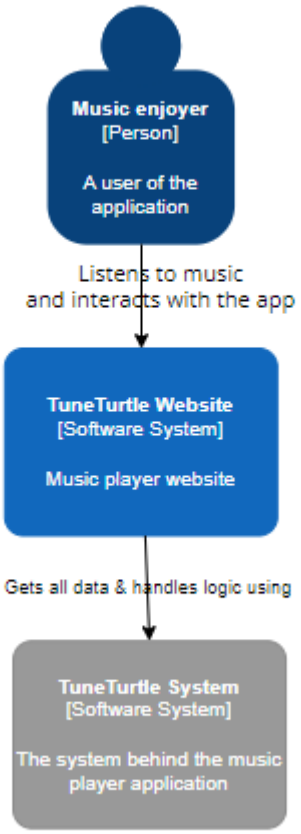
For my frontend I've decided to go with React in Javascript. This is because I'm the most familiar with it.

Messaging

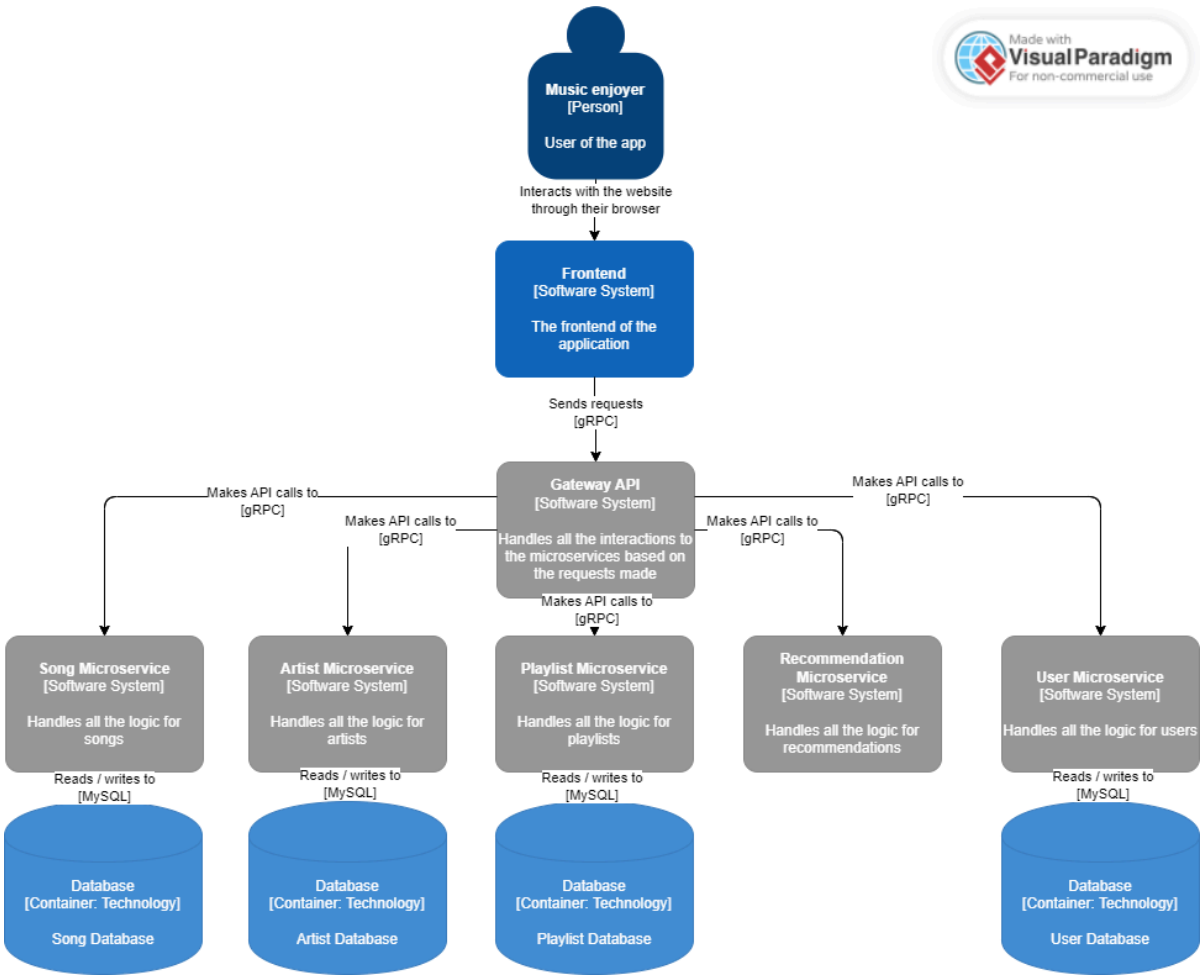
For messaging between microservices, I will be using Redis. I'm using Redis over RabbitMQ because I've already used it in the past and I'm the most familiar with it. The messaging model is also simpler to use than RabbitMQ's. Redis is very performant and is definitely a good fit for this application

C4 Diagrams

C1:



C2:



Sequence Diagram: