**UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO**

ELTON PENICELA

eltonpenicela@gmail.com

Mestrado em Engenharia Informática e Tecnologia Web

**INTEGRAÇÃO DE SISTEMAS**

Link das do projecto completo no github

<https://github.com/epenicela/Sis_ApostasEuroMil.git>

1. **APLICAÇÕES CLIENTE REST | GRPC**

As aplicações cliente REST e gRPC são duas tecnologias amplamente utilizadas na construção de sistemas distribuídos e comunicação entre serviços. REST (Representational State Transfer) é um estilo arquitetônico que define um conjunto de princípios para o desenvolvimento de APIs (Application Programming Interfaces) baseadas em HTTP. Por outro lado, gRPC é um protocolo de comunicação remota desenvolvido pelo Google que utiliza o Protocol Buffers para serialização de dados e suporta diversas linguagens de programação.

Entretanto na presente, pretende-se criar dois projectos de integração de sistemas exemplificando as duas tecnologias, usando nodejs para backend (usando como tecnologias REST, e Grpc), vuejs para frontend.

* 1. **Aplicações REST**

Nsta estapa é criado um servidor Node.js que usa o framework Express para criar uma API RESTful que fornece acesso a recursos específicos através de endpoints HTTP. Incluindo as dependências necessárias, que são o *Axios* para fazer chamadas *HTTP* e o *Express* para criar o servidor.

Em seguida, há três rotas diferentes definidas com o método app.get(). Cada rota tem um endpoint diferente, com dois parâmetros de consulta: credit\_account\_id e value. As rotas fornecem acesso a diferentes recursos:

/check/:credit\_account\_id/ammount/:value: Esta rota é responsável por verificar se um cheque digital pode ser autenticado com um determinado credit\_account\_id e value. Ele usa o pacote Axios para fazer uma chamada para outro servidor que possui uma API para verificar a autenticidade do cheque. A resposta é analisada para verificar se o cheque digital é autêntico ou não e retorna a resposta correspondente como JSON.

/lua/:credit\_account\_id/ammount/:value: Esta rota é um endpoint de teste que sempre retorna um objeto JSON específico.

/lista/:credit\_account\_id/ammount/:value: Esta rota é responsável por retornar a primeira entrada em uma lista de cheques digitais autenticados. A resposta é novamente analisada para recuperar o primeiro cheque digital autenticado e é retornada como JSON.

* 1. **Aplicações GRPC**

O gRPC é um sistema de chamada de procedimento remoto de alta performance e multiplataforma que permite a comunicação entre diferentes serviços em uma rede.

O código começa importando os módulos necessários, como o gRPC, o sqlite3 e o arquivo proto que contém a definição do serviço. O arquivo proto é carregado usando o método grpc.load() e o serviço é adicionado ao servidor usando o método server.addService().

O servidor define quatro métodos RPC (Remote Procedure Call): list, insert, update e delete. Esses métodos são responsáveis por consultar, inserir, atualizar e excluir registros de uma tabela chamada apostas que é criada em um banco de dados SQLite em memória usando o objeto sqlite3.Database.

Os métodos RPC usam o objeto call para obter informações da requisição e o objeto callback para enviar a resposta. Quando ocorre um erro, ele é passado para o callback como o primeiro parâmetro, caso contrário, a resposta é passada como o segundo parâmetro.

O servidor é iniciado usando o método server.start() e está vinculado ao endereço 127.0.0.1:50051. O método console.log é usado para imprimir uma mensagem informando que o servidor está ativo.

Desseguida foi criado um método em JavaScript é nodejs responsavel por criar um cliente gRPC (Remote Procedure Call) para se comunicar com um servidor que implementa o protocolo definido no arquivo "euromil.proto":

1. const grpc = require('grpc'); - essa linha importa a biblioteca gRPC do Node.js, que será usada para criar o cliente.
2. const PROTO\_PATH = './euromil.proto' - essa linha define o caminho relativo para o arquivo de definição do protocolo gRPC.
3. const EuroMil = grpc.load(PROTO\_PATH).EuroMil - essa linha carrega o arquivo de definição do protocolo gRPC usando a função load da biblioteca gRPC. Em seguida, ela retorna um objeto com a definição do protocolo gRPC, e a propriedade EuroMil é atribuída à constante EuroMil. O objeto retornado contém a definição dos serviços, métodos, mensagens e outras informações necessárias para criar o cliente gRPC.
4. const client = new EuroMil('localhost:50051', grpc.credentials.createInsecure()) - essa linha cria uma instância do cliente gRPC, usando a definição de protocolo carregada anteriormente na constante EuroMil. O primeiro parâmetro é o endereço do servidor gRPC, que nesse caso é localhost:50051. O segundo parâmetro é a autenticação usada para se conectar ao servidor, que nesse caso é grpc.credentials.createInsecure(), que significa que a conexão não será criptografada.
5. module.exports = client - essa linha exporta o cliente gRPC criado para que possa ser usado por outros módulos ou arquivos. Isso permite que o cliente seja importado em outras partes do código e usado para se comunicar com o servidor gRPC.
   1. **Frontend**

import axios from 'axios';

import { GrpcWebClient } from "grpc-web-client";

async registrarAposta() {

const respostaCrediBank = await axios.get(`http://localhost:3000/check/${this.checkid}/ammount/${this.value},{

valor: 10, idConta: this.checkid

})

this.chequeDigital = respostaCrediBank.data.chequeDigital;

if (respostaCrediBank.status) {

const chequeDigital = respostaCrediBank.data.checkId;

client.getData(respostaCrediBank, function (err, response) {

if (err) {

console.error(err);

} else {

enviarNovaAposta();

}

});

} else {

alert('Não foi possível obter o cheque digital. tarde.');

}

}

Este é um exemplo de um componente Vue.js que possui um método registrarAposta() que é chamado quando o usuário realiza uma ação específica (como clicar em um botão).

O objetivo do método é entrar em contato com o sistema CrediBank e solicitar a emissão de um cheque digital no valor de 10 créditos. O método usa o pacote axios para fazer uma solicitação GET para http://localhost:3000/check/${this.checkid}/ammount/${this.value}, com valor e idConta fornecidos como parâmetros. O resultado da solicitação é armazenado na variável respostaCrediBank.

Se a solicitação for bem-sucedida (ou seja, se o status da resposta for verdadeiro), o cheque digital é armazenado em this.chequeDigital. Em seguida, o método entra em contato com o sistema EuroMilRegister para inserir a aposta e o cheque digital. Um alerta informa ao usuário que a nova aposta foi inserida com sucesso.

Se a solicitação ao sistema CrediBank falhar, o usuário é informado por meio de um alerta que não foi possível obter o cheque digital e é recomendado que tente novamente mais tarde.

enviarNovaAposta(respostaCrediBank) {

const client = new GrpcWebClient({

host: "127.0.0.1:50051", protoPath: "../euromil.proto", transport: "http"});

client.rpcCall("insert", aposta, {}, (err, response) => {

if (err) {

console.error(err);

} else {

alert("NOVA APOSTA INSERIDA COM SUCCESSO");

}

});

}

Em resumo, esta função envia uma nova aposta para um serviço gRPC-Web e exibe uma mensagem de sucesso ou erro, dependendo da resposta recebida.

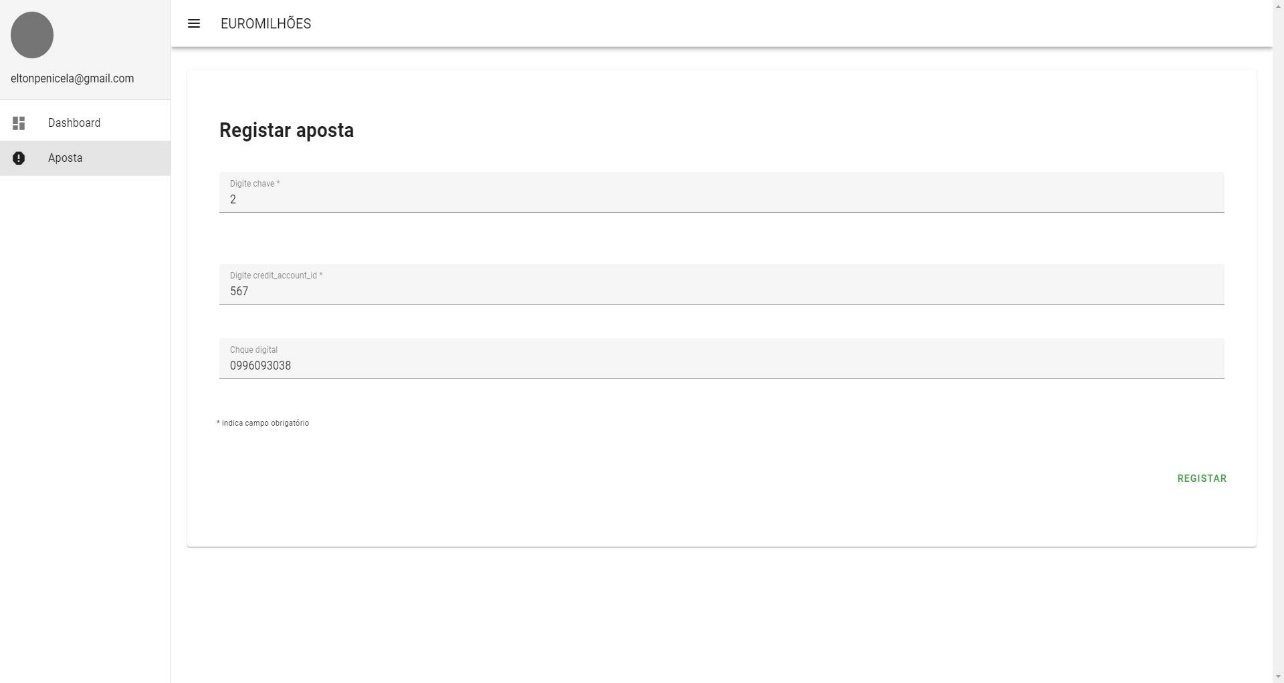


Figura 1: Aplicação de aposta (Euro milhões)

**Referências**

Richardson, L., Amundsen, M., & Ruby, S. (2013). RESTful web APIs: Services for a changing world. O'Reilly Media, Inc.

Sampaio, A., & Pinto, M. (2019). A Comparative Analysis of REST and gRPC for Microservices. In Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (pp. 307-314).

gRPC documentation: <https://grpc.io/docs/>

Conceitos básicos sobre gRPCs, <https://grpc.io/docs/guides/concepts/>