

A solution for Data Integrity in Relational Database Security

Bernardo José Azevedo Queirós Nogueira - A47334

Rui Miguel Silva Aires - A45296

Trabalho realizado sob a orientação de

Prof. Rui Alves

Prof. Tiago Pedrosa

Licenciatura em Engenharia Informática

2023-2024

A solution for Data Integrity in Relational Database Security

Relatório da UC de Projeto
Licenciatura em Engenharia Informática
Escola Superior de Tecnologia e Gestão

Bernardo José Azevedo Queirós Nogueira - A47334

Rui Miguel Silva Aires - A45296

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Declaro que o trabalho descrito neste relatório é da minha autoria e é
da minha vontade que o mesmo seja submetido a avaliação.

Bernardo José Azevedo Queirós Nogueira - A47334

Rui Miguel Silva Aires - A45296

Dedicatória

Dedico este trabalho a toda a gente que sempre acreditou em mim e que esteve no meu lado quando mais precisei, aos meus pais por nunca me deixaram ir abaixo e aos meus amigos que estiveram lá sempre para mim.

Dedico também ao professor Rui Alves que sempre acreditou em nós, mesmo quando as coisas não estavam tão favoráveis para o nosso lado.

Apesar de já ter falado nos meus amigos em geral, queria dedicar também ao meu parceiro do projeto, o Rui Aires porque sem ele nada disto seria possível.

Em suma, esta dedicatória é para todas as pessoas aqui referidas portanto, um enorme obrigado a todos vocês. (Bernardo Nogueira)

Dedico este trabalho , a toda à minha família inclusive aos meus pais que sem a força, exemplo, apoio durante esta caminhada, pelas palavras de conforto nos dias mais difíceis e por nunca me terem deixado desistir de tudo o que sempre sonhei.

Também dedico a todos os meus amigos e pessoas que se cruzaram comigo nesta aventura universitária, sem eles nada disto seria possível.

E a esta cidade incrível que foi durante estes últimos anos a minha segunda casa.

Por fim, dedicar também a paciência, trabalho e dedicação que os nossos professores orientadores tiveram nesta jornada magnífica.

Posto isto, um grande obrigado a todos. (Rui Aires)

Por fim mas não menos importante, queríamos agradecer ao stor Rui Alves mais uma vez pela paciência, compreensão, pelo conhecimento passado e pela ajuda inalcançável.

Agradecimentos

Agradecemos aos nossos colegas de curso que muitas vezes foram eles que nos deram forças para continuar com este projeto e foram a melhor companhia possível ao longo destes 4 anos de licenciatura . Agradecemos em especial aos nossos colegas Ivo Pimenta, Beatriz Alves, Nelson Guedes, Pedro Gonçalves e o José Saraiva que nos deram o suporte que necessitamos durante todo este tempo.

Agradecemos ao Professor Rui Alves, nosso orientador, por todo o tempo investido em nós e neste projeto, por todos os conhecimentos passados e por todas as explicações e ajuda que nos facultou.

Sem ele nada disto seria sequer possível, a paciência e o empenho foram fundamentais para a nossa aprendizagem e sucesso. Estamos profundamente agradecidos pelo seu apoio constante e pela sua orientação.

Resumo

A encriptação End-to-end é considerada uma criptografia assimétrica. A criptografia assimétrica criptografa e descriptografa dados usando duas chaves criptográficas: chaves públicas e privadas. A chave pública é usada para criptografar os dados e a chave privada para descriptografar. Como o nome sugere, a chave privada foi projetada para permanecer privada, de modo que apenas o destinatário alvo, possa decifrar os dados.

O que difere da encriptação End-to-end para outros sistemas de criptografia é que há apenas dois terminais, o sender e o receiver, estes são capazes de descriptografar e ler a mensagem para além desta criptografia usar tanto a chave pública como privada. A criptografia simétrica, também é conhecida por criptografia de chave única apenas usa uma chave única para criptografar e descriptografar informações.

Desta forma o objetivo aqui é idealizar/construir uma solução, utilizando tanto as chaves públicas como chaves primárias, analisando a assinatura dos dados assim permitindo apenas o acesso a clientes devidamente autenticados, impedindo assim qualquer tipo de fuga de dados.

A exploração desta proposta fará com que sistemas de informações sejam mais protegidos, que os clientes têm mais privacidade, confidencialidade e proteção na troca de informações com outros clientes, também impedindo ataques de hackers, Man-in-the-middle entre outros.

Palavras-chave: Criptografia, Proxy, Autenticação, Chaves(Public/Private) .

Abstract

End-to-end encryption is considered asymmetric cryptography. Asymmetric encryption encrypts and decrypts data using two cryptographic keys: public and private keys. The public key is used to encrypt the data and the private key to decrypt it. As the name suggests, the private key is designed to remain private so that only the target recipient can decrypt the data.

What makes end-to-end encryption different from other encryption systems is that there are only two terminals, the sender and the receiver. two terminals, the sender and the receiver, who are able to decrypt and read the message, and this encryption uses both public and private keys. Symmetric cryptography, also known as single key cryptography, only uses a single key to encrypt and decrypt information.

The aim here is to devise/build a solution using both public and primary keys, analyzing the signature of the data, thus allowing access only to duly authenticated clients, thus preventing any type of data leakage.

The exploitation of this proposal will make information systems more secure, giving customers have more privacy, confidentiality and protection when exchanging information with other customers, also preventing hacker attacks, Man-in-the-middle and others.

Keywords: Cryptography, Proxy, Authentication, Keys(Public/Private)

Conteúdo

1	Introdução	1
1.1	Enquadramento	2
1.2	Objetivos	2
1.3	Estrutura do Documento	3
2	Análise do Problema	5
2.1	Estado da arte	6
2.2	Tecnologias e Ferramentas	8
2.3	Vmware	9
2.4	Visual Studio Code	10
2.5	Mysql Workbench	10
2.6	WireShark	10
2.7	Docker	10
2.8	Mariadb	11
2.9	Proxysql	11
3	Implementação	13
3.1	Pesquisa por autentication token and asymmetric key DB	18
3.2	Wireshark + Mecanismos de comunicação BD	19
3.3	Criação de um exemplo em C	24
3.4	Proxysql	40

4 Resultados	51
4.1 Greeting message	51
4.2 ProxySql	53
4.3 Interseção de Queries	55
5 Conclusão	57
A Proposta Original do Projeto	A1
B Outro(s) Apêndice(s)	B1

Lista de Figuras

2.1 Ilustração de uma end-to-end encryption	7
3.1 Ilustração da porta da MariaDB	14
3.2 Ilustração da conexão do cliente á MariaDB	14
3.3 Ilustração do sequelize1	15
3.4 Ilustração do sequelize2	16
3.5 Ilustração da restapi e endpoint1	17
3.6 Ilustração da restapi e endpoint2	17
3.7 Ilustração da captura do wireshark	19
3.8 Ilustração do Makefile (ver apêndiceB)	25
3.9 Ilustração do client.c1 (ver apêndiceB)	25
3.10 Ilustração do client.c2	26
3.11 Ilustração do client.c3	27
3.12 Ilustração do client.c4	28
3.13 Ilustração do client.c5	29
3.14 Ilustração do server.c1 (ver apêndiceB)	30
3.15 Ilustração do server.c2	31
3.16 Ilustração do server.c3	32
3.17 Ilustração do server.c4	33
3.18 Ilustração do server.c5	34
3.19 Ilustração do server.c6	35
3.20 Ilustração do server.c7	36

3.21 Ilustração da estrutura para capturar o greeting message1	38
3.22 Ilustração da estrutura para capturar o greeting message2	38
3.23 Ilustração da captura do greeting message	39
3.24 Ilustração da criação das instâncias dentro do proxy	42
3.25 Ilustração do código da criação dos containers1	43
3.26 Ilustração do código da criação dos containers2	44
3.27 Ilustração do código da criação dos containers3	44
3.28 Ilustração do código da criação dos containers4	44
3.29 Ilustração do arranque dos containers	45
3.30 Ilustração da conexão do mysqlworkbench ao proxysql	46
3.31 Ilustração de um insert into através do workbench	47
3.32 Ilustração dos dados existentes na primary	48
3.33 Ilustração dos dados existentes na réplica1	48
3.34 Ilustração dos dados existentes na réplica1.2	49
3.35 Ilustração dos dados existentes na réplica2	49
4.1 Ilustração do funcionamento do código para obter o greeting message	52
4.2 Ilustração da captura do pacote greeting message	53
4.3 Ilustração do funcionamento das instâncias1	54
4.4 Ilustração do funcionamento das instâncias2	54
4.5 Ilustração do script lua	55

Capítulo 1

Introdução

Existe vários tipos encriptação que podemos utilizar para proteção de dados, encriptações simétricas ou assimétricas. As aplicações conhecidas como o Facebook, WhatsApp e o Zoom utilizam a encriptação assimétrica pois devido a sua segurança e a confidencialidade que dá aos utilizadores, permitindo-lhes ter a privacidade sobre as mensagens e informações que este passa para outro utilizador.

O importante neste tipo de encriptação é saber bem os conceitos e como utiliza-los, neste tipo de encriptação usamos dois tipos de chaves, pública e privada, uma vez que a chave pública é partilhada com outra pessoa/utilizador, o outro utilizador pode encriptar uma mensagem/informação e enviá-la para o proprietário da chave pública. A mensagem a partir de agora só pode ser desencriptada utilizando a chave privada correspondente.

Este tipo de encriptação tem várias vantagens para que o uso desta mesma encriptação seja usada por estas grandes aplicações. As vantagens são as seguintes:

- Maior segurança em trânsito;
- Inviolável.

Este tipo de encriptação tem também várias desvantagens, pois todas as encriptações têm as suas desvantagens. As desvantagens são as seguintes:

- Complexidade na definição dos endpoints;

- Segurança dos endpoints;
- Metadados visíveis.

O objetivo aqui é conseguir construir uma solução que utilize estas chaves aqui em cima citadas de forma que os clientes consigam ter a sua confidencialidade e privacidade asseguradas sem que haja qualquer tipo de fuga de informação e apenas permitindo o acesso a utilizadores devidamente autenticados fazendo com que este sistema seja inviolável.

1.1 Enquadramento

O projeto apresentado encontra-se inserido no tema de cibersegurança (ver apêndiceA), pois o seu principal objetivo passa por criar uma solução recorrendo ao uso das chaves (públicas/privadas), que analise a assinatura de cada trama de dados e que apenas permite acesso aos utilizadores autenticados. Além disso este projeto tem potencial para conseguir-se fazer algo extra para além do objetivo principal.

1.2 Objetivos

O objetivo desta proposta de projeto, tem como objetivo criar uma encriptação assimétrica de forma que os utilizadores têm a proteção e a privacidade devida, usando tanto a chave pública como privada pois, são necessárias para esta implementação. Para alcançar este propósito, foram definidos os seguintes objetivos:

- Instalar um servidor mysql;
- Criar uma RestAPI com um endpoint;
- Pesquisar pela autenticação entre servidor web (ou outro cliente) e o servidor de base de dados relacional;
- Criar um receiver funcional;

- Concluir, por fim criar um sistema de encriptação assimétrica como pedido já com a utilização da chave pública e privada.

1.3 Estrutura do Documento

Este relatório está organizado da seguinte forma para proporcionar uma clara compreensão sobre a estrutura adotada neste relatório:

- Capítulo 1: Aborda uma breve introdução sobre o tema do projeto.
- Capítulo 2: Aborda o levantamento/pesquisa do estado de arte e explicação das tecnologias usadas.
- Capítulo 3: Aborda detalhadamente o problema em questão e a proposta de resolução do mesmo contendo também, a explicação das tecnologias usadas.
- Capítulo 4: Aborda os testes realizados e os resultados obtidos para a verificação da conclusão dos objetivos e resolução do problema.
- Capítulo 5: Aborda as conclusões finais.
- Apêndice A: Contém a proposta apresentada do projeto.
- Apêndice B: Contém excertos de código utilizados na realização do projeto.

Capítulo 2

Análise do Problema

A análise de problemas é uma metodologia utilizada para resolução de problemas complexos em processos, produtos e serviços em organizações. Trata-se de uma metodologia para melhorias consideradas radicais, que contrasta com as metodologias de melhoria incremental [1]. Trata-se de um processo que envolve recolha de informações, identificação dos principais problemas e a proposta de soluções eficazes.

Esta metodologia é um caminho ordenado, composto de passos e sub-passos pré-definidos para a escolha de um problema, análise de suas causas, determinação e planejamento de um conjunto de ações que consistem uma solução, verificação do resultado da solução e realimentação do processo para a melhoria do aprendizado e da própria forma de aplicação em ciclos posteriores [1].

Este processo envolve várias etapas para assegurar a conclusão do processo. A primeira etapa é identificar o problema, assim definindo claramente o problema que tem de ser resolvido. Em seguida, é necessário reunir informações importantes e relevantes sobre o problema para entender quais são as causas e as suas consequências.

Este tipo de abordagem não apenas ajuda e resolve problemas em questão, como fortalece também a capacidade de enfrentar problemas futuros que possam surgir de forma mais acertada e consciente.

2.1 Estado da arte

A encriptação assimétrica contém vários conceitos que para quem utiliza este tipo de encriptação necessitam de estar bem cimentados para que se possa entender bem o próprio conceito da encriptação, logo o nosso primeiro pensamento foi pesquisar de forma objetiva e concreta informações sobre este tema, tentando sempre ser o mais objetivo possível, focando-nos sempre no cerne da questão.

O nosso objetivo é criar uma solução que recorra a encriptação assimétrica, posto isto fomos a procura de alguma informação que se adequasse ao que procuravamos, durante esta pesquisa encontramos um artigo com o seguinte tema: End-to-End Database Software Security [2]. O artigo fala sobre construir uma segurança End-to-end pois maior parte dos softwares de gestão de base de dados dão proteção aos dados em trânsito e proteção dos dados no lado do servidor, este artigo se propõe a proteger os dados em trânsito, no lado do servidor e também os dados em repouso do lado do cliente da aplicação. Este artigo encaixava-se na proposta do nosso projeto pois este era o nosso objetivo, mesmo assim fomos a procura de outro tipo de artigos que nos poderiam dar mais alguma ajuda de forma a que percebessemos na totalidade o que tínhamos que fazer para chegar ao resultado que pretendíamos.

Continuamos a nossa pesquisa intensiva sobre o nosso tema para tentar entender mais sobre os próprios conceitos do nosso tema. A nossa pesquisa deu frutos e encontramos mais artigos sobre o nosso tema, todos abordam mesmo tema do artigo referido em cima, encontramos primeiro este artigo com o seguinte tema: end-to-end encryption (E2EE) [3], este artigo é mais um artigo autoexplicativo do que um artigo sobre algo desenvolvido por alguém, neste caso este artigo explica o conceito end-to-end encryption, fala um bocado sobre o mesmo falando inclusive das suas vantagens e desvantagens sobre este tipo de encriptação, em seguida a nossa pesquisa levou-nos a um vídeo de youtube[4] que também ajuda a explicar e a entender mais os conceitos e ideias sobre a end-to-end encryption, este vídeo foi uma grande ajuda pois fez com que ficasse tudo muito mais claro e cimentado nas nossas cabeças qual era o objetivo e melhorou o nosso entendimento sobre este tipo

de encriptação. Continuamos a nossa pesquisa intensiva sobre o tema e acabamos por encontrar um link do stackoverflow [5], dentro deste link um utilizador pedia ajuda para tentar continuar o seu desenvolvimento num projeto que tem semelhanças com o nosso, neste caso o que nos chamou a atenção foi que este utilizador recebeu uma resposta de outro utilizador em que este responde com uma imagem que nos deu mais uma luz sobre o nosso tema em questão.

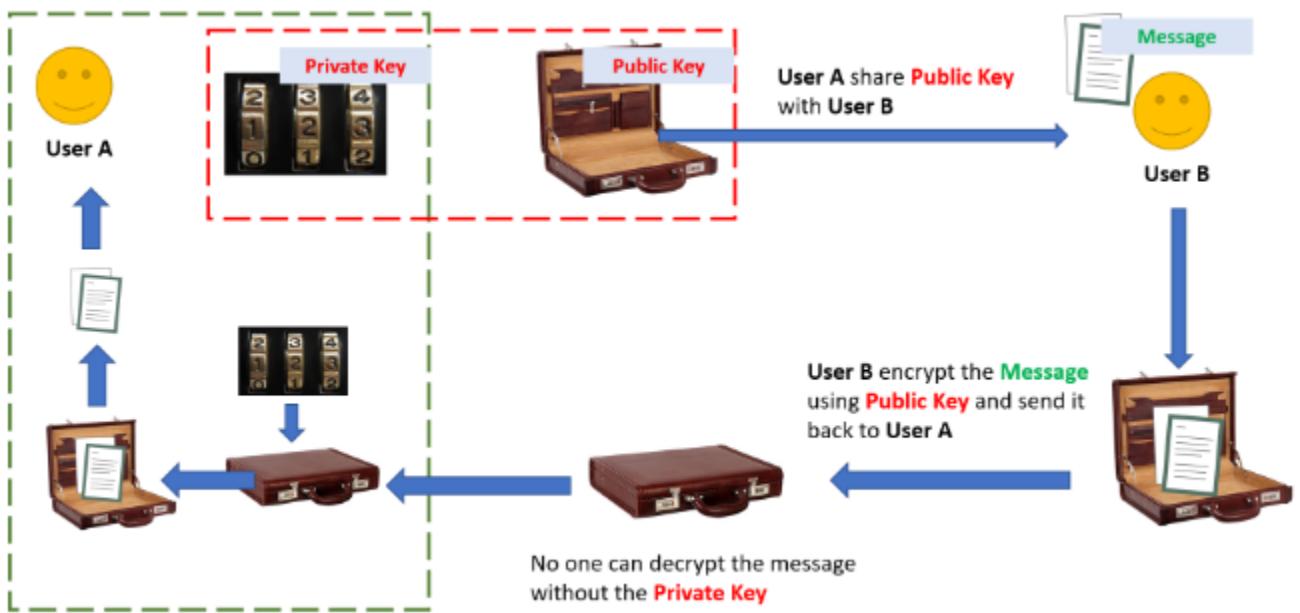


Figura 2.1: Ilustração de uma end-to-end encryption

2.2 Tecnologias e Ferramentas

Este projeto exegiu o uso de várias tecnologias e ferramentas para o desenvolvimento do nosso projeto, iremos falar deles separadamente e explicar detalhadamente a função que cada um desempenhou neste projeto, pois cada ferramenta/tecnologia desempenhou um papel crucial neste projeto.

As tecnologias usadas neste projeto foram as seguintes:

- Vmware;
- Visual Studio Code;
- Mysql Workbench;
- WireShark;
- Docker;
- Mariadb;
- Proxysql.

Estas ferramentas foram usadas em várias partes ao longo de todo o projeto, conseguindo com estas tentar desenvolver a solução pretendida.

2.3 Vmware

A primeira ferramenta utilizada foi a Vmware Workstation Pro. A função da Vmware é produzir um software e serviços para a computação em nuvem e virtualização, o que permite utilizar as máquinas virtuais [6].

Neste projeto basicamente usamos desde o início a Vmware e utilizamos o sistema operativo debian 12, pois como tínhamos de utilizar a mariadb e o proxysql foi fácil de chegar à conclusão que precisaríamos de utilizar a Vmware, a escolha do debian como sistema operativo foi devido a já termos trabalhado num passado recente com este sistema operativo em outras cadeiras da licenciatura, então juntamos o útil ao agradável. Além disso a ferramenta snapshot ajudou imenso pois tínhamos sempre um backup caso algo corresse mal.

A Vmware foi importantíssima para a realização deste projeto e sem dúvida a ferramenta mais utilizada ao longe deste projeto.

2.4 Visual Studio Code

O Visual Studio Code é um editor de código-fonte, super versátil pois permite trabalhar em várias linguagens diferentes, neste projeto as linguagens que mais utilizamos foi C e javascript. A linguagem C foi utilizado para criar estruturas com sockets entre outras coisas, além disso utilizamos javascript para fazer um Sequelize e para criar um cliente que se conseguisse ligar a MariaDB, o que nos foi muito útil neste projeto.

2.5 Mysql Workbench

O Mysql Workbench foi super importante no desenvolvimento pois este serviu para escrever e executar consultas sql nos bancos de dados em que posteriormente, essas consultas sql seriam monitoradas pelo wireshark e foi utilizado como uma ferramenta para se conectar ao proxysql, assim utilizando o workbench para escrever e executar queries.

2.6 WireShark

O WireShark é uma ferramenta de análise de pacotes e sniffer de rede. Ele captura o tráfego de rede na rede local e armazena esses dados para análise offline.

Esta foi mais uma ferramenta muito relevante para o nosso projeto, o objetivo seria capturar o tráfego entre o nodejs e a BD. Escolhemos esta ferramenta de captura de tráfego pois já tínhamos alguma experiência no uso da mesma num passado recente pois foi necessário em outras cadeiras.

2.7 Docker

O Docker é uma plataforma de código aberto que permite aos desenvolvedores construir, implementar, executar, atualizar e gerenciar containers. Esta ferramenta foi utilizada não tanto no início do nosso projeto mas sim mais na parte final aos criarmos um container principal e as suas réplicas, estando tudo interligado com o proxysql.

2.8 Mariadb

MariaDB é uma base de dados de código aberto para ajudar a armazenar e organizar dados. Esta ferramenta sempre foi essencial desde o início pois foi utilizada como a nossa base de dados desde o início do projeto até ao fim do mesmo.

2.9 Proxysql

O proxysql é um servidor proxy com reconhecimento de Sql que pode ser posicionado entre o aplicativo e o seu banco de dados. Esta ferramenta foi utilizada e necessária na parte final do projeto pois a abordagem do mesmo foi redirecionada para outro caminho mais intuitivo de forma a conseguirmos chegar ao nosso objetivo final.

Capítulo 3

Implementação

Antes de começarmos a implementação, como referimos em cima no estado de arte, o objetivo inicial foi consolidar bem as ideias e os conceitos da nossa proposta, pesquisar a fundo foi a nossa primeira tarefa que já foi referida no estado de arte detalhadamente. Depois desta primeira tarefa inicial, tivemos a nossa primeira tarefa de implementação, os primeiros objetivos foram os seguintes:

- Instalar um servidor mysql;
- Criar uma tabela simples, deve-se usar a PK como um auto incremental.;
- Criar uma RestAPI com um endpoint pode ser um get;
- Por cada pedido, que deve ser pedido pelo browser, ou postman deve ser inserido um registo na tabela criada;
- Dentro do endpoint deve-se gerar uma entity class com base na tabela da BD. Os campos que adicionarem serão os mesmos em cada nova inscrição so mudará o id que é autoincremental;

Aqui foi onde começamos a implementar o projeto, primeiro para podermos fazer o que o professor pediu precisavamos de escolher uma linguagem de programação para poder começar, aqui tivemos liberdade de escolhermos a que quisesssemos sem limitações

e acabamos por escolher a linguagem JavaScript. Primeiro passo foi instalar um servidor mysql e pensamos logo no MariaDB, instalamos o MariaDB e mudamos nos ficheiros de configuração para a porta 3306, pois será a porta onde iremos trabalhar sempre.

```
MariaDB [(none)]> SHOW VARIABLES LIKE 'port';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| port          | 3306  |
+-----+-----+
1 row in set (0,004 sec)
```

Figura 3.1: Ilustração da porta da MariaDB

Depois da instalação da MariaDB e da mudança de porta para a 3306 era necessário fazer a ligação a MariaDB através do código que fizemos em JavaScript, este código fará um papel de cliente que se conecta a nossa base de dados, neste caso a MariaDB (ver apêndice B).



```
JS db.js  x
JS dbjs > ...
1
2 const mysql = require('mysql');
3
4 // Configuração da conexão com o MySQL
5 const connection = mysql.createConnection({
6   host: 'localhost',
7   user: 'root',
8   password: 'Rui.aires1',
9   database: 'projeto',
10  port: 3306, // Porta configurada no servidor.c
11 });
12
13 // Adiciona um listener para o evento 'connect'
14 connection.connect(() => {
15   console.log('Conectado ao MySQL com sucesso!');
16
17   // Aqui você pode realizar consultas, inserções, atualizações, etc.
18   // Exemplo de consulta simples:
19   connection.query('SELECT 1 + 1 AS resultado', (queryErr, results) => {
20     if (queryErr) {
21       console.error('Erro na consulta:', queryErr.stack);
22       return;
23     }
24
25     console.log('Resultado da consulta:', results[0].resultado);
26
27     // Fecha a conexão após a consulta
28     connection.end((endErr) => {
29       if (endErr) {
30         console.error('Erro ao fechar a conexão:', endErr.stack);
31         return;
32       }
33
34       console.log('Conexão fechada com sucesso!');
35     });
36   });
37 });
38
```

Figura 3.2: Ilustração da conexão do cliente á MariaDB

O passo seguinte foi criar um sequelize, o sequelize vai nos dar a possibilidade de criar classes e estas vão ficar guardadas no nosso banco de dados. Assim usamos o sequelize para cirar a tabela (ver apêndice B).

```
JS many_to_many.js X
JS many_to_many.js > [0] sequelize
1  const {Sequelize, DataTypes} = require("sequelize");
2
3  const sequelize = new Sequelize(
4    'exemplodb',
5    'root',
6    'Rui.aires1',
7    {
8      host: 'localhost',
9      dialect: "mysql"
10    }
11  );
12
13  sequelize.authenticate().then(() => {
14    console.log('Connection has been established successfully.');
15  }).catch((error) => {
16    console.error('Unable to connect to the database: ', error);
17  });
18
19  const Student = sequelize.define("students", {
20    student_id: {
21      type: DataTypes.UUID,
22      defaultValue: DataTypes.UUIDV4,
23    },
24    name: {
25      type: DataTypes.STRING,
26      allowNull: false
27    }
28  });
29
30  const Course = sequelize.define("courses", {
31    course_name: {
32      type: DataTypes.STRING,
33      allowNull: false
34    }
35  });
36
37  const StudentCourse = sequelize.define('StudentCourse', {
38    id: {
39      type: DataTypes.INTEGER,
40      primaryKey: true,
41      autoIncrement: true,
42      allowNull: false
43    }
44  });
45
46  const course_data = [
47    {course_name : "Science"}, 
48    {course_name : "Maths"}, 
49    {course_name : "History"}]
```

Figura 3.3: Ilustração do sequelize1

```
JS many_to_manyjs X
js many_to_many.js > [sequelize
51  const student_data = [
52    {name : "John Baker", courseId: 2},
53    {name : "Max Butler", courseId: 1},
54    {name : "Ryan Fisher", courseId: 3},
55    {name : "Robert Gray", courseId: 2},
56    {name : "Sam Lewis", courseId: 1}
57  ]
58
59  const student_course_data = [
60    {studentId : 1, courseId: 1},
61    {studentId : 2, courseId: 1},
62    {studentId : 2, courseId: 3},
63    {studentId : 3, courseId: 2},
64    {studentId : 1, courseId: 2},
65  ]
66
67
68 Course.belongsToMany(Student, { through: 'StudentCourse'})
69 Student.belongsToMany(Course, { through: 'StudentCourse'})
70
71 sequelize.sync({ force: true }).then(() => {
72   Course.bulkCreate(course_data, { validate: true }).then(() => {
73     Student.bulkCreate(student_data, { validate: true }).then(() => {
74       StudentCourse.bulkCreate(student_course_data, { validate: true }).then(() => {
75         Course.findAll({
76           include: [
77             {
78               model: Student,
79             }
80           ],
81         }).then(result => {
82           console.log(result);
83         }).catch((error) => {
84           console.error('Failed to retrieve data : ', error);
85         });
86       }).catch((error) => {
87         console.log(error);
88       });
89     }).catch((error) => {
90       console.log(error);
91     });
92   }).catch((error) => {
93     console.error('Unable to create table : ', error);
94   });
95 });
96
97 }).catch((error) => {
98   console.error('Unable to create table : ', error);
99 });


```

Figura 3.4: Ilustração do sequelize2

No mesmo código criamos um app.js onde dentro desse mesmo ficheiro criamos uma RestApi com endpoint sendo este endpoint um get pois, foi o sugerido pelo professor. Além disso, dentro deste ficheiro app.js também fizemos um código que insere um registo na tabela como pedido pelo professor (ver apêndiceB).

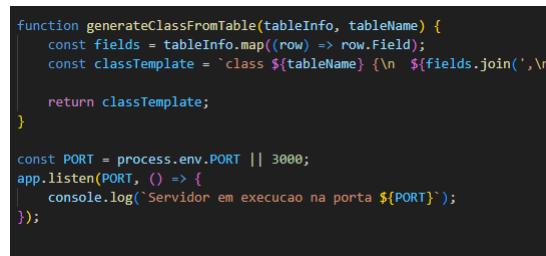


```

JS app.js
JS app.js > app.get('/inserir-registro') callback > generateEntityClass('Exemplo') callback
1 // api.js
2 const express = require('express');
3 const db = require('../db'); // Importa a configuração da base de dados
4
5 const app = express();
6 app.use(express.json());
7
8 // Endpoint GET que insere um registo na tabela e gera uma classe de entidade
9 app.get('/inserir-registro', (req, res) => {
10
11     // Gera uma classe de entidade com base na tabela
12     generateEntityClass('Exemplo', (entityClass) => {
13         const registro = {
14             nome: 'Exemplo',
15             idade: 30,
16             email: 'exemplo@email.com'
17         };
18
19         // Insira um registo na tabela
20         const sql = 'INSERT INTO Exemplo (nome, idade, email) VALUES (?, ?, ?)';
21         db.query(sql, [registro.nome, registro.idade, registro.email], (err, result) => {
22             if (err) {
23                 console.error('Erro ao inserir registo: ' + err);
24                 res.status(500).send('Erro interno do servidor');
25             } else {
26                 console.log('Registo inserido com sucesso');
27                 res.status(200).send(entityClass);
28             }
29         });
30     });
31 });
32
33 function generateEntityClass(tableName, callback) {
34
35     // Recuperar informações da tabela para gerar a classe de entidade
36     db.query(`DESCRIBE ${tableName}`, (err, result) => {
37         if (err) {
38             console.error('Erro ao obter informações da tabela: ' + err);
39             callback('');
40         } else {
41             const entityClass = generateClassFromTable(result, tableName);
42             callback(entityClass);
43         }
44     });
45 }
46

```

Figura 3.5: Ilustração da restapi e endpoint1



```

function generateClassFromTable(tableInfo, tableName) {
    const fields = tableInfo.map((row) => row.Field);
    const classTemplate = `class ${tableName} {${fields.join(',')}\n}`;
    return classTemplate;
}

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Servidor em execução na porta ${PORT}`);
});

```

Figura 3.6: Ilustração da restapi e endpoint2

Por fim, ligamos o workbench a porta 3306 e ao ip 127.0.0.1.

3.1 Pesquisa por authentication token and asymmetric key DB

Depois da primeira etapa ter sido feito precisavamos agora de pesquisar por uma forma de autenticação e pensamos na authentication token, então aqui deixamos em stand by a parte de implementação e voltamos a fazer pesquisas intensivas.

Aqui surgiu o nosso primeiro problema, fizemos uma pesquisa intensiva sobre a authentication token e encontramos alguns links que achamos que nos podia ajudar a entender melhor o objetivo da authentication token mas no fim não chegamos a nenhuma conclusão e descartamos esta ideia, mesmo assim vale ressaltar que encontramos informação que até podia ser útil mas pelo menos na nossa cabeça essa informação não nos fez perceber o conceito da forma que pretendíamos. Um dos links foi do stackoverflow com o seguinte tema: Token-based authentication: Application vs Server[7], o utilizador aqui tem uma dúvida sobre a utilização do token na sua aplicação. Para além deste link temos mais alguns links que apesar de conterem informações que nos podiam ser úteis, no fundo não era bem isto que procurávamos neste momento do projeto.

Ainda assim vale ressaltar todos os links encontrados apesar de não serem bem o que procuramos, pois o que pretendíamos era garantir a autenticidade dos dados que saiem do servidor web ou de outro cliente para o servidor de BD.

- Types of Database Encryption: Best Practices for Securing Your Data[8];
- SQL Server Best Practices: Using Asymmetric Keys to Implement Column Encryption [9];
- Database Encryption Using Asymmetric Keys: A Case Study [10];
- Database Encryption Using Asymmetric Keys: A Case Study [11];

- Asymmetric keys used by the DBMS for encryption of sensitive data should use DoD PKI Certificates [12];
- Using a MySQL Keyring SECRET and Asymmetric Encryption [13].

3.2 Wireshark + Mecanismos de comunicação BD

Depois da anterior tarefa não ter levado ao resultados que esperavamos partimos para outra etapa, utilizar o wireshark de forma a capturar o tráfego entre o nodejs e a BD (MySQL). Nesta etapa também fizemos uma pesquisa intensiva sobre como utilizar o wireshark de forma a usarmos esta ferramenta para nosso próprio proveito.

Encontramos este link que nos deu uma luz de como usar o wireshark neste situação: Understanding MySQL Client/Server Protocol Using Python Wireshark[14]. A partir daqui começamos a analisar todo o tipo de tráfego entre o nodejs e a BD (MySQL).

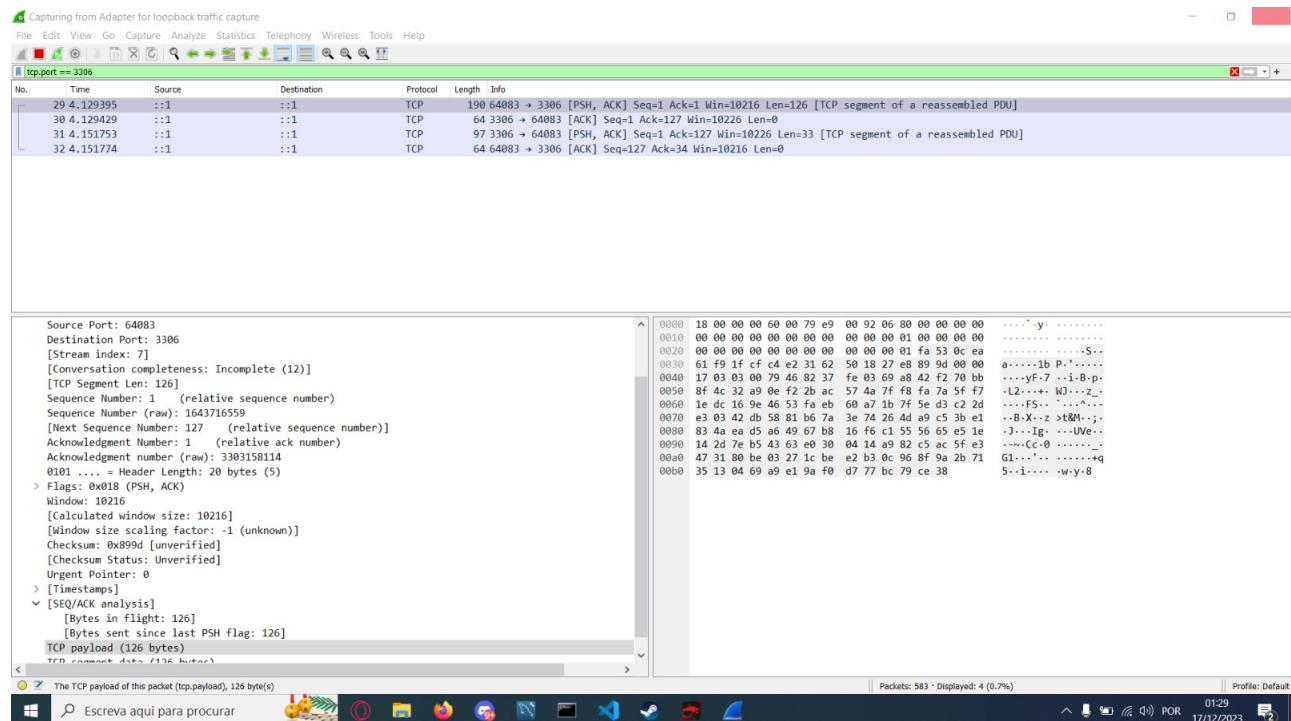


Figura 3.7: Ilustração da captura do wireshark

Além desta tarefa, temos também outro objetivo, entender como é que o sequileze interage com a BD, protocolo de comunicação, mensagens trocadas, perceber a diferença entre utilizar o jdbc para nodejs vs sequialize, compreender se faz sentido usar jdbc, que linguagens de programação ele suporta e afins e estudar a api dos connectores. Primeiro vamos começar pelas dois primeiras tarefas referidas, saber como o sequileze interage como a BD e as diferenças que este tem com o jdbc e se faz sentido usar o jdbc para a nossa solução, para descobrir isto fizemos uma pesquisa intensiva e encontramos algumas respostas. As respostas obtidas vieram nomeadamente do chatgpt que nos explicou tudo o que precisavamos de saber, para além de nos dar links importantíssimos que nos ajudaram no cumprimento destas tarefas.

Estas proximas informações que irei colocar aqui são frutos do chatgpt que nos ajudou no entendimento deste problema: A diferença entre JDBC (Java Database Connectivity) e Sequelize (um ORM para Node.js) é significativa, pois são ferramentas destinadas a diferentes linguagens de programação e possuem abordagens distintas para a interação com bancos de dados.

- 1. JDBC para Node.js

JDBC é uma API padrão em Java para conectar e executar consultas em bancos de dados. Ele oferece uma maneira direta e de baixo nível para se conectar a bancos de dados, executar consultas SQL e gerenciar transações.

Como funciona: JDBC envolve a criação de conexões com o banco de dados, a execução de consultas SQL (SELECT, INSERT, UPDATE, DELETE) e o tratamento de resultados em código Java. Para utilizar JDBC no Node.js, é necessário integrar bibliotecas Java com o Node.js, o que pode ser complicado e não é uma abordagem comum, já que o Node.js possui soluções mais nativas para trabalhar com bancos de dados.

Exemplo: Em um aplicativo Java, você usaria JDBC para conectar-se a um banco de dados MySQL, executar uma consulta SQL, e iterar pelos resultados. Isso envolve

bastante código para gerenciar a conexão e tratar erros.

- 2. Sequelize (ORM para Node.js)

Sequelize é um ORM (Object-Relational Mapping) para Node.js, que permite mapear tabelas de banco de dados para objetos JavaScript. Ele abstrai o SQL, tornando o acesso ao banco de dados mais intuitivo e reduzindo a quantidade de código necessário para manipular dados.

Como funciona: Com Sequelize, você define modelos para suas tabelas de banco de dados, e o Sequelize lida com as consultas SQL por você. Ele também suporta migrações de banco de dados, associações entre tabelas, e facilita a criação de consultas complexas.

Exemplo: Em um aplicativo Node.js, você define um modelo para uma tabela Users e usa métodos Sequelize como findAll() para obter todos os usuários, sem escrever consultas SQL diretamente.

- Principais Diferenças:

Linguagem: JDBC é para Java. Sequelize é para Node.js.

- Nível de Abstração:

JDBC oferece acesso de baixo nível ao banco de dados, exigindo que você escreva SQL e gerencie a conexão manualmente. Sequelize abstrai o SQL, permitindo que você trabalhe com objetos JavaScript e métodos de alto nível.

- Facilidade de Uso:

JDBC pode ser mais verboso e requer mais código para operações simples. Sequelize é mais amigável e simplifica a manipulação de dados com menos código.

- Migrações e Associações:

JDBC não fornece ferramentas integradas para migrações de banco de dados ou associações entre tabelas. Sequelize tem suporte embutido para migrações e associações, o que é uma vantagem para desenvolvedores que precisam manter a estrutura do banco de dados sincronizada com o código.

- Resumo:

Use JDBC se você estiver desenvolvendo em Java e precisa de controle detalhado sobre as consultas SQL. Use Sequelize se você estiver desenvolvendo em Node.js e quer uma interface de alto nível para o banco de dados, com suporte para migrações e associações.

Além desta explicação bastante completa por parte do chatgpt, este também forneceu diversos links super interessantes e cheios de informações que nos ajudaram a cimentar/- consolidar novos conhecimentos acerca deste assunto.

- JDBC Tutorial[15];
- Introduction to JDBC [16] ;
- Sequelize [17];
- Sequelize em 15 minutos [18];
- SQL no Node.js com Sequelize [19] .

Depois desta pesquisa intensiva e depois de todas as informações obtidas, concluimos que a opção que mais se adequava ao nossa solução seria o sequelize pois estamos e utilizar o node.js e o sequelize é mais adequado, isto é um ponto bastante crucial que fez com que a nossa escolha fosse rápida e acertada.

Depois de mais uma pesquisa intensiva para obtermos informações necessárias para o seguimento deste projeto, tivemos de fazer mais uma pesquisa, agora sobre os conectores Mysql. Estes conectores seriam importantes no futuro do nosso projeto e extremamente necessários. Voltamos a fazer mais uma pesquisa intensiva atrás de saber e entender os conectores Mysql e encontramos um link de documentação oficial do mysql que explica tudo o que precisavamos de saber acerca deste assunto: MySQL Connectors [20], O MySQL Connector é um conector de base de dados MySQL para aplicações que se ligam a servidores MySQL. O Connector pode ser utilizado para aceder a servidores MySQL que implementam um armazenamento de documentos ou de forma tradicional utilizando instruções SQL. Estes conectores seriam importantes para nos conseguirmos ligar ao nosso servidor Mysql local, neste caso a MariaDB.

Depois de entendermos os conectores Mysql fizemos mais uma pequena pesquisa atrás de ficarmos mais dentro do assunto sobre como funciona a arquitetura Mysql e encontramos mais um link que nos fez conseguir obter mais informações sobre este tema: Architecture of MySQL [21], além deste link encontramos mais um link que nos ajudou a perceber mais sobre o wireshark: SampleCaptures [22].

Neste momento, o ponto de situação estava virado para a procura de informações e material para que podessemos trabalhar da forma mais correta em busca do nosso objetivo final. Foi através de mais pesquisa que encontramos mais links que nos podiam ajudar a obter mais informações, dois dos links referem-se ao Mysql visto que são documentações oficiais do site do Mysql: MySQL Connector/J Developer Guide [23], MySQL 8.0 C API Developer Guide [24]. No mesmo momento em que pesquisavamos sobre Mysql também pesquisamos mais acerca do wireshark e encontramos mais dois links importantes para o nosso objetivo: MySQL queries coming on TCP instead of MySQL Protocol [25], Clear Capture of MySQL Query SQL Using Wireshark: A Guide [26].

Agora aqui surge um problema, depois de tanta pesquisa e procura de informação voltamos a mexer mais propriamente na parte de implementação do projeto, o nosso objetivo era ver se conseguíamos através do wireshark capturar as queries que nós executávamos,

o objetivo sendo mais concreto era capturar pacotes Mysql para haver se havia comunicações entre o cliente que criamos e a base de dados, quando capturamos os pacotes Mysql no wireshark reparamos que eles estavam estranhos, pareciam encriptados, então pensamos em várias soluções ate que nos veio a cabeça desativar o ssl na ligação do workbench ao MariaDB, a partir desse momento conseguimos ver as queries que era feitas dentro do pacote Mysql capturado no wireshark.

3.3 Criação de um exemplo em C

Depois de muita pesquisa e implementação, a nossa tarefa foi mudada para o seguinte objetivo: Exemplo em C de receção de dados com sockets bsd na porta 3306, ligar o exemplo do node a este exemplo com sockets bsd e fazer print das mensagens que estão a chegar. Este foi o objetivo que nos foi atribuído e a partir daqui começamos a desenvolver o código em C que nos foi pedido, para além disso encontramos mais alguns links como por exemplo este aqui: TCP Server-Client implementation in C [27], que nos ajuda a entender como podemos criar uma ligação tcp entre client e server.

A partir deste momento deixamos de lado a parte da pesquisa, apesar de que pontualmente íamos pesquisar algumas coisas caso o código que estavamos a desenvolver nos desse erro mas partir daqui, seguiu-se várias tentativas para conseguir ter um código que fizesse o que nós pretendíamos. A primeira versão do código foi esta.

```

compile:
    gcc -Wall -g3 -fsanitize=address -pthread server.c -o server -lmysqlclient
    gcc -Wall -g3 -fsanitize=address -pthread client.c -o client -lmysqlclient
FLAGS    = -L /lib64
LIBS     = -lusb-1.0 -l pthread

```

Figura 3.8: Ilustração do Makefile (ver apêndiceB)

```

client.c  x
client.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <signal.h>
5  #include <unistd.h>
6  #include <sys/types.h>
7  #include <sys/socket.h>
8  #include <netinet/in.h>
9  #include <arpa/inet.h>
10 #include <pthread.h>
11 #include <mysql/mysql.h>
12
13 #define LENGTH 2048
14 #define BUFFER_SIZE 4096
15
16 // Global variables
17 volatile sig_atomic_t flag = 0;
18 int sockfd = 0;
19 char name[32];
20 MYSQL *conn;
21
22 void str_overwrite_stdout() {
23     printf("%s", "> ");
24     fflush(stdout);
25 }
26
27 void str_trim_lf(char *arr, int length) {
28     int i;
29     for (i = 0; i < length; i++) { // trim \n
30         if (arr[i] == '\n') {
31             arr[i] = '\0';
32             break;
33         }
34     }
35 }
36
37 void catch_ctrl_c_and_exit(int sig) {
38     flag = 1;
39 }
40
41 void send_msg_handler() {
42     char message[LENGTH] = {};
43     char buffer[BUFFER_SIZE] = {};
44
45     while (1) {
46         str_overwrite_stdout();
47         fgets(message, LENGTH, stdin);
48         str_trim_lf(message, LENGTH);

```

Figura 3.9: Ilustração do client.c1 (ver apêndiceB)

```

50     if (strcmp(message, "exit") == 0) {
51         break;
52     } else {
53         // Utilize snprintf para evitar buffer overflow
54         snprintf(buffer, sizeof(buffer), "%s\n", name, message);
55         if (strlen(buffer) < sizeof(buffer)) {
56             send(sockfd, buffer, strlen(buffer), 0);
57         } else {
58             printf("Message too long, not sent.\n");
59         }
60     }
61
62     bzero(message, LENGTH);
63     bzero(buffer, BUFFER_SIZE);
64 }
65 catch_ctrl_c_and_exit(2);
66 }
67
68 void recv_msg_handler() {
69     char message[LENGTH] = {};
70     while (1) {
71         int receive = recv(sockfd, message, LENGTH, 0);
72         if (receive > 0) {
73             printf("%s", message);
74             str_overwrite_stdout();
75         } else if (receive == 0) {
76             break;
77         } else {
78             // -1
79         }
80         memset(message, 0, sizeof(message));
81     }
82 }
83
84
85
86
87
88 // conexao com a base de dados
89 int db_connect() {
90     conn = mysql_init(NULL);
91
92     if (conn == NULL) {
93         fprintf(stderr, "mysql_init() failed\n");

```

Figura 3.10: Ilustração do client.c2

```

58 void recv_msg_handler() {
59     char message[LENGTH] = {};
60     while (1) {
61         int receive = recv(sockfd, message, LENGTH, 0);
62         if (receive > 0) {
63             printf("%s", message);
64             str_overwrite_stdout();
65         } else if (receive == 0) {
66             break;
67         } else {
68             // -1
69         }
70         memset(message, 0, sizeof(message));
71     }
72 }
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88 // conexao com a base de dados
89 int db_connect() {
90     conn = mysql_init(NULL);
91
92     if (conn == NULL) {
93         fprintf(stderr, "mysql_init() failed\n");
94         return EXIT_FAILURE;
95     }
96
97     // Substitua os seguintes valores pelos do Workbench
98     const char *db_host = "127.0.0.1";
99     const char *db_user = "ruiaires";
100    const char *db_password = "ruiaires";
101    const char *db_name = "teste1";
102    unsigned int db_port = 3306;
103
104    if (mysql_real_connect(conn, db_host, db_user, db_password, db_name, db_port, NULL, 0) == NULL) {
105        fprintf(stderr, "mysql_real_connect() failed: %s\n", mysql_error(conn));
106        mysql_close(conn);
107        return EXIT_FAILURE;
108    }
109
110    printf("Connected to MySQL server\n");
111
112    return EXIT_SUCCESS;
113 }

```

Figura 3.11: Ilustração do client.c3

```

void db_disconnect() {
    if (conn != NULL) {
        mysql_close(conn);
        printf("Disconnected from MySQL server\n");
    }
}

int main(int argc, char **argv) {
    if (argc != 2) {
        printf("Usage: %s <port>\n", argv[0]);
        return EXIT_FAILURE;
    }

    char *ip = "127.0.0.1";
    int port = atoi(argv[1]);

    signal(SIGINT, catch_ctrl_c_and_exit);

    printf("Please enter your name: ");
    fgets(name, 32, stdin);
    str_trim_lf(name, strlen(name));

    if (strlen(name) > 32 || strlen(name) < 2) {
        printf("Name must be less than 30 and more than 2 characters.\n");
        return EXIT_FAILURE;
    }

    // Conectar a base de dados
    if (db_connect() != EXIT_SUCCESS) {
        return EXIT_FAILURE;
    }
    struct sockaddr_in server_addr;

    /* Socket settings */
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(ip);
    server_addr.sin_port = htons(port);

    // Connect to Server
    int err = connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr));
    if (err == -1) {
        printf("ERROR: connect\n");
    }
}

```

Figura 3.12: Ilustração do client.c4

```
    printf("ERROR: connect\n");
    return EXIT_FAILURE;
}

// Send name
send(sockfd, name, 32, 0);

printf("== WELCOME TO THE CHATROOM ==\n");

pthread_t send_msg_thread;
if (pthread_create(&send_msg_thread, NULL, (void *)send_msg_handler, NULL) != 0) {
    printf("ERROR: pthread\n");
    return EXIT_FAILURE;
}

pthread_t recv_msg_thread;
if (pthread_create(&recv_msg_thread, NULL, (void *)recv_msg_handler, NULL) != 0) {
    printf("ERROR: pthread\n");
    return EXIT_FAILURE;
}

while (1) {
    if (flag) {
        printf("\nBye\n");
        break;
    }
}

close(sockfd);

db_disconnect();

return EXIT_SUCCESS;
```

Figura 3.13: Ilustração do client.c5

```

C server.c
1  #include <sys/socket.h>
2  #include <netinet/in.h>
3  #include <arpa/inet.h>
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <unistd.h>
7  #include <errno.h>
8  #include <string.h>
9  #include <pthread.h>
10 #include <sys/types.h>
11 #include <signal.h>
12 #include <mysql/mysql.h>
13
14 #define MAX_CLIENTS 100
15 #define BUFFER_SZ 2048
16
17 static _Atomic unsigned int cli_count = 0;
18 static int uid = 10;
19
20 /* Client structure */
21 typedef struct
22 {
23     struct sockaddr_in address;
24     int sockfd;
25     int uid;
26     char name[32];
27 } client_t;
28
29 client_t *clients[MAX_CLIENTS];
30
31 pthread_mutex_t clients_mutex = PTHREAD_MUTEX_INITIALIZER;
32
33 void str_overwrite_stdout()
34 {
35     printf("\r\x1b[1A");
36     fflush(stdout);
37 }
38
39 void str_trim_lf(char *arr, int length)
40 {
41     int i;
42     for (i = 0; i < length; i++)
43     { // trim \n
44         if (arr[i] == '\n')
45         {
46             arr[i] = '\0';
47             break;
48         }

```

Figura 3.14: Ilustração do server.c (ver apêndiceB)

```

50 }
51
52 void print_client_addr(struct sockaddr_in addr)
53 {
54     printf("%d.%d.%d.%d",
55             addr.sin_addr.s_addr & 0xff,
56             (addr.sin_addr.s_addr & 0xffff00) >> 8,
57             (addr.sin_addr.s_addr & 0xff0000) >> 16,
58             (addr.sin_addr.s_addr & 0x000000) >> 24);
59 }
60
61 /* Add clients to queue */
62 void queue_add(client_t *cl)
63 {
64     pthread_mutex_lock(&clients_mutex);
65
66     for (int i = 0; i < MAX_CLIENTS; ++i)
67     {
68         if (!clients[i])
69         {
70             clients[i] = cl;
71             break;
72         }
73     }
74
75     pthread_mutex_unlock(&clients_mutex);
76 }
77
78 /* Remove clients to queue */
79 void queue_remove(int uid)
80 {
81     pthread_mutex_lock(&clients_mutex);
82
83     for (int i = 0; i < MAX_CLIENTS; ++i)
84     {
85         if (clients[i])
86         {
87             if (clients[i]->uid == uid)
88             {
89                 clients[i] = NULL;
90                 break;
91             }
92         }
93     }
94
95     pthread_mutex_unlock(&clients_mutex);
96 }

```

Figura 3.15: Ilustração do server.c2

```

97  /* Send message to all clients except sender */
98  void send_message(char *s, int uid)
99  {
100     pthread_mutex_lock(&clients_mutex);
101
102     for (int i = 0; i < MAX_CLIENTS; ++i)
103     {
104         if (clients[i])
105         {
106             if (clients[i]->uid != uid)
107             {
108                 if (write(clients[i]->sockfd, s, strlen(s)) < 0)
109                 {
110                     perror("ERROR: write to descriptor failed");
111                     break;
112                 }
113             }
114         }
115     }
116
117     pthread_mutex_unlock(&clients_mutex);
118 }
119
120
121
122
123 /* Inserir mensagem na base de dados */
124 void insert_message_into_database(const char *name, const char *message)
125 {
126     MYSQL *conn;
127     MYSQL_RES *res;
128     MYSQL_ROW row;
129
130     conn = mysql_init(NULL);
131
132     if (conn == NULL)
133     {
134         fprintf(stderr, "mysql_init() failed\n");
135         return;
136     }
137
138     if (mysql_real_connect(conn, "127.0.0.1", "ruiaires", "ruiaires", "teste1", 3306, NULL, 0) == NULL)
139     {
140         fprintf(stderr, "mysql_real_connect() failed\n");
141         mysql_close(conn);
142         return;
143     }

```

Figura 3.16: Ilustração do server.c3

```

145     char query[256];
146     sprintf(query, "INSERT INTO chat_messages (user, message) VALUES ('%s', '%s')", name, message);
147
148     if (mysql_query(conn, query))
149     {
150         fprintf(stderr, "INSERT failed: %s\n", mysql_error(conn));
151     }
152
153     mysql_close(conn);
154 }
155
156
157
158 /* Handle all communication with the client */
159 void *handle_client(void *arg)
160 {
161     char buff_out[BUFFER_SZ];
162     char name[32];
163     int leave_flag = 0;
164
165     cli_count++;
166     client_t *cli = (client_t *)arg;
167
168     // Name
169     if (recv(cli->sockfd, name, 32, 0) <= 0 || strlen(name) < 2 || strlen(name) >= 32 - 1)
170     {
171         printf("Didn't enter the name.\n");
172         leave_flag = 1;
173     }
174     else
175     {
176         strcpy(cli->name, name);
177         sprintf(buff_out, "%s has joined\n", cli->name);
178         printf("%s", buff_out);
179         send_message(buff_out, cli->uid);
180     }
181
182     bzero(buff_out, BUFFER_SZ);
183
184     while (1)
185     {
186         if (leave_flag)
187         {
188             break;
189         }
190         int receive = recv(cli->sockfd, buff_out, BUFFER_SZ, 0);

```

Figura 3.17: Ilustração do server.c4

```

191     int receive = recv(cli->sockfd, buff_out, BUFFER_SZ, 0);
192     if (receive > 0)
193     {
194         printf("Received message from %s: %s\n", cli->name, buff_out);
195         if (strlen(buff_out) > 0)
196         {
197             send_message(buff_out, cli->uid);
198
199             str_trim_lf(buff_out, strlen(buff_out));
200             printf("%s -> %s\n", buff_out, cli->name);
201
202             // Inserir a mensagem no banco de dados
203             insert_message_into_database(cli->name, buff_out);
204         }
205     }
206     else if (receive == 0 || strcmp(buff_out, "exit") == 0)
207     {
208         sprintf(buff_out, "%s has left\n", cli->name);
209         printf("%s", buff_out);
210         send_message(buff_out, cli->uid);
211         leave_flag = 1;
212     }
213     else
214     {
215         printf("ERROR: -1\n");
216         leave_flag = 1;
217     }
218
219     bzero(buff_out, BUFFER_SZ);
220 }
221
222 /* Delete client from queue and yield thread */
223 close(cli->sockfd);
224 queue_remove(cli->uid);
225 free(cli);
226 cli_count--;
227 pthread_detach(pthread_self());
228
229 return NULL;
230 }
231
232 int main(int argc, char **argv)
233 {
234     if (argc != 2)
235     {

```

Figura 3.18: Ilustração do server.c5

```

134     if (argc != 2)
135     {
136         printf("Usage: %s <port>\n", argv[0]);
137         return EXIT_FAILURE;
138     }
139
140     char *ip = "127.0.0.1";
141     int port = atoi(argv[1]);
142     int option = 1;
143     int listenfd = 0, connfd = 0;
144     struct sockaddr_in serv_addr;
145     struct sockaddr_in cli_addr;
146     pthread_t tid;
147
148     /* Socket settings */
149     listenfd = socket(AF_INET, SOCK_STREAM, 0);
150     serv_addr.sin_family = AF_INET;
151     serv_addr.sin_addr.s_addr = inet_addr(ip);
152     serv_addr.sin_port = htons(3306);
153
154     /* Ignore pipe signals */
155     signal(SIGPIPE, SIG_IGN);
156
157     if (setsockopt(listenfd, SOL_SOCKET, (SO_REUSEPORT | SO_REUSEADDR), (char *) &option, sizeof(option)) < 0)
158     {
159         perror("ERROR: setsockopt failed");
160         return EXIT_FAILURE;
161     }
162
163     /* Bind */
164     if (bind(listenfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
165     {
166         perror("ERROR: Socket binding failed");
167         return EXIT_FAILURE;
168     }
169
170     /* Listen */
171     if (listen(listenfd, 10) < 0)
172     {
173         perror("ERROR: Socket listening failed");
174         return EXIT_FAILURE;
175     }
176
177     printf("== WELCOME TO THE CHAT;] HAVE FUN! ==\n");
178
179     while (1)
180     {

```

Figura 3.19: Ilustração do server.c6

```

    perror("Error creating socket connection\n");
    return EXIT_FAILURE;
}

printf("== WELCOME TO THE CHAT;] HAVE FUN! ==\n");

while (1)
{
    socklen_t clilen = sizeof(cli_addr);
    connfd = accept(listenfd, (struct sockaddr *)&cli_addr, &clilen);
    printf("ola rui");
    /* Check if max clients is reached */
    if ((cli_count + 1) == MAX_CLIENTS)
    {
        printf("Max clients reached. Rejected: ");
        print_client_addr(cli_addr);
        printf(":d\n", cli_addr.sin_port);
        close(connfd);
        continue;
    }

    /* Client settings */
    client_t *cli = (client_t *)malloc(sizeof(client_t));
    cli->address = cli_addr;
    cli->sockfd = connfd;
    cli->uid = uid++;

    /* Add client to the queue and fork thread */
    queue_add(cli);
    pthread_create(&tid, NULL, &handle_client, (void *)cli);

    /* Reduce CPU usage */
    sleep(1);
}

return EXIT_SUCCESS;
}

```

Figura 3.20: Ilustração do server.c7

Esta primeira versão do código tem um código Makefile que faz a compilação do server.c e client.c, depois essa versão contém como já referido o server.c e o client.c. O server.c é basicamente a implementação de um servidor de chat multicliente com MySQL para armazenar as mensagens. Ele utiliza sockets para comunicação entre clientes e threads para tratar múltiplas conexões simultaneamente. O servidor aceita conexões de clientes, envia e recebe mensagens de todos eles, e armazena essas mensagens em um banco de dados MySQL.

O client.c é basicamente a implementação de um cliente de chat em C, que se conecta a um servidor via sockets e a um banco de dados MySQL. Ele também usa multithreading para gerenciar o envio e recebimento de mensagens de forma independente, além de permitir a conexão ao MySQL para futuras interações com o banco de dados.

Depois desta primeira versão do código, o objetivo foi melhorá-lo e corrigi-lo de forma a que se colocasse o wireshark à escuta na porta 3306 para o ip: 127.0.0.1 e tentar perceber o que o workbench envia para tentarmos ajeitar o receiver. O erro que começou a aparecer foi o bloqueio do receiver, pois este só desbloqueia depois de receber alguma resposta. A partir daqui era necessário compreender este erro pois se este erro permanecesse seria impossível continuar. Depois de mais pesquisa e a utilização em casos extremos do chatgpt fizemos outra versão do código (ver apêndiceB), ligamos ao workbench e utilizamos o wireshark para escutar a porta 3306, o objetivo a este ponto era tentar perceber o que o workbench enviava para assim podermos corrigir o receiver.

Depois de muito tempo perdido, muita tentativa de melhorar código e tentar ajeitar o receiver e depois de muita utilização do wireshark para tentar encontrar algo em concreto enviado pelo workbench, foi ai que demos um passo em frente, descobrimos o greeting message. O greeting message é estrutura sql, como se fosse uma mensagem de boas-vindas quando a conexão entre o client e o server funciona corretamente, encontramos esta estrutura num link de documentação oficial do Mysql depois de uma pesquisa intensiva sobre este assunto:smysqlgreetings Struct Reference [28], a partir desta descoberta isto foi o nosso foco, conseguir captar este greeting message pois isto significaria que a conexão foi estabelecida com sucesso e a partir daqui poderíamos prosseguir com o nosso objetivo para este projeto.

Depois disto criamos uma estrutura que estava quase completa que nos permitira capturar o greeting message e continuar no nosso objetivo, agora o ponto de situação seria completar este código (ver apêndiceB).

```

// Initialize the MySQL connection object
conn = mysql_init(NULL);
if (conn == NULL) {
    fprintf(stderr, "mysql_init() failed\n");
    close(socket_fd);
    return 1;
}

// Attach the socket descriptor to the MySQL connection object
if (mysql_real_connect(conn, NULL, "username", "password", "database_name", 0, socket_fd, 0) == NULL) {
    fprintf(stderr, "mysql_real_connect() failed: %s\n", mysql_error(conn));
    close(socket_fd);
    mysql_close(conn);
    return 1;
}

// Create a socket
if ((socket_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("socket");
    return 1;
}

// Set up the server address structure
struct sockaddr_in server_addr;
memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); // Replace with your server's IP address
server_addr.sin_port = htons(3306); // Replace with your server's port

// accept questions

//greeting message mysql

```

Figura 3.21: Ilustração da estrutura para capturar o greeting message1

```

// fazer depois de tudo que estiver para tras ficar estavel.
while(1)
{
    //recv client workebench

    //submeter para o unix socket da BD

    //fazer send para o cliente workebench
}

// Close the connection
mysql_close(conn);
close(socket_fd);

return 0;

```

Figura 3.22: Ilustração da estrutura para capturar o greeting message2

Depois de completarmos a estrutura, ela sofreu algumas alterações e conseguimos captar a greeting message através do wireshark que estava a escuta na porta 3306, o código utilizado foi este (ver apêndiceB).

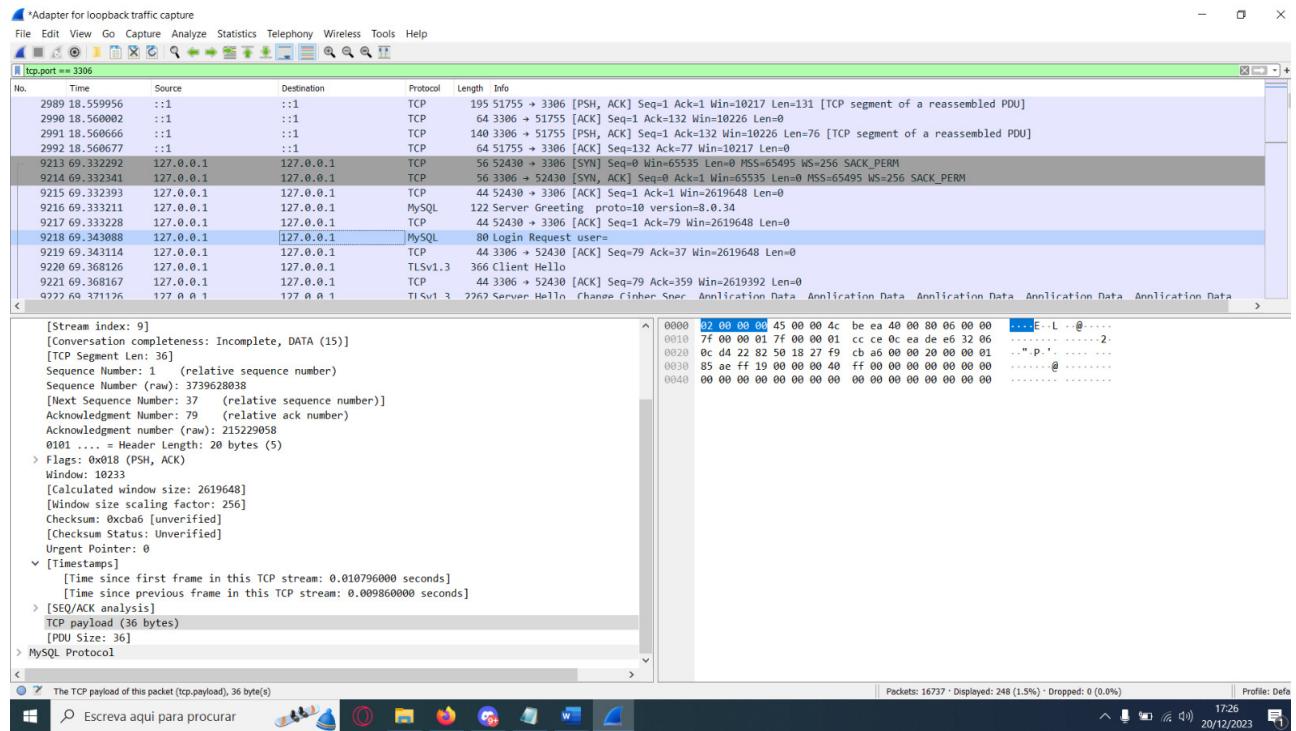


Figura 3.23: Ilustração da captura do greeting message

Depois de todo este progresso, veio o nosso maior problema, depois de receber a primeira mensagem de boas-vindas que seria como referido anteriormente o greeting message era suposto enviarmos uma resposta de volta e recebermos também uma resposta de volta por parte do servidor Mysql o que, não aconteceu mesmo depois de várias tentativas no sentido de conseguir receber essa mensagem de volta. Depois de muita insistência, pesquisa e alterações no código tivemos que infelizmente abandonar esta estratégia e partir para outra abordagem mais tranquila e segura, devido ao escasso tempo que tínhamos para continuar a desenvolver este projeto.

3.4 Proxysql

Foi então que migramos para o Proxysql mas antes de migrarmos para este proxy, tínhamos 3 opções: Proxysql, MaxScale e o MysqlProxy.

Antes de escolhermos qual quer um fizemos uma pesquisa para comparar os 3 e ver qual deles se adaptava ao que nós queríamos, então fizemos uma lista num bloco de notas onde falamos um bocadinho de cada um deles com informações acerca de cada um destes proxys:

- ProxySQL

ProxySQL é um proxy de alta performance para MySQL e MariaDB, projetado para lidar com conexões e consultas. ProxySQL permite extensões através de scripts Lua e uma API REST para administração e configuração.

ProxySQL é ideal para quem precisa de alta performance e flexibilidade, com suporte avançado para scripts Lua e uma API REST. ProxySQL permite uma ampla gama de personalizações como reescrita de consultas, roteamento e manipulação de eventos.

- MaxScale

MaxScale é um proxy modular e altamente configurável para MariaDB e MySQL, desenvolvido pela MariaDB Corporation. Permite a criação de módulos personalizados em C/C++

MaxScale é recomendado para ambientes que requerem funcionalidades modulares avançadas e segurança robusta, com suporte para extensões em C/C++. Embora seja mais focado em módulos C/C++, suporta scripts Lua para filtros de consultas, permitindo manipulações e personalizações rápidas.

- MySQL Proxy

MySQL Proxy é um projeto mais antigo e simples, desenvolvido pela MySQL (agora Oracle). É um proxy genérico que pode ser usado para monitoramento, análise, ou filtragem de tráfego entre clientes MySQL e servidores.

MySQL Proxy é uma solução mais simples e flexível para monitoramento e filtragem de tráfego SQL, adequada para quem precisa de uma ferramenta leve e extensível via Lua. Ideal para monitoramento, modificação de consultas e resultados, e outras personalizações em tempo real.

Depois de apresentarmos esta comparação entre estes 3 proxys, nós em conjunto com o professor Rui Alves decidimos utilizar o proxysql. Primeira coisa que fizemos vou instalar o ProxySql na máquina virtual através deste link: ProxySQL Installation [29]. Depois desta instalação foi nos dado 3 novas tarefas para fazer, vamos aborda-las uma a uma:

- Primeira tarefa

Ligar várias bases de dados ao proxy.

Ter várias instâncias ligadas ao proxy.

Quando digo base de dados não é para criar uma base de dados em SQL e mesmo uma instância do servidor

Para resolvemos a primeira tarefa dada fizemos uma pesquisa para descobrir do que que isto se tratava e acabamos por descobrir 4 links que seriam super importantes para conseguirmos realizar a primeira tarefa de forma rápida e fácil:

- Setting Up ProxySQL for Load Balancing MySQL Replication [30];
- Configuring Backend MySQL Servers in ProxySQL [31];
- Initial Configuration [32];

- nethalo/proxysql-basics-master-slave [33];;

Depois de analisarmos estes 4 links, acabamos por seguir o primeiro link, seguimos passo a passo o que é feito em termos de comandos e conseguimos no final ligar várias bases de dados ao proxy e ter também, várias instâncias ligadas ao proxy.

```
ProxySQLAdmin> SELECT hostgroup, srv_host, status, Queries FROM stats.stats_mysql_connection_pool;
+-----+-----+-----+
| hostgroup | srv_host | status | Queries |
+-----+-----+-----+
| 2 | 172.20.0.4 | ONLINE | 0 |
| 2 | 172.20.0.5 | ONLINE | 0 |
| 2 | 172.20.0.3 | ONLINE | 0 |
+-----+-----+-----+
3 rows in set (0,002 sec)
```

Figura 3.24: Ilustração da criação das instâncias dentro do proxy

Depois da ter concluido com sucesso a primeira tarefa, passamos agora para a segunda tarefa dada pelo professor Rui Alves:

- Segunda tarefa

Utilizar um cliente sql para ligar-se ao proxy. Pode ser um Workbench uma linha de comandos ou outro que queira.

Perceber se é possível interceptar as querys antes de elas serem executadas na BD.

Agora o objetivo é criar um client sql que se consiga ligar ao proxy. Antes de passarmos para esta parte precisamos de instalar o docker pois estas instâncias, a primary e as replicas vão estar inseridas em containers dentro do docker, caso contrário seria impossivel depois conseguir inserir dados nelas mesmas. Então fomos a procura de algo que nos ensinasse como criar esses containers que vão conter essas instâncias, depois de muita pesquisa encontramos um vídeo no youtube [34];, que nos mostra um codigo no visual studio code em que o objetivo é definir um ambiente de containers com o uso de Docker Compose para gerenciar um cluster de banco de dados MySQL com alta disponibilidade, incluindo

um ProxySQL e três instâncias do MySQL: um nó primário e duas réplicas, este ambiente vai estar ligado as instâncias criadas no proxysql através dos ips.

O código esta presente na apêndice, o código presente no ficheiro .yaml (ver apêndiceB), .txt (ver apêndiceB) e no ficheiro .sql (ver apêndiceB).

```
! docker-compose.yaml
1  docker-compose.yaml:
2
3  version: '3'
4  services:
5    proxy:
6      image: proxysql/proxysql:latest
7      restart: always
8      ports:
9        - "6034:6032"
10       - "6035:6033"
11      volumes:
12        - /data/proxysql:/var/lib/proxysql
13      networks:
14        proxysql_mysqlnet:
15          | ipv4_address: 172.20.0.2
16
17  mysql_primary:
18    image: mysql:5.7
19    restart: always
20    environment:
21      MYSQL_ROOT_PASSWORD: rootpassword
22      MYSQL_DATABASE: mydatabase
23      MYSQL_USER: myuser
24      MYSQL_PASSWORD: mypassword
25    ports:
26      - "3307:3306"
27    volumes:
28      - /data/mysql_primary:/var/lib/mysql
29    networks:
30      proxysql_mysqlnet:
31        | ipv4_address: 172.20.0.3
32
33  mysql_replica1:
34    image: mysql:5.7
35    restart: always
36    environment:
37      MYSQL_ROOT_PASSWORD: rootpassword
38      MYSQL_DATABASE: mydatabase
39      MYSQL_USER: myuser
40      MYSQL_PASSWORD: mypassword
41    ports:
42      - "3308:3306"
43    volumes:
44      - /data/mysql_replica1:/var/lib/mysql
45    networks:
46      proxysql_mysqlnet:
47        | ipv4_address: 172.20.0.4
48
```

Figura 3.25: Ilustração do código da criação dos containers1

```
mysql_replica2:
  image: mysql:5.7
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: rootpassword
    MYSQL_DATABASE: mydatabase
    MYSQL_USER: myuser
    MYSQL_PASSWORD: mypassword
  ports:
    - "3309:3306"
  volumes:
    - /data/mysql_replica2:/var/lib/mysql
  networks:
    proxyysql_mysqlnet:
      ipv4_address: 172.20.0.5

  networks:
    proxyysql_mysqlnet:
      driver: bridge
      ipam:
        config:
          - subnet: "172.19.20.0/16"
```

Figura 3.26: Ilustração do código da criação dos containers2

```
-- My mock data

username=root
password=4pYTzpy$TDjNsQKe

monitor_username=monitor
monitor_password=TsQRDzNfh5Wvhm8

Master Database:
  url=172.19.0.3 # IP do MySQL Primário

Replica1:
  url=172.19.0.4 # IP da Replica 1

Replica2:
  url=172.19.0.5 # IP da Replica 2
```

Figura 3.27: Ilustração do código da criação dos containers3

```
-- Suggested Permission--
GRANT SELECT, REPLICATION CLIENT, SHOW DATABASES, SUPER, PROCESS
ON *
TO 'mysqluser'@'localhost'
IDENTIFIED BY 'agent_password';

-- permissions you will actually need--
GRANT SELECT, REPLICATION CLIENT, SHOW DATABASES, PROCESS
ON *
TO 'monitor'@'*'
IDENTIFIED BY 'TsQRDzNfh5Wvhm8';
```

Figura 3.28: Ilustração do código da criação dos containers4

Depois de estar tudo pronto fazemos no terminal do visual studio code o seguinte comando: docker-compose up -d, como podes ver na próxima imagem.

