

**Universidade Federal do Rio de Janeiro
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia**



**Programa de Engenharia de Sistemas e
Computação - PESC**

*CPS-769 - Introduction to Artificial Intelligence and
Generative Learning*

Professores:

Edmundo de Souza e Silva (PESC/COPPE/UFRJ)

Rosa M. Leão (PESC/COPPE/UFRJ)

Participação Especial: Gaspare Bruno (Diretor de Inovação,
ANLIX)

Trabalho Final - Lista de Exercícios 06

Luiz Henrique Souza Caldas / Luis Paulo Albuquerque Guedes
lhscaldas@cos.ufrj.br / luis.guedes@coppe.ufrj.br

19 de setembro de 2024

1 Introdução

O presente trabalho tem como objetivo desenvolver uma aplicação que utiliza inteligência artificial para analisar a Qualidade de Experiência (QoE) em transmissões de vídeo. O foco é a criação de uma interface que aceite perguntas dos usuários em linguagem natural e forneça respostas, também em linguagem natural, baseadas nos dados de bitrate e latência. Para tal, a aplicação fará uso das APIs da OpenAI, implementando o raciocínio *Chain of Thought* (CoT) e *function calling* para processar os dados fornecidos, analisar a qualidade da transmissão e responder de maneira dinâmica às questões dos usuários.

O contexto do trabalho está relacionado aos desafios enfrentados por serviços de streaming de vídeo, nos quais a latência entre o servidor e o cliente, bem como a taxa de transmissão, são fatores críticos. A aplicação deverá integrar a análise dos dados de latência e bitrate para fornecer respostas a perguntas como "Qual cliente tem a pior qualidade de recepção de vídeo?" ou "Qual é a melhor estratégia de troca de servidor para otimizar a qualidade de experiência do cliente?". O trabalho exige a implementação de uma lógica de raciocínio em etapas (CoT) para decompor e resolver questões complexas, além da utilização de chamadas de função para realizar cálculos e normalizações necessárias no conjunto de dados.

Os dados fornecidos incluem medições de bitrate e latência, e o principal desafio reside na combinação dessas informações para calcular a QoE, identificar pontos críticos na transmissão e recomendar soluções para otimizar a experiência do usuário.

2 Conceitos Relevantes

2.1 Qualidade de Experiência (QoE)

A *Qualidade de Experiência* (QoE) refere-se à percepção subjetiva do usuário final sobre a qualidade de um serviço, como o streaming de vídeo, com base em fatores técnicos e não técnicos. A QoE é influenciada por parâmetros como a taxa de transmissão de dados (*bitrate*) e a latência (*Round-Trip Time* - RTT), que impactam diretamente a fluidez da transmissão e a qualidade visual. Uma alta QoE é alcançada quando o serviço proporciona uma experiência satisfatória ao usuário, com alta qualidade de vídeo e baixa interrupção, resultante de uma boa gestão de rede e otimização dos recursos de transmissão.

2.2 Chain of Thought (CoT)

O conceito de *Chain of Thought* (CoT) está relacionado ao processo de raciocínio utilizado por modelos de inteligência artificial para resolver problemas complexos de forma sequencial. No contexto deste trabalho, o CoT é implementado para analisar e responder perguntas dos usuários. O modelo divide a pergunta em etapas lógicas e interdependentes, processa cada uma dessas etapas e gera uma resposta clara e explicativa. Esse método permite que a aplicação lide com perguntas mais complexas ao seguir uma sequência estruturada de passos para chegar à resposta correta.

subsectionFunction Calling O *function calling* refere-se à capacidade de uma aplicação de chamar funções específicas durante o processamento dos dados, a fim de realizar cálculos ou operações. No caso deste trabalho, a aplicação faz uso de *function calling* para calcular a QoE e outras métricas de desempenho de vídeo, como a normalização dos dados de *bitrate* e *latência*, ou para determinar os melhores servidores para otimizar a transmissão. Ao chamar essas funções de forma dinâmica, a aplicação pode responder de maneira eficiente a perguntas específicas dos usuários com base nos dados fornecidos.

2.3 Normalização de Dados

A normalização de dados é um processo utilizado para ajustar os valores de diferentes variáveis a uma escala comum, facilitando a comparação e análise. Neste trabalho, a normalização *min-max* é utilizada para trazer os valores de *bitrate* e *latência* para uma faixa comum, garantindo que ambos tenham o mesmo peso no cálculo da QoE. Esse processo é fundamental para evitar que variáveis com escalas diferentes influenciem de maneira desproporcional os resultados finais da análise de qualidade.

2.4 Bitrate e Latência

O *bitrate* representa a taxa de transmissão de dados em uma rede, medido em kilobits por segundo (kbps). Ele define a qualidade e a quantidade de dados transmitidos por unidade de tempo. Já a *latência* (RTT) refere-se ao tempo que um pacote de dados leva para ser enviado do cliente ao servidor e de volta ao cliente, medido em milissegundos (ms). Ambos os parâmetros são essenciais para determinar a qualidade de uma transmissão de vídeo, influenciando diretamente a QoE.

3 Análise Exploratória dos Dados

Essa seção trata-se da Análise Exploratória de Dados (EDA), que visa explorar e entender a estrutura do conjunto de dados, oferecendo uma visão inicial de suas principais características. A EDA utiliza métodos estatísticos e gráficos para investigar a distribuição das variáveis, identificar *outliers*, analisar correlações e padrões. Por meio de ferramentas como histogramas, *boxplots*, gráficos de dispersão e matrizes de correlação, é possível obter uma visão geral dos dados e orientar as próximas etapas de modelagem e análise. Essa abordagem permite garantir que os dados estão adequadamente preparados para análises posteriores.

Inicialmente, foram plotados os registros de *bitrate* e RTT (latência) em função do tempo, para ver como eles se comportam no período registrado no banco de dados. Os gráficos podem ser visualizados nas figuras 1 a 4.

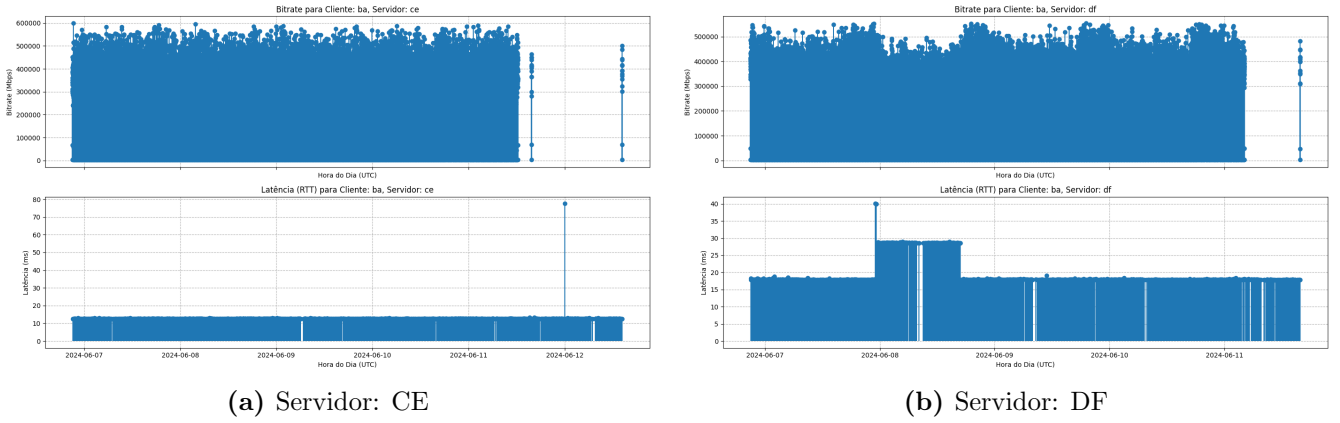


Figura 1: Registros de *bitrate* e de rtt para o cliente BA em todo o período do *dataset* (parte 1).

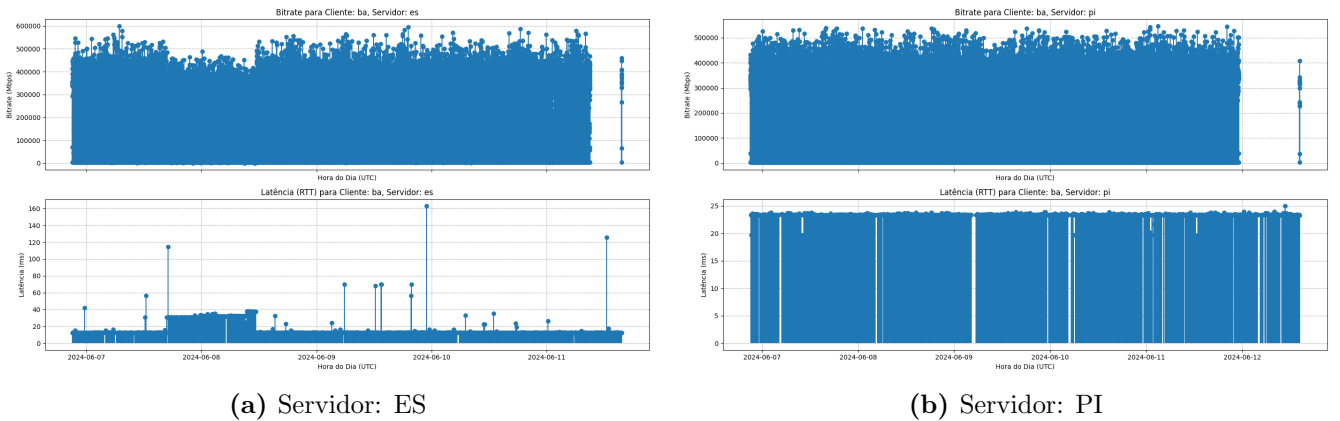
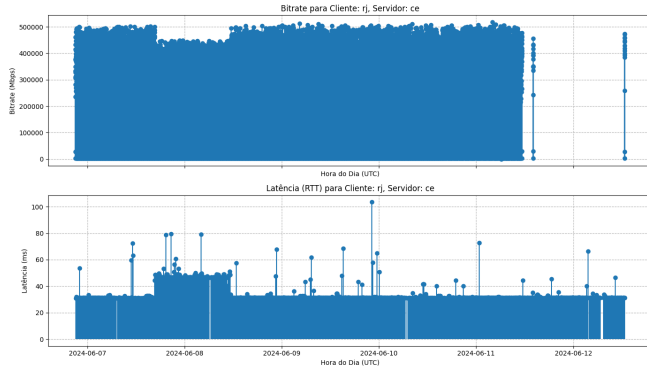
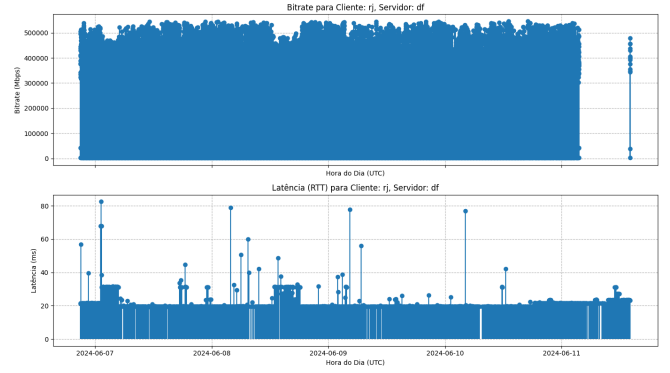


Figura 2: Registros de *bitrate* e de rtt para o cliente BA em todo o período do *dataset* (parte 2).

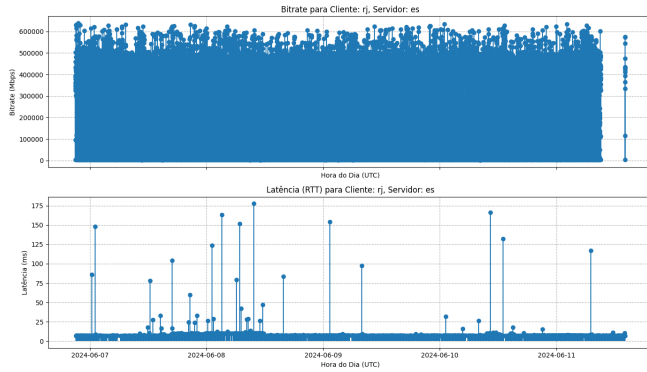


(a) Servidor: CE

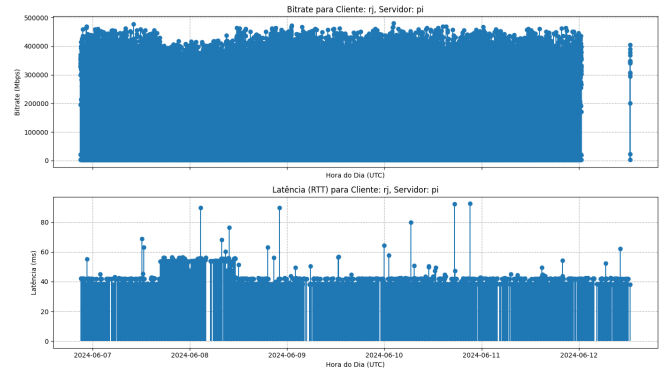


(b) Servidor: DF

Figura 3: Registros de *bitrate* e de *rtt* para o cliente RJ em todo o período do *dataset* (parte 1).



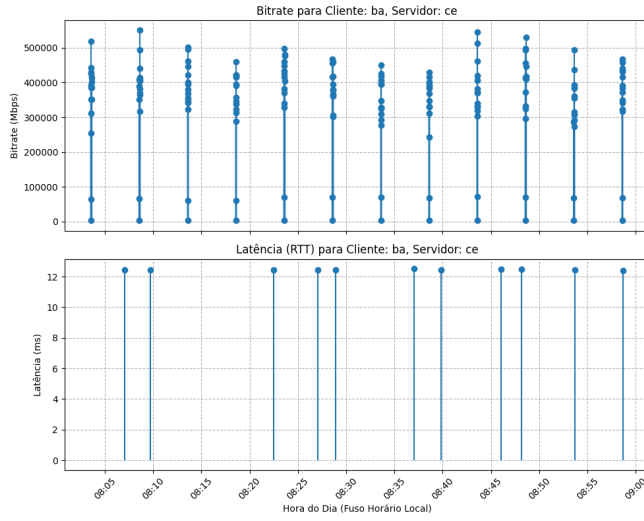
(a) Servidor: CE



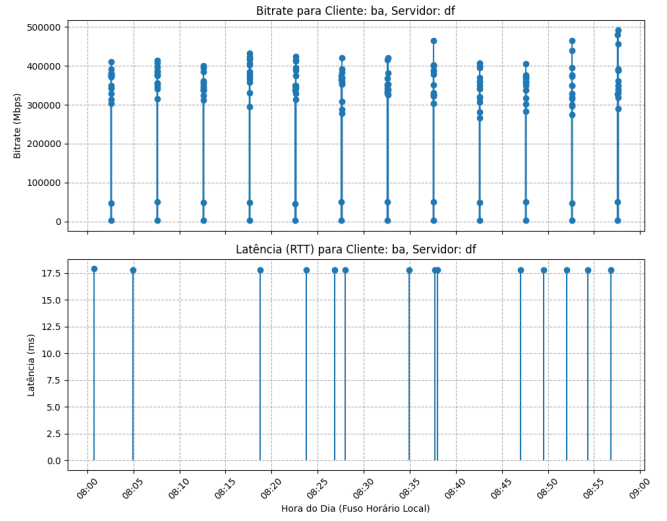
(b) Servidor: PI

Figura 4: Registros de *bitrate* e de *rtt* para o cliente RJ em todo o período do *dataset* (parte 2).

Para analisar o comportamento da distribuição de *bitrate* e RTT em um intervalo de tempo mais restrito, delimitamos o período entre 08:00 e 09:00 do dia 7 de junho de 2024. A partir dessa seleção temporal, foram gerados gráficos que permitem uma análise detalhada dessas variáveis durante o período especificado, possibilitando uma observação mais precisa de seu comportamento, o que pode ser observado nas figuras de 5 a 8.

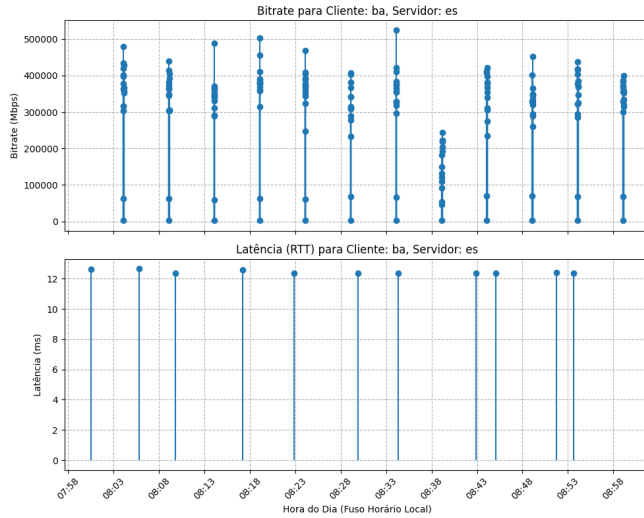


(a) Servidor: CE

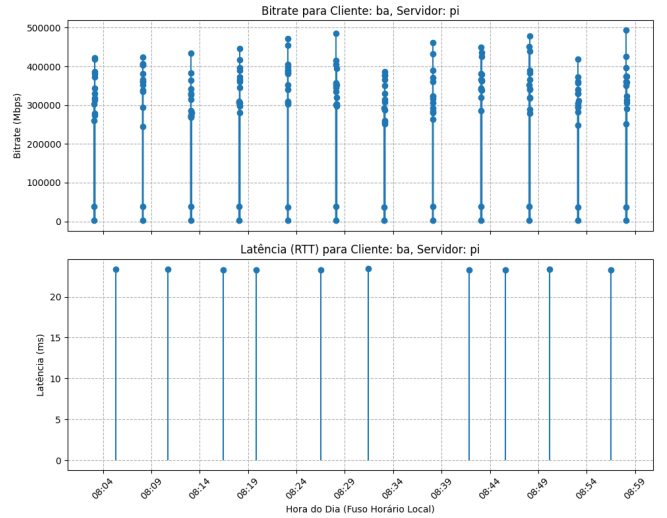


(b) Servidor: DF

Figura 5: Plotagem de *bitrate* e de *rtt* do cliente BA para o período específico (parte 1).

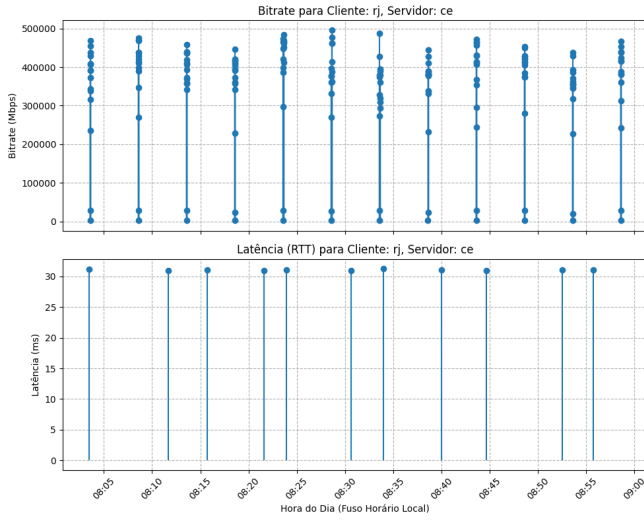


(a) Servidor: ES

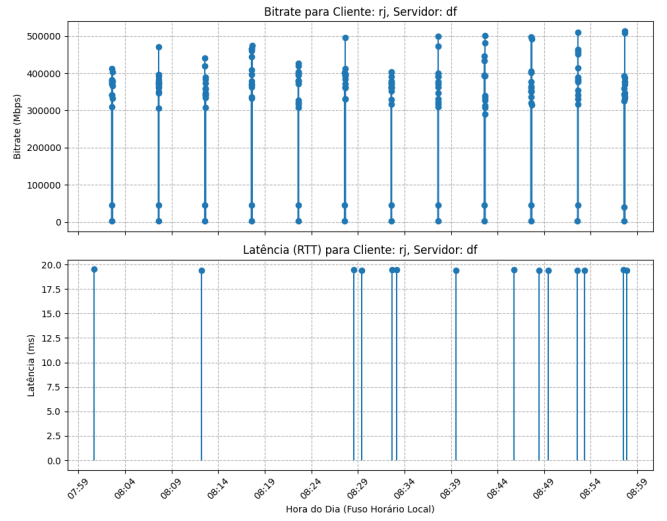


(b) Servidor: PI

Figura 6: Plotagem de *bitrate* e de *rtt* do cliente BA para o período específico (parte 2).

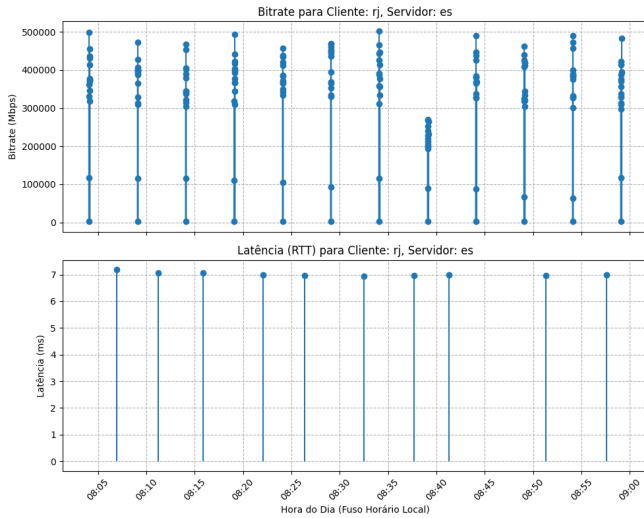


(a) Servidor: CE

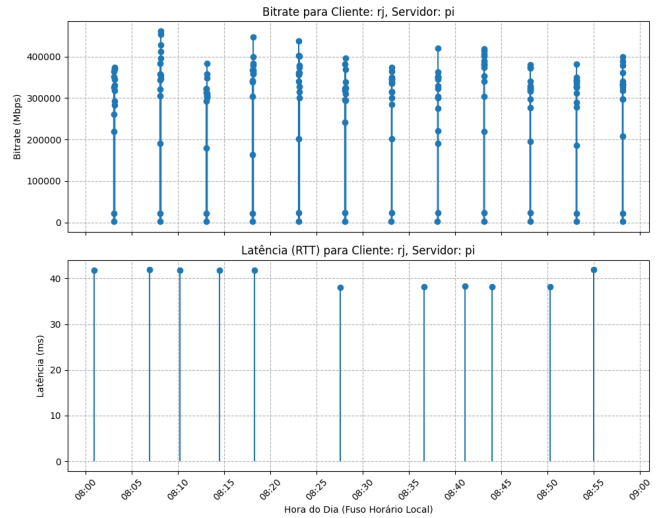


(b) Servidor: DF

Figura 7: Plotagem de *bitrate* e de *rtt* do cliente RJ para o período específico (parte 1).



