Équipe EOS Marketplace

EOS Marketplace V1

Guide du développeur

Version 1.0

Developer’s Guide

1. Firebase

We use Firebase to authenticate the application’s users. To register in the application, we use Firebase.auth().createUserWithEmailAndPassword(email, password) to create a user with the provided email and password. The user is then stored inside Firebase’s database with all the other registered users, and he can now access the application.

To log in the application, Firebase.auth().signInWithEmailAndPassword(email, password) is used to sign in a user with the provided email and password. If the user exists, the user can then access the application.

2. MongoDB

We use the MongoDB database to store information from the application. It is deployed on Google Cloud through the MongoDB Atlas service. We use the MongoClient module, a Nodejs library, to manage the connection and interaction with the MongoDB database. We use this database to store collections of services, users, contracts and more.

3. Express

We used the middleware and routing features of Express to build the backend and communicate with the front of our application. We created different paths with Express’ router object which the frontend can access with an HTTP request call.

4. Google Cloud

Our server is hosted on the Google Cloud platform. We have also deployed our MongoDB database and Firebase Authentication on Google Cloud. Using the same hosting service, as well as servers in the same region, for our different services allows for better compatibility and performance. The Google Cloud service also gives us good control over the resources associated with the server, allowing them to be adjusted according to our performance needs.

5. Socket IO

Socket.IO allows two-way communication between client and server. It is mainly used for our instant messaging system. When sending a message, Socket.IO will emit to the right room on the server the message to be sent. Users connected to this room will then receive the message on the client side of the application. The message is sent and received instantly. We also share the contracts through Socket.IO so that users who have made or are about to make an agreement can see the state of the contract in place (open, in progress, complete or expired).

6. EOSJS

EOSJS is a Javascript library which provides an API for integrating with EOSIO-based blockchains using the EOSIO Nodeos RPC API. This library provides a JavaScript interface for interaction between end-user web applications and EOSIO blockchains. In our case, we used the library to interact between the blockchain and our NodeJs server. The server communicates directly to a contract that is on the blockchain under the account name: eosmartkplce.

7. ESR

The ESR protocol allows for an application to generate signature requests which can then be passed to signers (wallets) for signature creation. These signature requests can be used within URI links, QR Codes, or other transports between applications and signers.

8. EOS Smart Contracts

Smart Contracts are self-executing programs whose terms and conditions are written directly as code. On EOS, they are registered on the blockchain and executed on an EOSIO node. Their execution and resource consumption are handled just like a typical C++ application running on an OS. While written in C++, they are converted to web assembly before being sent to a node. We use the smart contracts as an escrow service for our marketplace. It allows for the management of the funds for a service that is being offered. The smart contract ensures that both parties are agreeing before transferring the funds. It also allows for the deal to be canceled under certain conditions. In the event that both parties are disagreeing, there is an arbiter assigned to every deal that can take care of that disagreement.