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'PROXYSQL' containing 'crete-monitor.sql', 'Databases.txt', and 'docker-compose.yaml'. The 'docker-compose.yaml' file is open in the main editor area, displaying the following YAML configuration:

```
version: '3'
services:
  proxy:
    image: proxysql/proxysql:latest
    restart: always
    ports:
      - "6034:6032"
      - "6035:6033"
    volumes:
      - /data/proxysql:/var/lib/proxysql
    networks:
      proxysql_mysqlnet:
        ipv4_address: 172.20.0.2
  mysql_primary:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: mydatabase
      MYSQL_USER: myuser
      MYSQL_PASSWORD: mypassword
    ports:
      - "3307:3306"
```

Below the editor, the terminal tab is active, showing the command being run: `docker-compose up -d`. The output from the terminal shows:

```
rui@debian:~/Documentos/ProxySQL$ docker-compose up -d
WARN[0000] /home/rui/Documentos/ProxySQL/docker-compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[*] Running 4/0
  ✓ Container proxysql-mysql_primary-1  Running
  ✓ Container proxysql-mysql_replica1-1  Running
  ✓ Container proxysql-mysql_replica2-1  Running
  ✓ Container proxysql-proxy-1          Running
rui@debian:~/Documentos/ProxySQL$
```

At the bottom of the terminal window, the status bar indicates: Ln 15, Col 17 Spaces: 2 UTF-8 LF Compose.

Figura 3.29: Ilustração do arranque dos containers

Com tudo direito e a funcionar, agora sim utilizamos um cliente sql para se conectar ao proxysql, o cliente que utilizamos neste caso foi o mysql workbench para nos podermos conectar ao proxysql. Fizemos a ligação e funcionou, a ligação foi conseguida com sucesso sem quaisquer problemas.

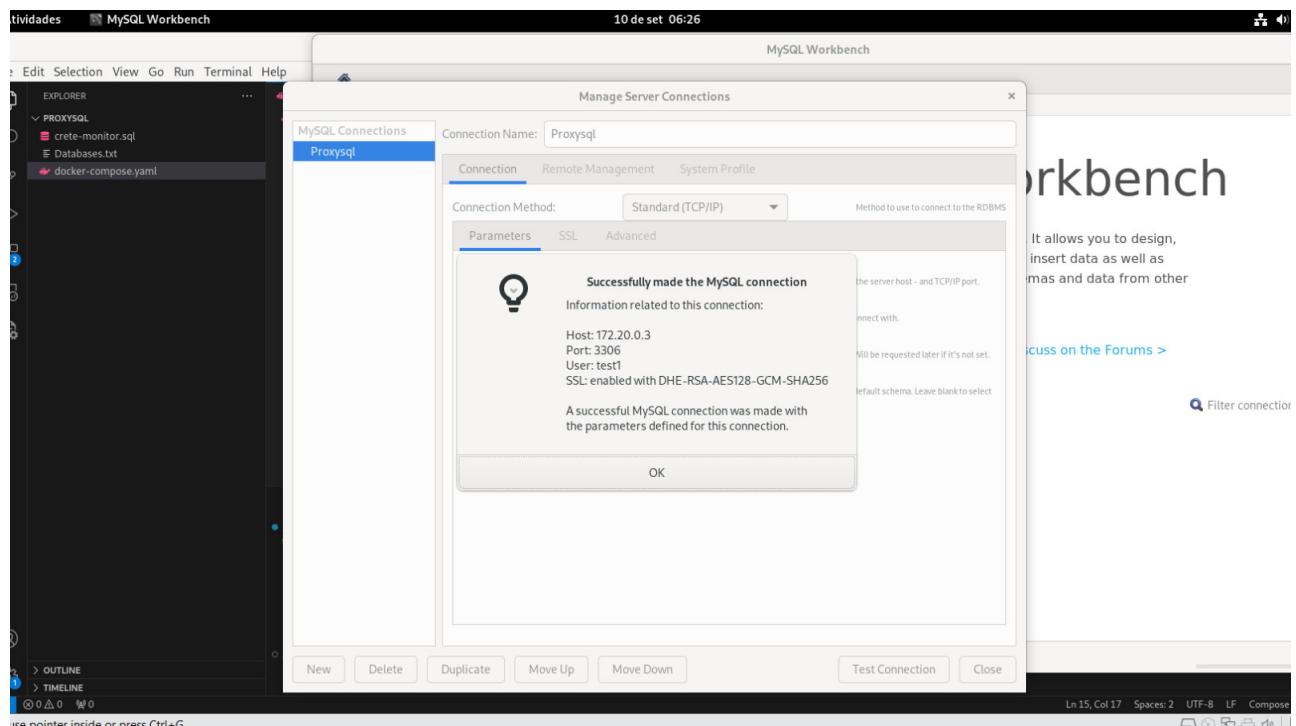


Figura 3.30: Ilustração da conexão do mysqlworkbench ao proxysql

Agora se quisermos adicionar algum dado as instâncias(tanto na primary como nas réplicas) podemos utilizar o workbench para o fazer como se conseguir perceber na próxima imagem.

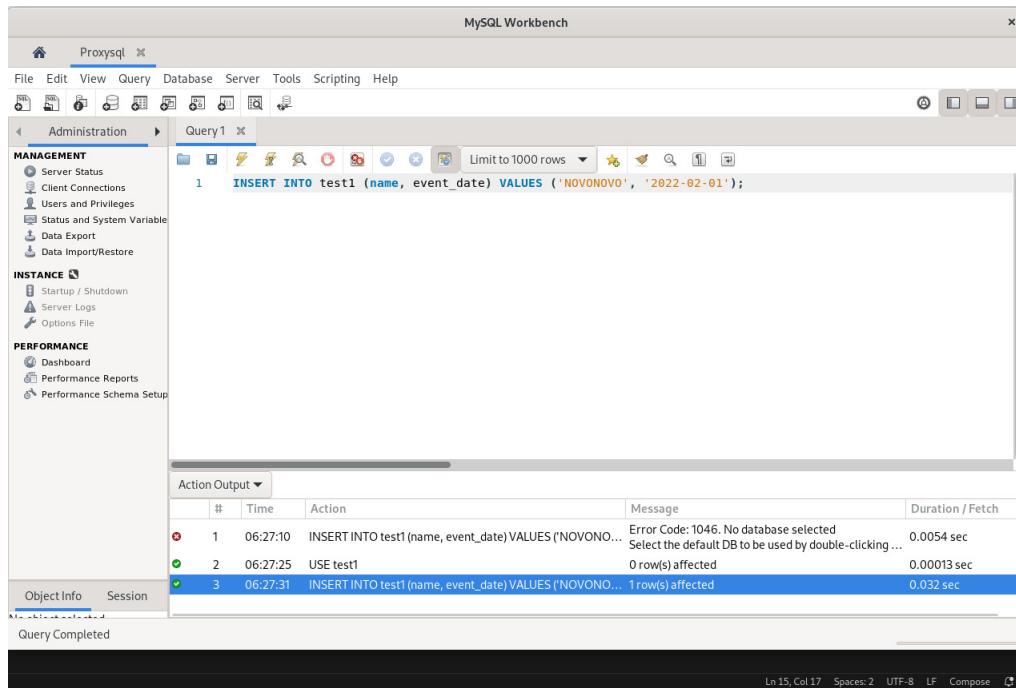
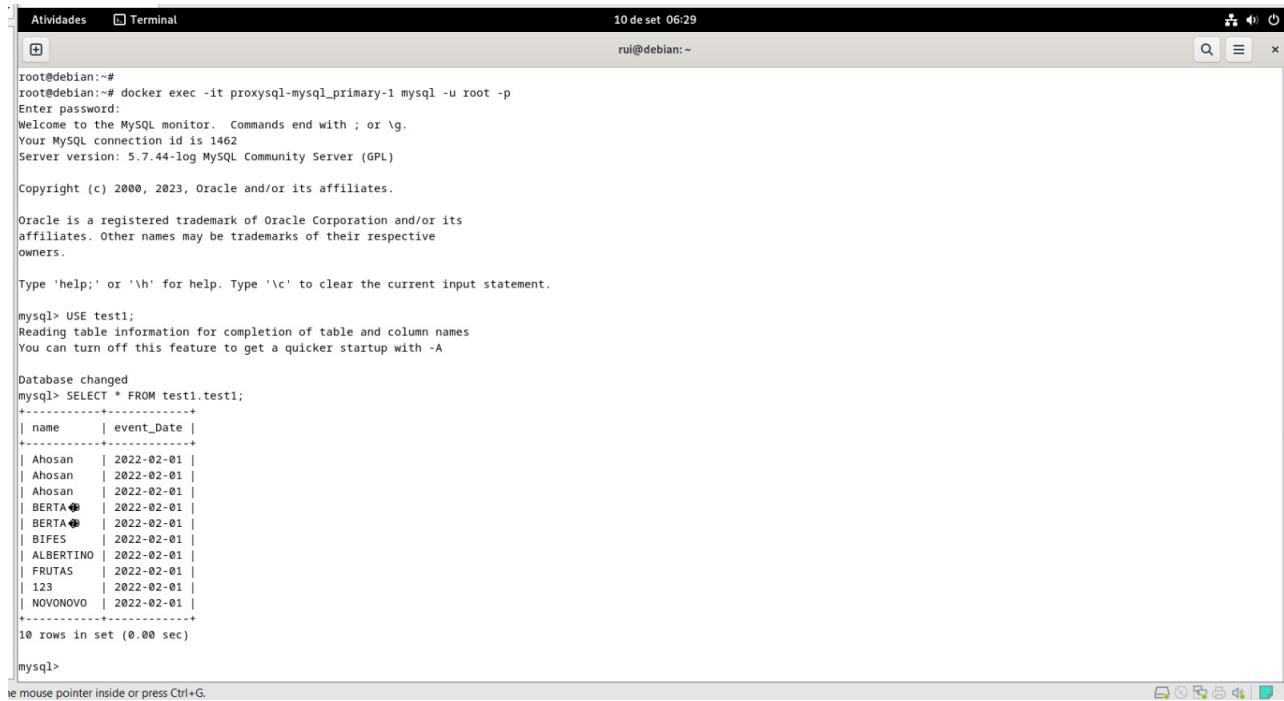


Figura 3.31: Ilustração de um insert into através do workbench

Para poder comprovar que o insert into feito no workbench foi ou não guardado tanto na instância principal como nas réplicas, conectamo-nos a cada uma delas separadamente e fizemos um SELECT para verificar se os dados foram inseridos.



```

root@debian:~#
root@debian:~# docker exec -it proxysql-mysql_primary-1 mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1462
Server version: 5.7.44-log MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE test1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

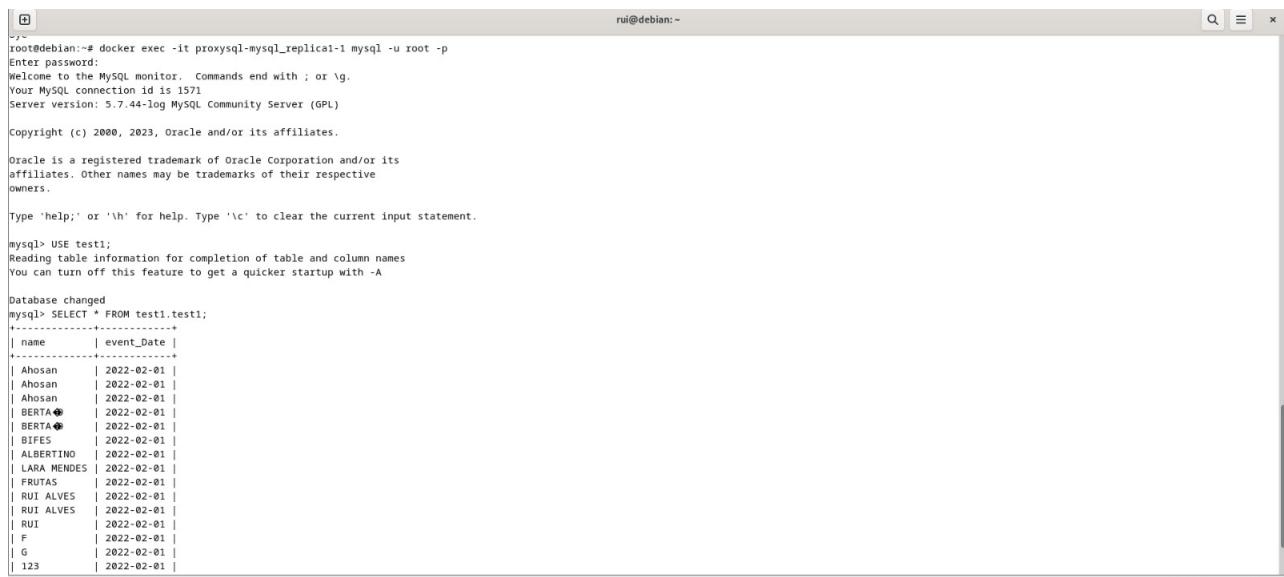
Database changed
mysql> SELECT * FROM test1.test1;
+-----+-----+
| name | event_Date |
+-----+-----+
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| BERTA❶ | 2022-02-01 |
| BERTA❶ | 2022-02-01 |
| BIFES | 2022-02-01 |
| ALBERTINO | 2022-02-01 |
| FRUTAS | 2022-02-01 |
| 123 | 2022-02-01 |
| NOVONOVO | 2022-02-01 |
+-----+
10 rows in set (0.00 sec)

mysql>

```

mouse pointer inside or press Ctrl+G.

Figura 3.32: Ilustração dos dados existentes na primary



```

root@debian:~#
root@debian:~# docker exec -it proxysql-mysql_replica1-1 mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1571
Server version: 5.7.44-log MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE test1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM test1.test1;
+-----+-----+
| name | event_Date |
+-----+-----+
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| BERTA❶ | 2022-02-01 |
| BERTA❶ | 2022-02-01 |
| BIFES | 2022-02-01 |
| ALBERTINO | 2022-02-01 |
| LARA MENDES | 2022-02-01 |
| FRUTAS | 2022-02-01 |
| RUI ALVES | 2022-02-01 |
| RUI ALVES | 2022-02-01 |
| RUI ALVES | 2022-02-01 |
| F | 2022-02-01 |
| G | 2022-02-01 |
| 123 | 2022-02-01 |
+-----+

```

Figura 3.33: Ilustração dos dados existentes na réplica1

name	event_Date
Ahosan	2022-02-01
Ahosan	2022-02-01
Ahosan	2022-02-01
BERTA	2022-02-01
BERTA	2022-02-01
BITES	2022-02-01
ALBERTINO	2022-02-01
LARA MENDES	2022-02-01
FRUTAS	2022-02-01
RUI ALVES	2022-02-01
RUI ALVES	2022-02-01
RUI	2022-02-01
F	2022-02-01
G	2022-02-01
123	2022-02-01
tt	2022-02-01
NOVONOVO	2022-02-01

Figura 3.34: Ilustração dos dados existentes na réplica1.2

```

root@debian:~# docker exec -it proxysql-mysql_replica2_1 mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1718
Server version: 5.7.44-log MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> test1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM test1;
+-----+-----+
| name | event_Date |
+-----+-----+
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| BERTA | 2022-02-01 |
| BERTA | 2022-02-01 |
| BITES | 2022-02-01 |
| ALBERTINO | 2022-02-01 |
| LARA MENDES | 2022-02-01 |
| FRUTAS | 2022-02-01 |
| RUI ALVES | 2022-02-01 |
| RUI ALVES | 2022-02-01 |
| RUI | 2022-02-01 |
| F | 2022-02-01 |
| G | 2022-02-01 |
| 123 | 2022-02-01 |
| tt | 2022-02-01 |
| NOVONOVO | 2022-02-01 |
+-----+-----+
10 rows in set (0.03 sec)

mysql>

```

Figura 3.35: Ilustração dos dados existentes na réplica2

A partir deste ponto chegamos a um beco sem saída, tentamos resolver o segundo objetivo pedido da segunda tarefa mas chegamos a um caminho que não nos levou a lado nenhum, tentamos fazer um script lua para tentar interceptar as querys antes de elas serem executadas na BD mas sem sucesso, fica na mesma uma nota para um link que encontramos durante a pesquisa que fizemos sobre este tema que poderá ser importante para se algum dia este tema volte a ser alvo de pesquisas e de implementações pois sentimos que o caminho a seguir para continuar a desenvolver mais este projeto era por aqui mesmo mas as nossas tentativas não nos levaram a lado nenhum, ficando assim sem qualquer tipo de efeito.

- ProxySQL Read/Write Split [35].

Capítulo 4

Resultados

Ao longe deste projeto, como foi referido no capítulo da implementação, foram feitos vários testes e pesquisas no sentido de encontrar e desenvolver uma solução para o nosso projeto, estes testes e pesquisas foram divididos por partes e camadas de forma a que podessemos obter o que procuravamos. Depois de tanta pesquisa e testes sentimos que nunca chegamos a algo concreto como gostaríamos, apesar disso temos alguns resultados em subetapas dos objetivos principais que são relevantes o suficiente para serem aqui destacadas e enunciadas.

4.1 Greeting message

A descoberta do greeting message foi um resultado muito positivo e relevante para o nosso projeto, depois da utilização deste código (ver apêndice B), começamos a captar no wireshark os pacotes Mysql com o greeting message o que significa que a ligação ao servidor Mysql foi estabelecida com sucesso, por outras palavras, o primeiro contacto foi estabelecido.