(a) Servidor: ES



(b) Servidor: PI

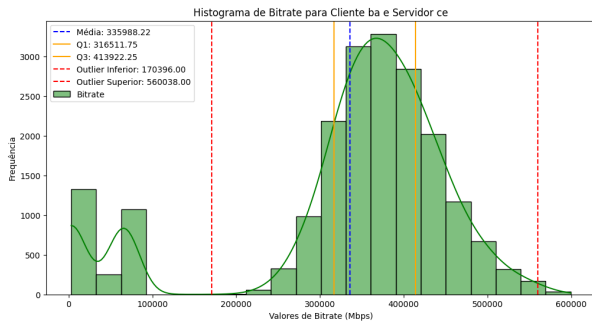
Figura 8: Plotagem de *bitrate* e de *rtt* do cliente RJ para o período específico (parte 2).

A partir da análise dos gráficos de registros de *bitrate* e de *rtt* ao longo do período determinado, é possível notar que a porcentagem de coincidência temporal entre a medida do RTT e do *bitrate* é bastante baixa. Implementando um código em *python* para verificar isso, temos o resultado da tabela 1, em que a coincidência situa-se em torno de 5% para cada par cliente-servidor.

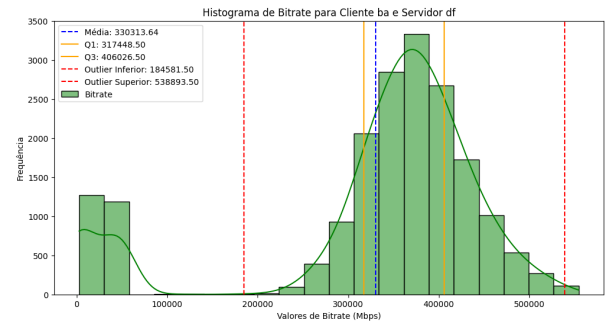
Tabela 1: Coincidência Temporal entre *bitrate* e RTT por Cliente-Servidor

Cliente	Servidor	Qtd. de <i>bitrate</i>	Qtd. de rtt	Qtd. de medições coincidentes no tempo	% de medições coincidentes no tempo
BA	CE	19860	1492	89	5.97 %
BA	DF	18480	1347	73	5.42 %
BA	ES	19305	1283	92	7.17 %
BA	PI	21900	1361	89	6.54 %
RJ	CE	18665	1487	50	3.36 %
RJ	DF	18360	1339	71	5.30 %
RJ	ES	19290	1262	70	5.55 %
RJ	PI	22170	1330	65	4.87 %

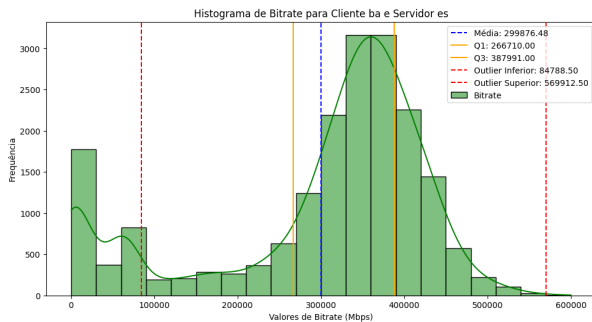
Plotando o histograma da medida de *bitrate* para cada par cliente-servidor foi possível observar que a medida do *bitrate* tem uma distribuição aproximadamente normal, com médias diferentes para cada par. É possível observar também que cada par possui barras fora dessa distribuição normal e com valores mais baixos de *bitrate*, o que pode indicar *outliers*. Os histogramas poder ser observados nas figuras de 9 a 12.



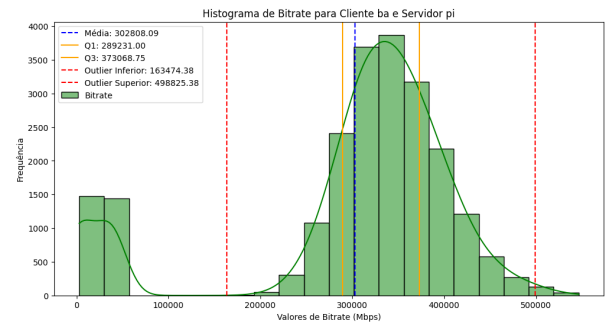
(a) Servidor: CE



(b) Servidor: DF

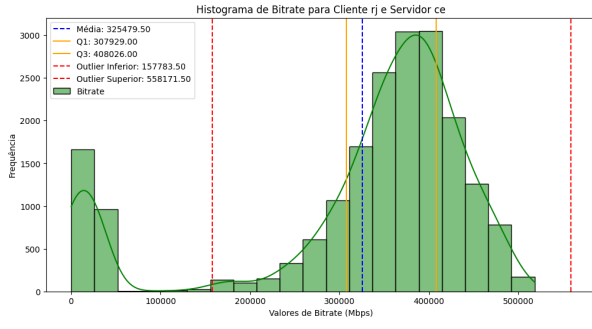
Figura 9: Histograma de distribuição de *bitrate* do cliente BA ao longo de todo o período do *dataset* (parte 1).

