API RESTFUL
Bianca Hendy

O sistema é baseado em uma locadora de filmes ao qual tem 3 variáveis name, price e active, onde o name será o nome do filme, price o preço de custo e active se o filme pode ou não ser locado.
Há dois requests GET, sendo um que acessa o banco, mongoDB pelo mongoose, ao qual lista todos os dados existentes no banco.

GET	Listagem de todos os filmes
200	Lista os dados encontrados no banco
500	Ocorreu um erro no servidor

E o outro que irá listar um filme específico pelo id que está registrado no banco

GET :id	Lista um filme específico
200	Lista o filme do id
422	Não encontrou o filme com o id
500	Ocorreu um erro no servidor

Método POST irá criar um filme no banco de dados

POST	Cria um novo filme
200	Conseguiu criar o filme
500	Ocorreu um erro no servidor

Método PUT irá atualizar os dados dentro de uma requisição a partir do id do filme

PUT	Atualiza vários dados
200	Encontrou o filme pelo id e atualizou os dados
500	Ocorreu um erro no servidor

Também para atualização temos o método PATCH, caso precise atualizar apenas um dado do filme em questão

PATCH	Atualiza um dado
200	Conseguiu encontrar o filme e atualizou os dados
422	Não encontrou o filme ao qual está querendo ser atualizado (id)
500	Ocorreu um erro no servidor

Método DELETE que irá deletar um filme a partir do id, caso não encontre o id dará um erro específico

DELETE	Deleção de filme
200	Conseguiu deletar o filme
422	Não encontrou o filme para deleção
500	Ocorreu um erro no servidor

No trabalho foi utilizado middleware do express para que possa ler um json e possa receber como resposta um json também.
As renderizações podem ocorrer de várias formas do lado do servidor, universal, estática ou no lado do cliente, que é o caso da nossa aplicação devido que espera uma resposta como json.
Os frameworks disponibilizados para confecção do trabalho foi escolhido o express devido que é o mais conhecido e que se tem uma documentação muito abrangente, porém este não é o melhor.
Como vias de pesquisa foi feito um comparativo entre os frameworks fastify, express, hapi e KOA.
Exemplos de como seria um “Hello world” em cada framework.

Express

```
import express from "express";
const app = express();
const port = 3000;

app.get("/", (req, res) => {
  res.send("Hello World!");
});

app.listen(port, () => console.log("Listening on", port));
```

Fastify

```
import Fastify from "fastify";
const fastify = Fastify({ logger: false });

fastify.get("/", (request, reply) => {
  return "Hello world!";
});

fastify.listen({ port: 3000 });
```

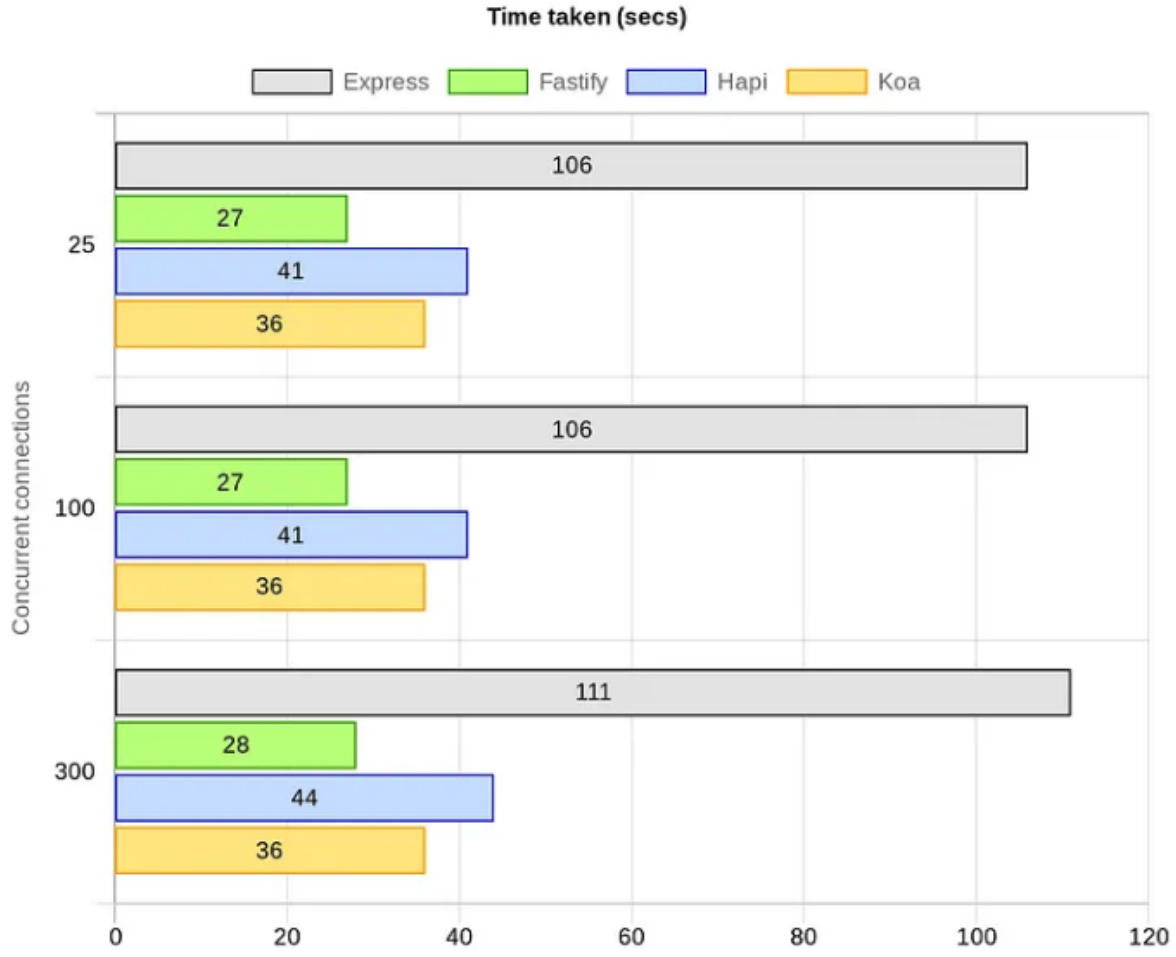
Hapi

```
import Hapi from "@hapi/hapi";
const server = Hapi.server({
  port: 3000,
  host: "localhost",
});

server.route({
  method: "GET",
  path: "/",
  handler: (request, h) => {
    return "Hello World!";
  },
});

server.start();
```

Koa

```
import Koa from "koa";
import Router from "@koa/router";

const app = new Koa();
const router = new Router();

router.get("/", (ctx, next) => {
  ctx.body = "Hello World!";
});

app
  .use(router.routes())
  .use(router.allowedMethods());

app.listen(3000);
```

O teste consistia em fazer 2M de requests e são repetidos para 25, 100 e 300 conexões simultâneas



Observa-se que o express foi o que mais demorou para finalizar a quantidade de requests enquanto o fastify foi um tanto mais rápido para a mesma execução.