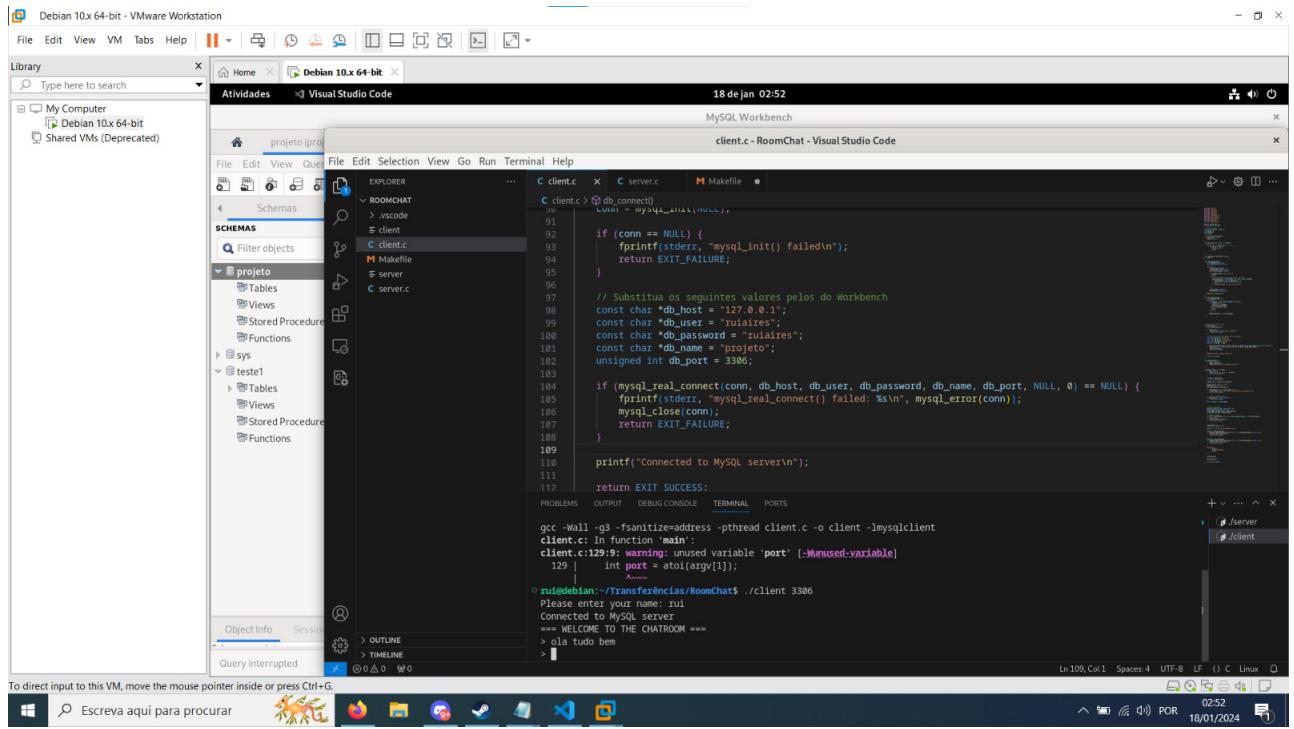


Figura 4.1: Ilustração do funcionamento do código para obter o greeting message

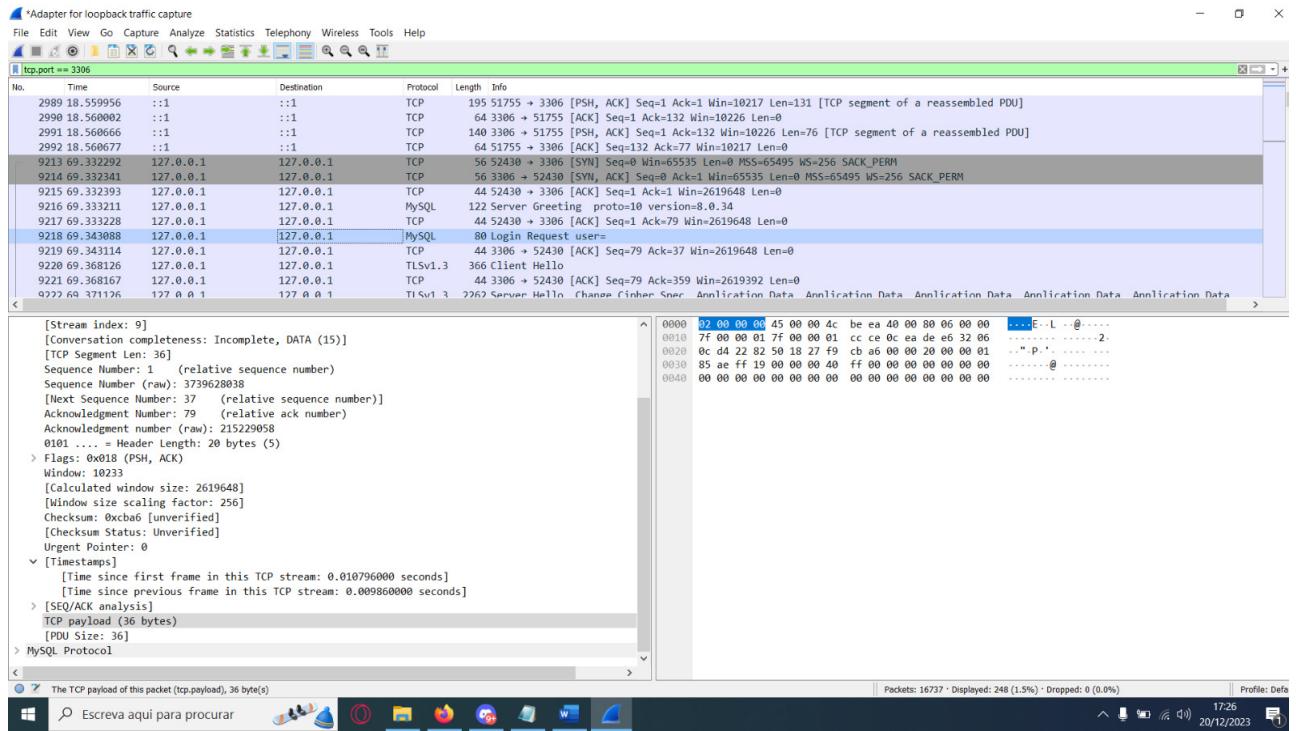


Figura 4.2: Ilustração da captura do pacote greeting message

Depois da descoberta do greeting message fizemos uma série de testes e pesquisas e alterações ja mencionadas no capítulo da implementação para tentarmos obter a segunda resposta esperada do lado do servidor Mysql, o que não aconteceu e nos levou a mudar totalmente de abordagem devido, a ser um beco sem saída mesmo depois de vários testes e também devido a escassas de tempo para a realização do projeto, o que nos fez abandonar por completo esta abordagem.

4.2 ProxySql

Depois da mudança de abordagem como já referida na implementação, utilizamos o proxysql para tentar obter resultados no sentido de conseguirmos concluir o nosso objetivo final. Como também já foi referido na própria implementação que obtivemos resultados interessantes neste ponto. Conseguimos obter o funcionamento total das instâncias(tanto na primary como nas réplicas), conseguindo assim inserir dados nelas e estes ficarem

guardados. Além disso conseguimos ligar um client ao proxysql, este sendo o Mysql workbench, tendo o total funcionamento do mesmo, permitindo fazer inserts, deletes e select que afetam diretamente as instâncias anteriormente referidas.

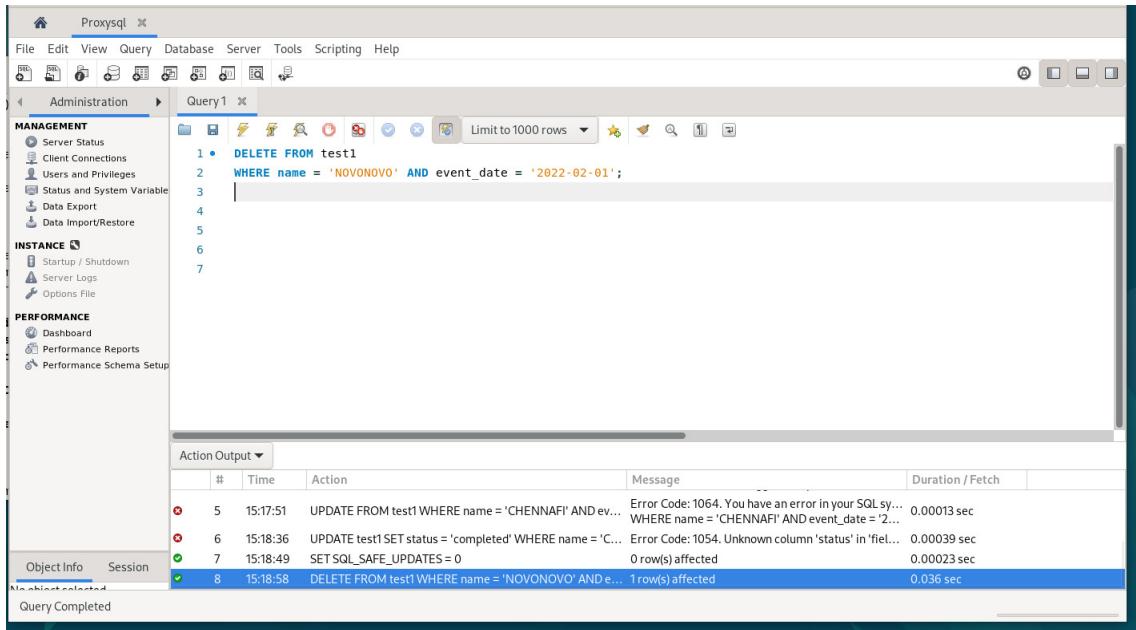


Figura 4.3: Ilustração do funcionamento das instâncias1

```

mysql> USE test1;
Database changed
mysql> SELECT * FROM test1;
+-----+-----+
| name | event_date |
+-----+-----+
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| Ahosan | 2022-02-01 |
| BERTA@ | 2022-02-01 |
| BERTA@ | 2022-02-01 |
| BIFES | 2022-02-01 |
| ALBERTINO | 2022-02-01 |
| FRUTAS | 2022-02-01 |
| 123 | 2022-02-01 |
+-----+
9 rows in set (0.00 sec)

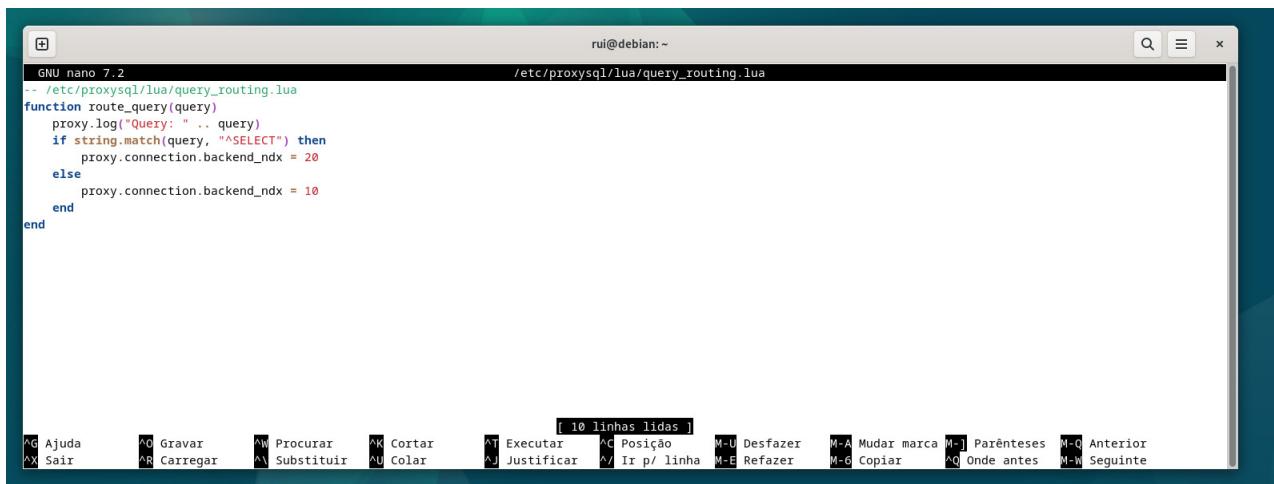
```

Figura 4.4: Ilustração do funcionamento das instâncias2

4.3 Interseção de Queries

A nossa última tentativa foi tentar fazer a interseção de queries antes de elas serem executadas na BD. Apesar de não termos conseguido obter nenhum resultado deixamos aqui uma nota pois sentimos que ficamos demasiado perto de uma resposta. Deixamos aqui uma nota para o script lua que tentamos utilizar para conseguir finalizar este objetivo (ver apêndice B), deixamos também aqui uma nota para um link que encontramos(já referido na implementação) que poderá ser o caminho certo a seguir.

- ProxySQL Read/Write Split [35].



```
GNU nano 7.2
rui@debian: ~
/etc/proxysql/lua/query_routing.lua
/etc/proxysql/lua/query_routing.lua
function route_query(query)
    proxy.log("Query: " .. query)
    if string.match(query, "^SELECT") then
        proxy.connection.backend_idx = 20
    else
        proxy.connection.backend_idx = 10
    end
end
```

Figura 4.5: Ilustração do script lua

Capítulo 5

Conclusão

A encriptação assimétrica é um tipo de encriptação que requer muita pesquisa e conhecimento sobre este mesmo tema para se poder desenvolver algo utilizando esta encriptação. Apesar de no fim desta pesquisa e implementação, não termos chegado a uma conclusão definitiva acho que há pontos a se ressalvar sobre este projeto.

Primeiro achamos que o desenvolvimento deste projeto nos levou a aprender e cimentar bastantes conhecimentos que a partida, nunca soubemos da sua existência e nem o quão importantes estes conhecimentos são para a área de cibersegurança, este projeto fez-nos aprender mais sobre a utilização da chave pública e primária e o quão importante é os utilizadores terem a sua privacidade.

Este projeto também nos ajudou a descobrir ferramentas novas que nunca tínhamos utilizado na nossa vida académica como estudantes de engenharia informática, como por exemplo o Proxysql e além de obtermos mais conhecimento do que aquele que já tínhamos sobre as outras ferramentas também usadas como o Mysql workbench, Docker, MariaDB, Visual Studio Code, Wireshark e a VmWare.

Sentimos que este tema poderá ser mais aprofundado no futuro, tanto corrigir os problemas que nos tivemos como conseguir algo a mais para além do nosso objetivo final. Posto isto, este tema tem muita coisa a se explorar pois é um tema complexo e cheio de possibilidades, dependendo sempre do que se quer alcançar e implementar.

Em suma, a realização deste trabalho foi uma experiência super enriquecedora em todos os aspetos, desde o trabalho em equipa, até a aprender a implementar novos conceitos e cimentar alguns conhecimentos que não estavam cem por cento claros na nossa cabeça. Apesar de os resultados não terem ido encontra as nossas expectativas e desejos, consideramos que todo o processo e tempo perdido valeram a pena para nosso enriquecimento ao nos fazer aprender conhecimentos novos como também na aprendizagem da implementação de novos conceitos que ainda não estavam presentes na nossa vida académica.

Bibliografia

- [1] Wikipédia, *Método de Análise e Solução de Problemas*, Acesso em: 9 set. 2024, 2024. URL: https://pt.wikipedia.org/wiki/M%C3%A9todo_de_an%C3%A1lise_e_solu%C3%A7%C3%A3o_de_problemas.
- [2] I. Ahmad, R. Albatal e A. Kheli, “Blockchain for IoT Cybersecurity: Review of Security Concerns and Attack Surfaces”, *Journal of Sensor Networks*, vol. 2, n.º 2, pp. 57–71, 2023. URL: <https://www.mdpi.com/2674-113X/2/2/7>.
- [3] Techtarget, *End-to-End Encryption (E2EE)*, Acesso em: 9 set. 2024, 2024. URL: <https://www.techtarget.com/searchsecurity/definition/end-to-end-encryption-E2EE>.
- [4] *Método de Análise e Solução de Problemas - MASP*, Acesso em: 9 set. 2024, 2023. URL: <https://www.youtube.com/watch?v=hwQbPgvEQyw&t=731s>.
- [5] StackOverflow, *End-to-end encryption in Flutter with Firebase Realtime Database*, Acesso em: 9 set. 2024, 2022. URL: <https://stackoverflow.com/questions/74766374/end-to-end-encryption-flutter-firebase-real-time-database>.
- [6] Wikipédia, *VMware*, Acesso em: 9 set. 2024, 2024. URL: <https://pt.wikipedia.org/wiki/VMware>.
- [7] StackOverflow, *Token-based authentication: application vs server*, Acesso em: 9 set. 2024, 2016. URL: <https://stackoverflow.com/questions/39391391/token-based-authentication-application-vs-server>.

- [8] RedSwitches, *A Guide to Encrypted Databases and Best Practices*, Acesso em: 9 set. 2024, 2023. URL: <https://www.redswitches.com/blog/encrypted-database/>.
- [9] A. Cloud, *SQL Server Best Practices Using Asymmetric Keys to Implement Column Encryption*, Acesso em: 9 set. 2024, 2023. URL: https://www.alibabacloud.com/blog/sql-server-best-practices-using-asymmetric-keys-to-implement-column-encryption_594390.
- [10] A. A. e A. B., “Database Encryption Using Asymmetric Keys: A Case Study”, *ResearchGate*, 2017. URL: https://www.researchgate.net/publication/318295607_Database_Encryption_Using_Asymmetric_Keys_A_Case_Study.
- [11] J. Doe e J. Doe, “Encryption Techniques for Databases”, em *IEEE International Conference on Computer Science and Engineering*, 2017. URL: <https://ieeexplore.ieee.org/document/7968578>.
- [12] S. Viewer, *Microsoft SQL Server 2005 Database Security Guide*, Acesso em: 9 set. 2024, 2015. URL: https://www.stigviewer.com/stig/microsoft_sql_server_2005_database/2015-06-16/finding/V-15142.
- [13] M. Team, *Using a MySQL Keyring for Secret and Asymmetric Encryption*, Acesso em: 9 set. 2024, 2023. URL: <https://dev.mysql.com/blog-archive/using-a-mysql-keyring-secret-and-asymmetric-encryption/>.
- [14] T. Team, *Understanding MySQL Client-Server Protocol Using Python and Wireshark (Part 1)*, Acesso em: 9 set. 2024, 2023. URL: <https://www.turing.com/blog/understanding-mysql-client-server-protocol-using-python-and-wireshark-part-1>.
- [15] TutorialsPoint, *JDBC Tutorial*, Acesso em: 9 set. 2024, 2023. URL: <https://www.tutorialspoint.com/jdbc/index.htm>.
- [16] Baeldung, *JDBC with Java*, Acesso em: 9 set. 2024, 2023. URL: <https://www.baeldung.com/java-jdbc>.