(a) Servidor: ES

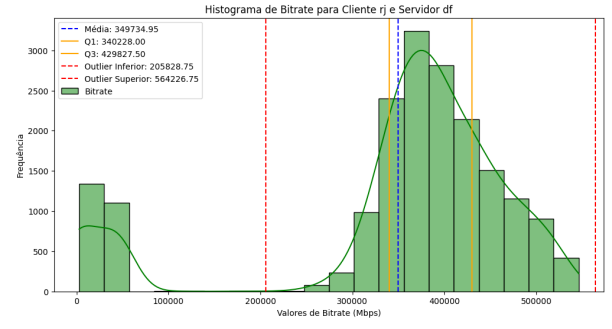


(b) Servidor: PI

Figura 10: Histograma de distribuição de *bitrate* do cliente BA ao longo de todo o período do *dataset* (parte 2).

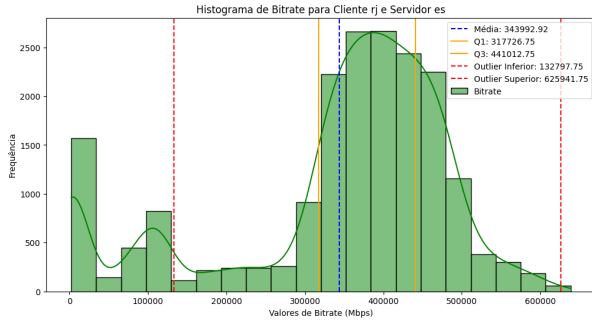


(a) Servidor: CE

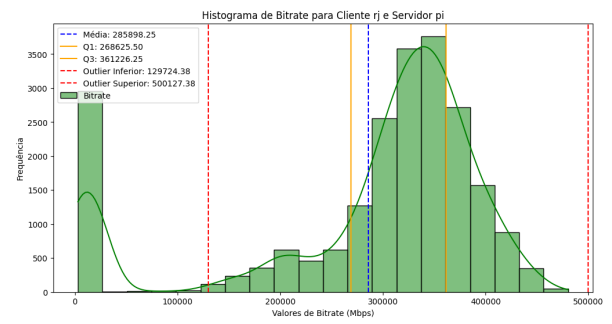


(b) Servidor: DF

Figura 11: Histograma de distribuição de *bitrate* do cliente RJ ao longo de todo o período do *dataset* (parte 1).



(a) Servidor: ES



(b) Servidor: PI

Figura 12: Histograma de distribuição de *bitrate* do cliente RJ ao longo de todo o período do *dataset* (parte 2).

Para construir a tabela 2, foram classificadas como *outliers* as medições de *bitrate* que ficaram acima de $Q3 + 1,5IQR$ ou abaixo de $Q1 - 1,5IQR$, em que *IQR*(distância interquartil) é a distância entre o primeiro ($Q1$) e o terceiro quartil ($Q3$).

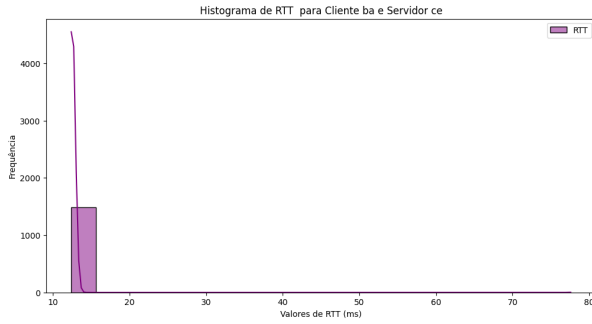
Tabela 2: Distribuição de *bitrate* e Identificação de *outliers* por Cliente-Servidor

Cliente	Servidor	Média <i>bitrate</i>	Variância <i>bitrate</i>	% Outliers Acima	% Outliers Abaixo
RJ	PI	285898	1.47602e+10	0.00	13.65
BA	ES	299876	1.82216e+10	0.04	15.23
BA	PI	302808	1.47749e+10	0.56	13.33
RJ	CE	325479	1.82443e+10	0.00	13.61
BA	DF	330314	1.72322e+10	0.20	13.38
BA	CE	335988	1.7437e+10	0.34	13.35
RJ	ES	343993	2.1588e+10	0.07	15.52
RJ	DF	349735	1.93355e+10	0.00	13.38

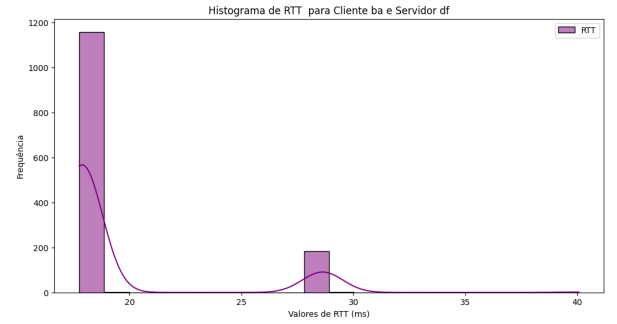
A partir da análise conjunta dos gráficos e da tabela acima, conclui-se que, em geral, o servidor DF tende a apresentar as melhores médias, independentemente do cliente, enquanto o servidor PI

exibe as piores médias, também sem influência do cliente. O cliente RJ, exceto quando conectado ao servidor PI, registra as maiores médias de *bitrate*, sugerindo que o desempenho inferior do servidor PI compromete significativamente o *bitrate* na interação servidor-cliente. Isso indica que o servidor PI é um fator crítico que impacta negativamente a eficiência da comunicação.

