KAZAKH BRITISH TECHNICAL UNIVERSITY

Report

Topic:
Microservice Architecture

Made by: Z. Yergeldi
Checled by: K. Sherkhan

Almaty
2021

Contents

Chapter 1

Introduction to project

It's worth starting with what my project is. This is a market for non-fungible tokens, that is, you can virtually talk about some unique goods, hereinafter I will simply call them goods. And on the project they can be bought, sold, created, put up for auction, donated and the like. We can say that everything revolves around these products. Since the goods are created by the users themselves, and they will not be deleted, since they are at least stored in the blockchain, and then in my database, they will only accumulate. In addition to the number of users who make requests to create a product, to create an auction, to participate in tenders, all this will require powerful, fat servers, and this will have to expand horizontally, for example, requests for an auction can be large, and every time, when you need to quickly reply that someone has placed a bet before someone else has placed a bet during another bet. And the most appropriate solution is to use a microservice architecture, although we know that this is not a panacea.

Chapter 2

Dividing to microservices

First, I would separate the auction as a service from the rest. It will have its own separate database, where all bets will be stored and operations will be performed on them.

Secondly, I would separate the payment service, since creating a product, buying a product, bidding for an auction requires payment

Thirdly, I would separate the cache service

Fourthly, it would be cool to separate the logging service so that all data is filtered there, and whatever else is saved.

And the rest of the operations, I will simply divide it into a database, such as a service, data storage will be, where there will be users, products, categories, and the like.

Chapter 3

Compliance with characteristics

And it turns out from the characteristics is observed, for the beginning, the **components via services**, so to speak. Although, of course, the absence of a main database will lead to collapse) Can I say about the fact that there is an **organization near business capability**, I did not really understand, but the database is difficult to say. **Smart endpoints and dumb channels**. Basically, yes, the channel will simply enter the auction, and he himself knows what to do there, if someone's bid is less than the main one, it will return something, if it’s big, and so on. **Decentralized data management,** but we have our own database for each service. **Design for failure**, for example, if the auction falls, then I'll just close the bids for now and stop the auction time until that service rises again. If the cache service crashes, I'll just do a database search, without saving it to the cache and the like. If the payment falls, then the creation of the product, the purchase and payment at the auction will not work yet. I can't say anything about **evolutionary design**, but I think it will be possible to divide the main database, then separately into users and their operations, although it turns out that there will be a dumb endpoint, like we will indicate what to do and the like.

Chapter 4

UML diagram