- [17] Sequelize, *Sequelize ORM Documentation*, Acesso em: 9 set. 2024, 2024. URL: <https://sequelize.org>.
- [18] Wireshark Tutorial, Acesso em: 9 set. 2024, 2023. URL: <https://www.youtube.com/watch?v=g5ij7NIPR2s>.
- [19] ProxySQL Tutorial, Acesso em: 9 set. 2024, 2023. URL: <https://www.youtube.com/watch?v=Fbu7z5dXcRs>.
- [20] M. Team, *MySQL Connector APIs Documentation*, Acesso em: 9 set. 2024, 2023. URL: <https://dev.mysql.com/doc/connectors/en/connectors-apis.html>.
- [21] GeeksForGeeks, *Architecture of MySQL*, Acesso em: 9 set. 2024, 2023. URL: <https://www.geeksforgeeks.org/architecture-of-mysql/>.
- [22] Wireshark, *Sample Captures for Network Analysis*, Acesso em: 9 set. 2024, 2023. URL: <https://wiki.wireshark.org/SampleCaptures>.
- [23] M. Team, *MySQL C API Documentation*, Acesso em: 9 set. 2024, 2024. URL: <https://dev.mysql.com/doc/connector-j/en/>.
- [24] ——, *MySQL C API Documentation*, Acesso em: 9 set. 2024, 2024. URL: <https://dev.mysql.com/doc/c-api/8.0/en/>.
- [25] StackOverflow, *MySQL queries coming on TCP instead of MySQL protocol*, Acesso em: 9 set. 2024, 2017. URL: <https://stackoverflow.com/questions/43188560/mysql-queries-coming-on-tcp-instead-of-mysql-protocol>.
- [26] CopyProgramming, *How to use Wireshark to capture MySQL query SQL clearly*, Acesso em: 9 set. 2024, 2024. URL: <https://copyprogramming.com/howto/how-to-use-wireshark-to-capture-mysql-query-sql-clearly>.
- [27] GeeksForGeeks, *TCP Server Client Implementation in C*, Acesso em: 9 set. 2024, 2023. URL: <https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/>.

- [28] M. Team, *MySQL Server Structs Documentation*, Acesso em: 9 set. 2024, 2023. URL: https://dev.mysql.com/doc/dev/mysql-server/latest/structs_mysql_greetings.html#details.
- [29] P. Team, *Installing ProxySQL*, Acesso em: 9 set. 2024, 2023. URL: <https://proxysql.com/documentation/installing-proxysql/>.
- [30] A. Habib, *Setting Up ProxySQL for Load Balancing MySQL Replication*, Acesso em: 9 set. 2024, 2023. URL: <https://medium.com/@ahosanhabib.974/setting-up-proxysql-for-load-balancing-mysql-replication-1f00d16d82e8>.
- [31] P. Team, *Backend Server Configuration*, Acesso em: 9 set. 2024, 2023. URL: <https://proxysql.com/documentation/backend-server-configuration/>.
- [32] ——, *Backend Server Configuration*, Acesso em: 9 set. 2024, 2023. URL: <https://proxysql.com/documentation/proxysql-configuration/>.
- [33] G. Nethalo, *ProxySQL basics - Master-Slave configuration*, Acesso em: 9 set. 2024, 2023. URL: <https://github.com/nethalo/proxysql-basics-master-slave>.
- [34] tutorialproxy, Acesso em: 9 set. 2024, 2023. URL: <https://www.youtube.com/watch?v=nZB9ScxWxjY>.
- [35] P. Team, *ProxySQL Read Write Split How-To*, Acesso em: 9 set. 2024, 2023. URL: <https://proxysql.com/documentation/proxysql-read-write-split-howto/>.

Apêndice A

Proposta Original do Projeto

Curso de Licenciatura em Engenharia Informática

Projeto 3º Ano - Ano letivo de 2023/2024

A solution for Data Integrity in Relational Database Security

Inteligência artificial	Multimédia/Realidade aumentada/...
Aplicações móveis	Redes e gestão de sistemas
Plataformas web	X *Cibersegurança

* - A especificar pelo proponente

Orientador: Rui Alves

Coorientador: Tiago Pedrosa

1 Objetivo

Conceber, implementar e validar uma solução que garanta, recorrendo a criptografia assimétrica, a autenticidade dos dados inseridos/lidos numa base de dados relacional entre o cliente e o servidor.

2 Detalhes

Nos dias atuais, os ataques à integridade dos dados gerados nos sistemas de informação são cada vez mais comuns. Em aplicações que necessitem de uma base de dados para se executar, este tipo de ataques muitas vezes acontece porque na fase de desenvolvimento, muitas vezes não são tidas em consideração algumas das medidas de mitigação já existentes (ex: controlo de permissões de acesso do utilizador). Desta forma, com o presente projeto pretende-se construir e idealizar uma solução (proxy), que recorrendo a um par de chaves (private/public), que analisar a assinatura de cada trama de dados, apenas permita acesso a clientes devidamente autenticados. A exploração deste cenário em bases de dados distribuídas poderá ser possível e considerado com objetivo extra.

3 Metodologia de trabalho

- 1 - Levantamento do estado da arte.
- 2 – Exploração de soluções existentes.
- 3 – Exploração do mecanismo de troca de dados entre o cliente e o servidor.
- 4 – Desenvolvimento da camada de código para implementação dos objetivos descritos.
- 5 – Escrita do Relatório.

Dimensão da equipa:	Bernardo Nogueira a47334
Recursos necessários:	Computador

Apêndice B

Outro(s) Apêndice(s)

Exertos de código: db.js

```
1
2
3 const mysql = require('mysql');
4
5 // Configuração da conexão com o MySQL
6 const connection = mysql.createConnection({
7   host: 'localhost',
8   user: 'root',
9   password: 'Rui.aires1',
10  database: 'projeto',
11  port: 3306, // Porta configurada no servidor.
12 });
13
14 // Adiciona um listener para o evento 'connect'
15 connection.connect(() => {
16   console.log('Conectado ao MySQL com sucesso!');
17
18   // Aqui você pode realizar consultas, inserções, atualizações, etc.
19   // Exemplo de consulta simples:
20   connection.query('SELECT 1 + 1 AS resultado', (queryErr, results) => {
21     if (queryErr) {
22       console.error('Erro na consulta:', queryErr.stack);
23       return;
24     }
25
26     console.log('Resultado da consulta:', results[0].resultado);
27   })
28 })
```

```

28 // Fecha a conexão após a consulta
29 connection.end((endErr) => {
30   if (endErr) {
31     console.error('Erro ao fechar a conexão:', endErr.stack);
32     return;
33   }
34
35   console.log('Conexão fechada com sucesso!');
36 });
37 );
38 });

```

manytomany.js

```

1
2 const {Sequelize, DataTypes} = require("sequelize");
3
4 const sequelize = new Sequelize(
5   'exemplodb',
6   'root',
7   'Rui.aires1',
8   {
9     host: 'localhost',
10    dialect: 'mysql'
11  }
12 );
13
14 sequelize.authenticate().then(() => {
15   console.log('Connection has been established successfully.');
16 }).catch((error) => {
17   console.error('Unable to connect to the database.', error);
18 });
19
20 const Student = sequelize.define("students", {
21   student_id: {
22     type: DataTypes.UUID,
23     defaultValue: DataTypes.UUIDV4,
24   },
25   name: {
26     type: DataTypes.STRING,
27     allowNull: false
28   }
29 });

```

```

31 const Course = sequelize.define("courses", {
32   course_name: {
33     type: DataTypes.STRING,
34     allowNull: false
35   }
36 });
37
38 const StudentCourse = sequelize.define('StudentCourse', {
39   id: {
40     type: DataTypes.INTEGER,
41     primaryKey: true,
42     autoIncrement: true,
43     allowNull: false
44   }
45 });
46
47 const course_data = [
48   {course_name : "Science"},
49   {course_name : "Maths"},
50   {course_name : "History"}
51 ]
52
53 const student_data = [
54   {name : "John Baker", courseId: 2},
55   {name : "Max Butler", courseId: 1},
56   {name : "Ryan Fisher", courseId: 3},
57   {name : "Robert Gray", courseId: 2},
58   {name : "Sam Lewis", courseId: 1}
59 ]
60
61 const student_course_data = [
62   {studentId : 1, courseId: 1},
63   {studentId : 2, courseId: 1},
64   {studentId : 2, courseId: 3},
65   {studentId : 3, courseId: 2},
66   {studentId : 1, courseId: 2},
67 ]
68
69 Course.belongsToMany(Student, { through: 'StudentCourse'})
70 Student.belongsToMany(Course, { through: 'StudentCourse'})
71
72 sequelize.sync({ force: true }).then(() => {
73   Course.bulkCreate(course_data, { validate: true }).then(() => {
74     Student.bulkCreate(student_data, { validate: true }).then(() => {

```

```

75     StudentCourse.bulkCreate(student_course_data, { validate: true }).then(() =>
76       {
77         Course.findAll({
78           include: [
79             { model: Student },
80           ],
81         }).then(result => {
82           console.log(result);
83         }).catch((error) => {
84           console.error('Failed to retrieve data : ', error);
85         });
86       }).catch((error) => {
87         console.log(error);
88       });
89     }).catch((error) => {
90       console.log(error);
91     });
92   }).catch((error) => {
93     console.log(error);
94   });
95 }).catch((error) => {
96   console.error('Unable to create table : ', error);
97 });

```

app.js

```

1
2 // api.js
3 const express = require('express');
4 const db = require('./db'); // Importa a configuracao da base de dados
5
6 const app = express();
7 app.use(express.json());
8
9 // Endpoint GET que insere um registo na tabela e gera uma classe de entidade
10 app.get('/inserir-registro', (req, res) => {
11
12   // Gere uma classe de entidade com base na tabela
13   generateEntityClass('Exemplo', (entityClass) => {
14     const registro = {
15       nome: 'Exemplo',
16       idade: 30,
17       email: 'exemplo@email.com'
18     };

```

```

19
20     // Insira um registo na tabela
21     const sql = 'INSERT INTO Exemplo (nome, idade, email) VALUES (?, ?, ?)';
22     db.query(sql, [registro.nome, registro.idade, registro.email], (err, result) => {
23         if (err) {
24             console.error('Erro ao inserir registo: ' + err);
25             res.status(500).send('Erro interno do servidor');
26         } else {
27             console.log('Registro inserido com sucesso');
28             res.status(200).send(entityClass);
29         }
30     });
31 });
32 });
33
34 function generateEntityClass(tableName, callback) {
35
36     // Recuperar informacoes da tabela para gerar a classe de entidade
37     db.query(`DESCRIBE ${tableName}`, (err, result) => {
38         if (err) {
39             console.error('Erro ao obter informacoes da tabela: ' + err);
40             callback('');
41         } else {
42             const entityClass = generateClassFromTable(result, tableName);
43             callback(entityClass);
44         }
45     });
46 }
47
48 function generateClassFromTable(tableInfo, tableName) {
49     const fields = tableInfo.map((row) => row.Field);
50     const classTemplate = `class ${tableName} {\n    ${fields.join(',\n    ')}\n}`;
51
52     return classTemplate;
53 }
54
55 const PORT = process.env.PORT || 3000;
56 app.listen(PORT, () => {
57     console.log(`Servidor em execucao na porta ${PORT}`);
58 });

```

makefile.C

```

2 compile:
3   gcc -Wall -g3 -fsanitize=address -pthread server.c -o server -lmysqlclient
4   gcc -Wall -g3 -fsanitize=address -pthread client.c -o client -lmysqlclient
5 FLAGS      = -L /lib64
6 LIBS       = -lusb-1.0 -l pthread

```

client.C

```

1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <signal.h>
6 #include <unistd.h>
7 #include <sys/types.h>
8 #include <sys/socket.h>
9 #include <netinet/in.h>
10 #include <arpa/inet.h>
11 #include <pthread.h>
12 #include <mysql/mysql.h>
13
14 #define LENGTH 2048
15 #define BUFFER_SIZE 4096
16
17 // Global variables
18 volatile sig_atomic_t flag = 0;
19 int sockfd = 0;
20 char name[32];
21 MYSQL *conn;
22
23 void str_overwrite_stdout() {
24     printf("%s", "> ");
25     fflush(stdout);
26 }
27
28 void str_trim_lf(char *arr, int length) {
29     int i;
30     for (i = 0; i < length; i++) { // trim \n
31         if (arr[i] == '\n') {
32             arr[i] = '\0';
33             break;
34         }
35     }
36 }

```

```

37
38 void catch_ctrl_c_and_exit(int sig) {
39     flag = 1;
40 }
41
42 void send_msg_handler() {
43     char message[LENGTH] = {};
44     char buffer[BUFFER_SIZE] = {};
45
46     while (1) {
47         str_overwrite_stdout();
48         fgets(message, LENGTH, stdin);
49         str_trim_lf(message, LENGTH);
50
51         if (strcmp(message, "exit") == 0) {
52             break;
53         } else {
54             // Utilize sprintf para evitar buffer overflow
55             sprintf(buffer, sizeof(buffer), "%s: %s\n", name, message);
56             if (strlen(buffer) < sizeof(buffer)) {
57                 send(sockfd, buffer, strlen(buffer), 0);
58             } else {
59                 printf("Message too long, not sent.\n");
60             }
61         }
62
63         bzero(message, LENGTH);
64         bzero(buffer, BUFFER_SIZE);
65     }
66     catch_ctrl_c_and_exit(2);
67 }
68
69 void recv_msg_handler() {
70     char message[LENGTH] = {};
71     while (1) {
72         int receive = recv(sockfd, message, LENGTH, 0);
73         if (receive > 0) {
74             printf("%s", message);
75             str_overwrite_stdout();
76         } else if (receive == 0) {
77             break;
78         } else {
79             // -1
80         }
81     }
82 }
```

```

81     memset(message, 0, sizeof(message));
82 }
83 }
84
85
86
87
88
89 // conexao com a base de dados
90 int db_connect() {
91     conn = mysql_init(NULL);
92
93     if (conn == NULL) {
94         fprintf(stderr, "mysql_init() failed\n");
95         return EXIT_FAILURE;
96     }
97
98     // Substitua os seguintes valores pelos do Workbench
99     const char *db_host = "127.0.0.1";
100    const char *db_user = "ruiaires";
101    const char *db_password = "ruiaires";
102    const char *db_name = "teste1";
103    unsigned int db_port = 3306;
104
105    if (mysql_real_connect(conn, db_host, db_user, db_password, db_name, db_port, NULL,
106                           0) == NULL) {
107        fprintf(stderr, "mysql_real_connect() failed: %s\n", mysql_error(conn));
108        mysql_close(conn);
109        return EXIT_FAILURE;
110    }
111
112    printf("Connected to MySQL server\n");
113
114    return EXIT_SUCCESS;
115}
116 void db_disconnect() {
117     if (conn != NULL) {
118         mysql_close(conn);
119         printf("Disconnected from MySQL server\n");
120     }
121 }
122
123 int main(int argc, char **argv) {

```

```

124     if (argc != 2) {
125         printf("Usage: %s <port>\n", argv[0]);
126         return EXIT_FAILURE;
127     }
128
129     char *ip = "127.0.0.1";
130     int port = atoi(argv[1]);
131
132     signal(SIGINT, catch_ctrl_c_and_exit);
133
134     printf("Please enter your name: ");
135     fgets(name, 32, stdin);
136     str_trim_lf(name, strlen(name));
137
138     if (strlen(name) > 32 || strlen(name) < 2) {
139         printf("Name must be less than 30 and more than 2 characters.\n");
140         return EXIT_FAILURE;
141     }
142
143     // Conectar a base de dados
144     if (db_connect() != EXIT_SUCCESS) {
145         return EXIT_FAILURE;
146     }
147     struct sockaddr_in server_addr;
148
149
150
151     /* Socket settings */
152     sockfd = socket(AF_INET, SOCK_STREAM, 0);
153     server_addr.sin_family = AF_INET;
154     server_addr.sin_addr.s_addr = inet_addr(ip);
155     server_addr.sin_port = htons(port);
156
157     // Connect to Server
158     int err = connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr));
159     if (err == -1) {
160         printf("ERROR: connect\n");
161         return EXIT_FAILURE;
162     }
163
164     // Send name
165     send(sockfd, name, 32, 0);
166
167     printf("== WELCOME TO THE CHATROOM ==\n");

```

```

168     pthread_t send_msg_thread;
169     if (pthread_create(&send_msg_thread, NULL, (void *)send_msg_handler, NULL) != 0) {
170         printf("ERROR: pthread\n");
171         return EXIT_FAILURE;
172     }
173
174
175     pthread_t recv_msg_thread;
176     if (pthread_create(&recv_msg_thread, NULL, (void *)recv_msg_handler, NULL) != 0) {
177         printf("ERROR: pthread\n");
178         return EXIT_FAILURE;
179     }
180
181     while (1) {
182         if (flag) {
183             printf("\nBye\n");
184             break;
185         }
186     }
187
188     close(sockfd);
189
190     db_disconnect();
191
192     return EXIT_SUCCESS;
193 }
```

server.C

```

1
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <arpa/inet.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <unistd.h>
8 #include <errno.h>
9 #include <string.h>
10 #include <pthread.h>
11 #include <sys/types.h>
12 #include <signal.h>
13 #include <mysql/mysql.h>
14
15 #define MAX_CLIENTS 100
```

```

16 #define BUFFER_SZ 2048
17
18 static _Atomic unsigned int cli_count = 0;
19 static int uid = 10;
20
21 /* Client structure */
22 typedef struct
23 {
24     struct sockaddr_in address;
25     int sockfd;
26     int uid;
27     char name[32];
28 } client_t;
29
30 client_t *clients[MAX_CLIENTS];
31
32 pthread_mutex_t clients_mutex = PTHREAD_MUTEX_INITIALIZER;
33
34 void str_overwrite_stdout()
35 {
36     printf("\r%s", "> ");
37     fflush(stdout);
38 }
39
40 void str_trim_lf(char *arr, int length)
41 {
42     int i;
43     for (i = 0; i < length; i++)
44     { // trim \n
45         if (arr[i] == '\n')
46         {
47             arr[i] = '\0';
48             break;
49         }
50     }
51 }
52
53 void print_client_addr(struct sockaddr_in addr)
54 {
55     printf("%d.%d.%d.%d",
56            addr.sin_addr.s_addr & 0xff,
57            (addr.sin_addr.s_addr & 0xff00) >> 8,
58            (addr.sin_addr.s_addr & 0xff0000) >> 16,
59            (addr.sin_addr.s_addr & 0xff000000) >> 24);