De forma análoga, também é possível analisar o histograma da medida de *rtt* para cada par cliente-servidor, verificando-se que os valores de *rtt*, apesar de estarem, em sua maior parte, concentrados em uma única barra, possuem médias diferentes para cada par. Os histogramas podem ser observados nas figuras de 13 a 16.

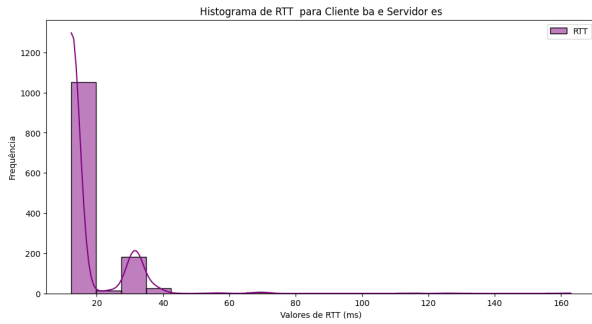


(a) Servidor: CE

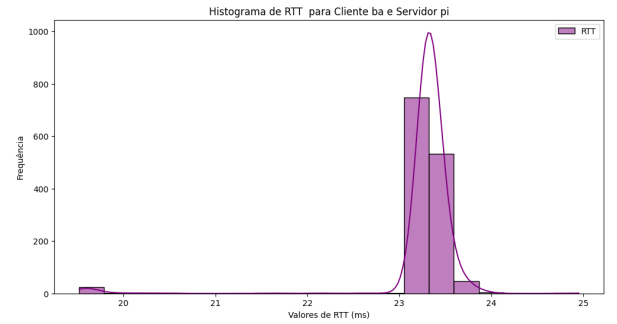


(b) Servidor: DF

Figura 13: Histograma de distribuição de rtt do cliente RJ ao longo de todo o período do *dataset* (parte 1).

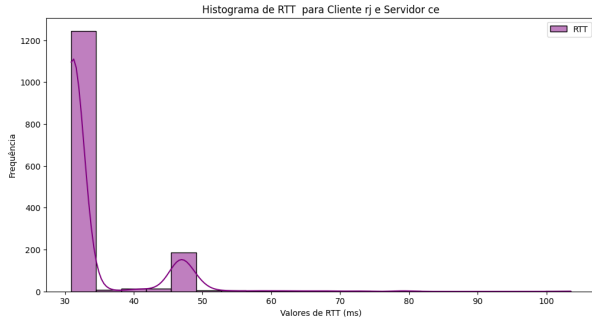


(a) Servidor: ES

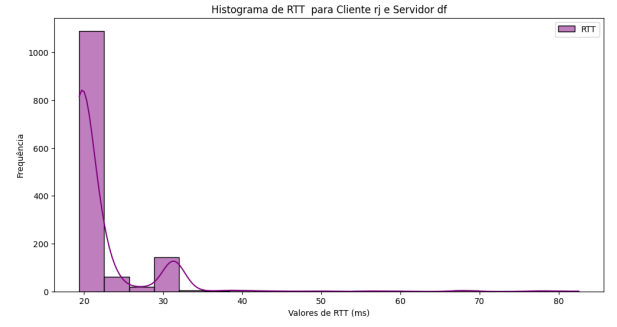


(b) Servidor: PI

Figura 14: Histograma de distribuição de rtt do cliente BA ao longo de todo o período do *dataset* (parte 2).

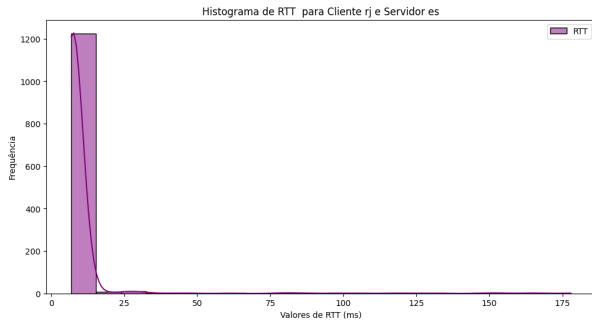


(a) Servidor: CE

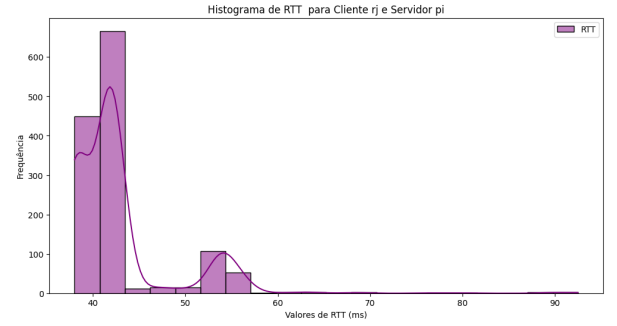


(b) Servidor: DF

Figura 15: Histograma de distribuição de rtt do cliente RJ ao longo de todo o período do *dataset* (parte 1).



(a) Servidor: ES



(b) Servidor: PI

Figura 16: Histograma de distribuição de rtt do cliente RJ ao longo de todo o período do *dataset* (parte 2).

A tabela 3 exibe a contagem de medições de *rtt* para cada par cliente-servidor.

Tabela 3: Contagem de RTT por Cliente-Servidor

Cliente	Servidor	Contagem de RTT
BA	CE	1492
BA	DF	1347
BA	ES	1283
BA	PI	1361
RJ	CE	1487
RJ	DF	1339
RJ	ES	1262
RJ	PI	1330

Conclusão Parcial:

A análise das tabelas e gráficos desta seção revela que a relação entre o cliente RJ e o servidor PI apresentou a maior quantidade de medições em um nível elevado de latência, com a faixa de 35 a

45 ms persistindo por mais de 1100 medições. Além disso, esse par registrou a pior média de *bitrate* entre todos os pares, com 285898 Mbps. Em contraste, a relação entre o cliente RJ e o servidor DF apresentou a maior média de *bitrate* (349735 Mbps) e uma baixa incidência de *outliers* para valores baixos, além de concentrar latências elevadas em um nível relativamente baixo, variando entre 18 e 23 ms, comparado aos demais pares servidor-cliente.