```

```

60 }
61
62 /* Add clients to queue */
63 void queue_add(client_t *cl)
64 {
65     pthread_mutex_lock(&clients_mutex);
66
67     for (int i = 0; i < MAX_CLIENTS; ++i)
68     {
69         if (!clients[i])
70         {
71             clients[i] = cl;
72             break;
73         }
74     }
75
76     pthread_mutex_unlock(&clients_mutex);
77 }
78
79 /* Remove clients to queue */
80 void queue_remove(int uid)
81 {
82     pthread_mutex_lock(&clients_mutex);
83
84     for (int i = 0; i < MAX_CLIENTS; ++i)
85     {
86         if (clients[i])
87         {
88             if (clients[i]->uid == uid)
89             {
90                 clients[i] = NULL;
91                 break;
92             }
93         }
94     }
95
96     pthread_mutex_unlock(&clients_mutex);
97 }
98
99 /* Send message to all clients except sender */
100 void send_message(char *s, int uid)
101 {
102     pthread_mutex_lock(&clients_mutex);

```

```

104     for (int i = 0; i < MAX_CLIENTS; ++i)
105     {
106         if (clients[i])
107         {
108             if (clients[i]->uid != uid)
109             {
110                 if (write(clients[i]->sockfd, s, strlen(s)) < 0)
111                 {
112                     perror("ERROR: write to descriptor failed");
113                     break;
114                 }
115             }
116         }
117     }
118
119     pthread_mutex_unlock(&clients_mutex);
120 }
121
122
123
124 /* Inserir mensagem na base de dados */
125 void insert_message_into_database(const char *name, const char *message)
126 {
127     MYSQL *conn;
128     MYSQL_RES *res;
129     MYSQL_ROW row;
130
131     conn = mysql_init(NULL);
132
133     if (conn == NULL)
134     {
135         fprintf(stderr, "mysql_init() failed\n");
136         return;
137     }
138
139     if (mysql_real_connect(conn, "127.0.0.1", "ruiaires", "ruiaires", "teste1", 3306,
140                           NULL, 0) == NULL)
141     {
142         fprintf(stderr, "mysql_real_connect() failed\n");
143         mysql_close(conn);
144         return;
145     }
146     char query[256];

```

```

147     sprintf(query, "INSERT INTO chat_messages (user, message) VALUES ('%s', '%s')", name,
148             message);
149
150     if (mysql_query(conn, query))
151     {
152         fprintf(stderr, "INSERT failed: %s\n", mysql_error(conn));
153     }
154
155     mysql_close(conn);
156
157
158
159 /* Handle all communication with the client */
160 void *handle_client(void *arg)
161 {
162     char buff_out[BUFFER_SZ];
163     char name[32];
164     int leave_flag = 0;
165
166     cli_count++;
167     client_t *cli = (client_t *)arg;
168
169     // Name
170     if (recv(cli->sockfd, name, 32, 0) <= 0 || strlen(name) < 2 || strlen(name) >= 32 - 1)
171     {
172         printf("Didn't enter the name.\n");
173         leave_flag = 1;
174     }
175     else
176     {
177         strcpy(cli->name, name);
178         sprintf(buff_out, "%s has joined\n", cli->name);
179         printf("%s", buff_out);
180         send_message(buff_out, cli->uid);
181     }
182
183     bzero(buff_out, BUFFER_SZ);
184
185     while (1)
186     {
187         if (leave_flag)
188         {

```

```

189         break;
190     }
191
192     int receive = recv(cli->sockfd, buff_out, BUFFER_SZ, 0);
193     if (receive > 0)
194     {
195         printf("Received message from %s: %s\n", cli->name, buff_out);
196         if (strlen(buff_out) > 0)
197         {
198             send_message(buff_out, cli->uid);
199
200             str_trim_lf(buff_out, strlen(buff_out));
201             printf("%s -> %s\n", buff_out, cli->name);
202
203             // Inserir a mensagem no banco de dados
204             insert_message_into_database(cli->name, buff_out);
205         }
206     }
207     else if (receive == 0 || strcmp(buff_out, "exit") == 0)
208     {
209         sprintf(buff_out, "%s has left\n", cli->name);
210         printf("%s", buff_out);
211         send_message(buff_out, cli->uid);
212         leave_flag = 1;
213     }
214     else
215     {
216         printf("ERROR: -1\n");
217         leave_flag = 1;
218     }
219
220     bzero(buff_out, BUFFER_SZ);
221 }
222
223 /* Delete client from queue and yield thread */
224 close(cli->sockfd);
225 queue_remove(cli->uid);
226 free(cli);
227 cli_count--;
228 pthread_detach(pthread_self());
229
230 return NULL;
231 }
232

```

```

233 int main(int argc, char **argv)
234 {
235     if (argc != 2)
236     {
237         printf("Usage: %s <port>\n", argv[0]);
238         return EXIT_FAILURE;
239     }
240
241     char *ip = "127.0.0.1";
242     int port = atoi(argv[1]);
243     int option = 1;
244     int listenfd = 0, connfd = 0;
245     struct sockaddr_in serv_addr;
246     struct sockaddr_in cli_addr;
247     pthread_t tid;
248
249     /* Socket settings */
250     listenfd = socket(AF_INET, SOCK_STREAM, 0);
251     serv_addr.sin_family = AF_INET;
252     serv_addr.sin_addr.s_addr = inet_addr(ip);
253     serv_addr.sin_port = htons(3306);
254
255     /* Ignore pipe signals */
256     signal(SIGPIPE, SIG_IGN);
257
258     if (setsockopt(listenfd, SOL_SOCKET, (SO_REUSEPORT | SO_REUSEADDR), (char *)&option,
259                   sizeof(option)) < 0)
260     {
261         perror("ERROR: setsockopt failed");
262         return EXIT_FAILURE;
263     }
264
265     /* Bind */
266     if (bind(listenfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
267     {
268         perror("ERROR: Socket binding failed");
269         return EXIT_FAILURE;
270     }
271
272     /* Listen */
273     if (listen(listenfd, 10) < 0)
274     {
275         perror("ERROR: Socket listening failed");
276         return EXIT_FAILURE;

```

```

276     }
277
278     printf("== WELCOME TO THE CHAT;] HAVE FUN! ==\n");
279
280     while (1)
281     {
282         socklen_t clilen = sizeof(cli_addr);
283         connfd = accept(listenfd, (struct sockaddr *)&cli_addr, &clilen);
284         printf("ola rui");
285         /* Check if max clients is reached */
286         if ((cli_count + 1) == MAX_CLIENTS)
287         {
288             printf("Max clients reached. Rejected: ");
289             print_client_addr(cli_addr);
290             printf(":%d\n", cli_addr.sin_port);
291             close(connfd);
292             continue;
293         }
294
295         /* Client settings */
296         client_t *cli = (client_t *)malloc(sizeof(client_t));
297         cli->address = cli_addr;
298         cli->sockfd = connfd;
299         cli->uid = uid++;
300
301         /* Add client to the queue and fork thread */
302         queue_add(cli);
303         pthread_create(&tid, NULL, &handle_client, (void *)cli);
304
305         /* Reduce CPU usage */
306         sleep(1);
307     }
308
309     return EXIT_SUCCESS;
310 }
```

segundaversÁčo

```

1
2
3 #include <mysql/mysql.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
```

```

7 #include <unistd.h>
8 #include <arpa/inet.h>
9
10#define MAX_MSG_SIZE 1024
11
12int main() {
13    int socket_fd, client_fd;
14    MYSQL *conn;
15
16    // Initialize the MySQL connection object
17    conn = mysql_init(NULL);
18    if (conn == NULL) {
19        fprintf(stderr, "mysql_init() failed\n");
20        return 1;
21    }
22
23    // Create a socket
24    if ((socket_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
25        perror("socket");
26        mysql_close(conn); // Close MySQL connection if socket creation fails
27        return 1;
28    }
29
30    // Set up the server address structure
31    struct sockaddr_in server_addr, client_addr;
32    memset(&server_addr, 0, sizeof(server_addr));
33    server_addr.sin_family = AF_INET;
34    server_addr.sin_addr.s_addr = INADDR_ANY; // Accept connections from any IP address
35    server_addr.sin_port = htons(3306); // Replace with your server's port
36
37    // Bind the socket to the server address
38    if (bind(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1) {
39        perror("bind");
40        close(socket_fd);
41        mysql_close(conn);
42        return 1;
43    }
44
45    // Listen for incoming connections
46    if (listen(socket_fd, 5) == -1) {
47        perror("listen");
48        close(socket_fd);
49        mysql_close(conn);
50        return 1;

```

```

51     }
52
53     // Accept connections and handle client requests
54     while(1) {
55         socklen_t client_len = sizeof(client_addr);
56         // Accept a connection from a client
57         if ((client_fd = accept(socket_fd, (struct sockaddr *)&client_addr, &client_len))
58             == -1) {
59             perror("accept");
60             continue;
61         }
62         printf("Client connected\n");
63
64         // Receive data from the client
65         char recv_buffer[MAX_MSG_SIZE];
66         ssize_t bytes_received;
67         if ((bytes_received = recv(client_fd, recv_buffer, MAX_MSG_SIZE, 0)) <= 0) {
68             if (bytes_received == 0) {
69                 printf("Client disconnected\n");
70             } else {
71                 perror("recv");
72             }
73             close(client_fd);
74             continue;
75         }
76
77         // Submit received data to the MySQL database
78         if (mysql_real_query(conn, recv_buffer, strlen(recv_buffer)) != 0) {
79             fprintf(stderr, "mysql_real_query() failed: %s\n", mysql_error(conn));
80             close(client_fd);
81             continue;
82         }
83
84         // Receive the result from the database
85         MYSQL_RES *result = mysql_store_result(conn);
86         if (result == NULL) {
87             fprintf(stderr, "mysql_store_result() failed: %s\n", mysql_error(conn));
88             close(client_fd);
89             continue;
90         }
91
92         // Send the result back to the client
93         MYSQL_ROW row;

```

```

94     while ((row = mysql_fetch_row(result)) != NULL) {
95         // Assuming each row is a string and sending it back to the client
96         ssize_t bytes_sent = send(client_fd, row[0], strlen(row[0]), 0);
97         if (bytes_sent == -1) {
98             perror("send");
99             mysql_free_result(result);
100            close(client_fd);
101            break;
102        }
103    }
104
105    // Free the result
106    mysql_free_result(result);
107
108    // Close the client connection
109    close(client_fd);
110}
111
112 // Close the MySQL connection and socket
113 mysql_close(conn);
114 close(socket_fd);
115
116 return 0;
117}

```

estrutura greeting

```

1
2
3 // Initialize the MySQL connection object
4 conn = mysql_init(NULL);
5 if (conn == NULL) {
6     fprintf(stderr, "mysql_init() failed\n");
7     close(socket_fd);
8     return 1;
9 }
10
11 // Attach the socket descriptor to the MySQL connection object
12 if (mysql_real_connect(conn, NULL, "username", "password", "database_name", 0, socket_fd,
13     0) == NULL) {
14     fprintf(stderr, "mysql_real_connect() failed: %s\n", mysql_error(conn));
15     close(socket_fd);
16     mysql_close(conn);
17     return 1;
18 }

```

```

17 }
18
19
20 // Create a socket
21 if ((socket_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
22     perror("socket");
23     return 1;
24 }
25
26 // Set up the server address structure
27 struct sockaddr_in server_addr;
28 memset(&server_addr, 0, sizeof(server_addr));
29 server_addr.sin_family = AF_INET;
30 server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); // Replace with your server's IP
31 server_addr.sin_port = htons(3306); // Replace with your server's port
32
33 // accept questions
34
35
36 //greeting message mysql
37
38
39 // fazer depois de tudo que estiver para tras ficar estavel.
40 while(1)
41 {
42     //recv client workebench
43
44
45     //submeter para o unix socket da BD
46
47
48     //fazer send para o cliente workebench
49
50 }
51
52
53
54
55
56 // Close the connection
57 mysql_close(conn);
58 close(socket_fd);
59

```

```
60 return 0;  
61 }
```

codigofinal

```
1  
2 #include <sys/socket.h>  
3 #include <netinet/in.h>  
4 #include <stdio.h>  
5 #include <unistd.h>  
6 #include <string.h>  
7 #include <iostream>  
8  
9 // Definições de porta e buffer  
10#define PORT 3306 // Porta alterada para 3306  
11#define BUF_SIZE 4096  
12  
13// Estrutura para status do serviço MySQL  
14typedef enum {  
15    MYSQL_SERVICE_OK,  
16    MYSQL_SERVICE_ERROR  
17} mysql_service_status_t;  
18  
19// Classe s_mysql_greetings  
20class s_mysql_greetings {  
21public:  
22    static mysql_service_status_t say_hello(const char **hello_string) {  
23        if (hello_string && *hello_string) {  
24            std::cout << "Hello, " << *hello_string << "!" << std::endl;  
25            return MYSQL_SERVICE_OK;  
26        }  
27        return MYSQL_SERVICE_ERROR;  
28    }  
29};  
30  
31// Função principal do servidor  
32int main() {  
33    int server_fd, client_fd;  
34    struct sockaddr_in server_addr;  
35    char buffer[BUF_SIZE];  
36    ssize_t num_bytes;  
37  
38    // Mensagem de saudação para simular a resposta do MySQL  
39    const char *greeting = "World";
```

```

40     mysql_service_status_t (*say_hello_ptr)(const char **) = s_mysql_greetings::say_hello
41     ;
42
43     // Criar socket do servidor
44     server_fd = socket(AF_INET, SOCK_STREAM, 0);
45     if (server_fd == -1) {
46         perror("Falha ao criar o socket");
47         return -1;
48     }
49
50     // Limpar a estrutura de endereço do servidor e configurar
51     memset(&server_addr, 0, sizeof(server_addr));
52     server_addr.sin_family = AF_INET;
53     server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
54     server_addr.sin_port = htons(PORT);
55
56     // Bind socket ao endereço e porta
57     if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1) {
58         perror("Falha ao fazer bind");
59         close(server_fd);
60         return -1;
61     }
62
63     // Listen para conexões de clientes
64     if (listen(server_fd, 10) == -1) {
65         perror("Falha ao escutar no socket");
66         close(server_fd);
67         return -1;
68     }
69     printf("Servidor escutando na porta %d...\n", PORT);
70
71     // Aceitar a conexão de um cliente
72     client_fd = accept(server_fd, NULL, NULL);
73     if (client_fd == -1) {
74         perror("Falha ao aceitar a conexão");
75         close(server_fd);
76         return -1;
77     }
78
79     printf("Cliente conectado.\n");
80
81     // Ler e processar query do cliente
82     while ((num_bytes = recv(client_fd, buffer, BUF_SIZE, 0)) > 0) {

```

```

83     buffer[num_bytes] = '\0'; // Finalizar a string recebida
84     printf("Query recebida do cliente: %s\n", buffer);
85
86     // Simular o processamento de uma saudação MySQL
87     mysql_service_status_t status = say_hello_ptr(&greeting);
88     const char *response;
89
90     if (status == MYSQL_SERVICE_OK) {
91         response = "Query recebida e saudada com sucesso!";
92     } else {
93         response = "Falha ao processar a saudação!";
94     }
95
96     // Enviar resposta ao cliente
97     send(client_fd, response, strlen(response), 0);
98 }
99
100    // Fechar sockets
101    close(client_fd);
102    close(server_fd);
103
104    return 0;
105 }
```

docker-compose.yaml

```

1
2 docker-compose.yaml:
3
4 version: '3'
5 services:
6     proxy:
7         image: proxysql/proxysql:latest
8         restart: always
9         ports:
10            - "6034:6032"
11            - "6035:6033"
12         volumes:
13            - /data/proxysql:/var/lib/proxysql
14         networks:
```

```

15    proxysql_mysqlnet:
16        ipv4_address: 172.20.0.2
17
18 mysql_primary:
19     image: mysql:5.7
20     restart: always
21     environment:
22         MYSQL_ROOT_PASSWORD: rootpassword
23         MYSQL_DATABASE: mydatabase
24         MYSQL_USER: myuser
25         MYSQL_PASSWORD: mypassword
26     ports:
27         - "3307:3306"
28     volumes:
29         - /data/mysql_primary:/var/lib/mysql
30     networks:
31         proxysql_mysqlnet:
32             ipv4_address: 172.20.0.3
33
34 mysql_replica1:
35     image: mysql:5.7
36     restart: always
37     environment:
38         MYSQL_ROOT_PASSWORD: rootpassword
39         MYSQL_DATABASE: mydatabase
40         MYSQL_USER: myuser
41         MYSQL_PASSWORD: mypassword
42     ports:
43         - "3308:3306"
44     volumes:
45         - /data/mysql_replica1:/var/lib/mysql
46     networks:
47         proxysql_mysqlnet:
48             ipv4_address: 172.20.0.4
49

```

```

50 mysql_replica2:
51   image: mysql:5.7
52   restart: always
53   environment:
54     MYSQL_ROOT_PASSWORD: rootpassword
55     MYSQL_DATABASE: mydatabase
56     MYSQL_USER: myuser
57     MYSQL_PASSWORD: mypassword
58   ports:
59     - "3309:3306"
60   volumes:
61     - /data/mysql_replica2:/var/lib/mysql
62   networks:
63     proxysql_mysqlnet:
64       ipv4_address: 172.20.0.5
65
66 networks:
67 proxysql_mysqlnet:
68   driver: bridge
69   ipam:
70     config:
71       - subnet: "172.19.20.0/16"

```

Databases.txt

```

1
2 -- My mock data
3
4 username=root
5 password=4pYTzpysTDjNsQKe
6
7
8 monitor_username=monitor
9 monitor_password=TsQDRdZNfh5Wvhm8
10
11
12 Master Database:
13
14   url=172.19.0.3 # IP do MySQL Primário

```

```

15
16 Replica1:
17
18     url=172.19.0.4 # IP da Replica 1
19
20 Replica2:
21
22     url=172.19.0.5 # IP da Replica 2

```

create-monitor.SQL

```

1
2 -- Suggested Permission --
3 GRANT SELECT, REPLICATION CLIENT, SHOW DATABASES, SUPER, PROCESS
4 ON .
5 TO 'mysqluser'@'localhost'
6 IDENTIFIED BY 'agent_password';
7
8 -- permissions you will actually need --
9 GRANT SELECT, REPLICATION CLIENT, SHOW DATABASES, PROCESS
10 ON .
11 TO 'monitor'@'%'
12 IDENTIFIED BY 'TsQDRdZnfh5WVhm8';

```

script.lua

```

1
2 -- /etc/proxysql/lua/query_routing.lua
3 function route_query(query)
4     proxy.log("Query: " .. query)
5     if string.match(query, "^SELECT") then
6         proxy.connection.backend_ndx = 20
7     else
8         proxy.connection.backend_ndx = 10
9     end
10 end

```