4 Implementação

4.1 Estrutura do Sistema

O código principal utiliza o conceito de *Chain of Thoughts* (Cadeia de Pensamentos) em conjunto com a API da *OpenAI* e o framework *LangChain* para resolver perguntas complexas sobre Qualidade de Experiência (QoE) em uma rede de transmissão de vídeo. O *Chain of Thoughts* permite que o sistema divida o raciocínio em várias etapas lógicas, processando as informações de forma estruturada. A API da *OpenAI* é utilizada para interpretar a pergunta e gerar respostas em linguagem natural, enquanto o *LangChain* organiza o fluxo de processamento, conectando cada etapa de forma coerente para que a resposta seja precisa e relevante. O fluxograma pode ser visualizado na figura abaixo:

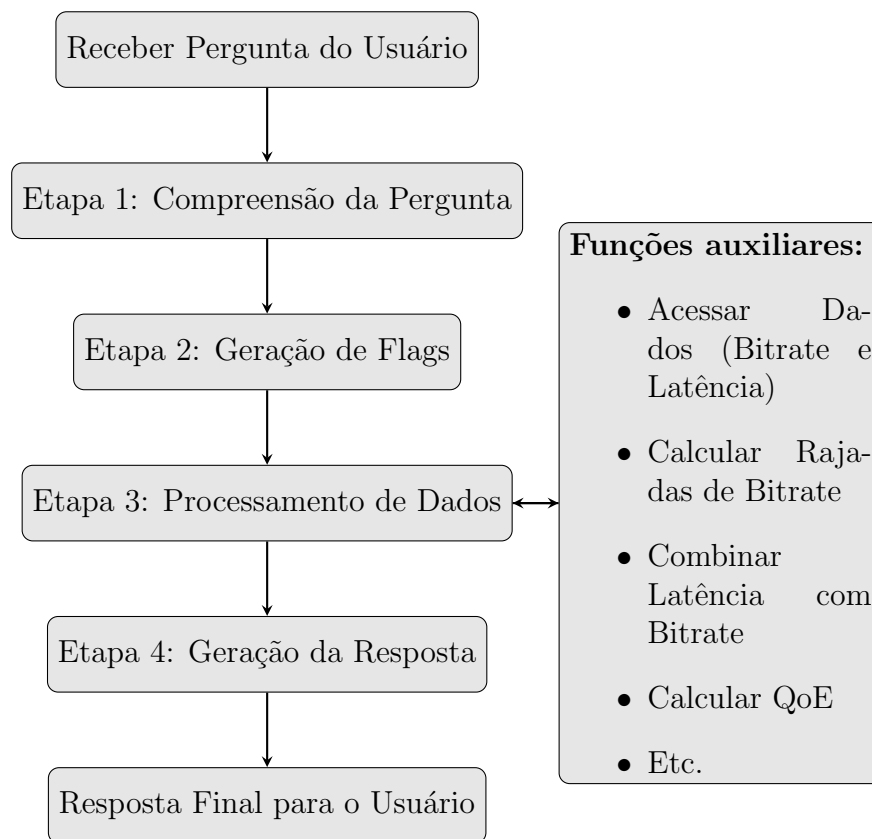


Figura 17: Diagrama de Blocos do Funcionamento do Sistema

A seguir são descritas de forma detalhada cada etapa do processo:

- **Primeira Etapa - Compreensão da Pergunta:** Nesta etapa, o sistema utiliza uma instância do modelo GPT-4o-mini para interpretar a pergunta do usuário em linguagem natural. O modelo é chamado com um *prompt* específico para decompor a pergunta em passos lógicos que guiam o processamento dos dados no contexto do banco de dados disponível. Essa instância do modelo foca em analisar a estrutura da pergunta e definir quais informações precisam ser extraídas ou calculadas. O uso do modelo aqui pode ser considerado como uma instância de **compreensão de linguagem natural**.

- **Segunda Etapa - Geração de Flags:** Nesta etapa, o *GPT-4o-mini* é utilizado novamente, mas agora com um *prompt* orientado para analisar os passos lógicos gerados no *step* anterior. O modelo converte esses passos em parâmetros e *flags* que serão usados para guiar o processamento dos dados. Cada *flag* indica uma ação específica a ser tomada, como filtrar por um período de tempo ou calcular uma métrica. Essa é uma nova instância do mesmo modelo, atuando como um **modelo de parametrização e extração de informações**. Entre as *flags* disponíveis estão:

- **unrelated_to_db:** Indica se a pergunta não está relacionada aos dados do banco de dados.
- **bitrate_bursts:** Ativada quando a pergunta envolve o cálculo da média do bitrate dentro de rajadas de medições.
- **latency_match:** Usada quando a pergunta requer a latência que coincide com as rajadas de bitrate.
- **worst_qoe_client:** Sinaliza que o sistema deve identificar qual cliente teve a pior QoE.
- **server_qoe_consistency:** Indica a necessidade de verificar qual servidor tem a QoE mais consistente.
- **server_change_strategy:** Ativada para determinar a melhor estratégia de troca de servidor para otimizar a QoE.
- **qoe_change:** Ativada para simular como mudanças de bitrate ou latência afetariam a QoE.

Além dessas, há parâmetros como `client`, `server`, `datahora_inicio` e `datahora_final`, que filtram os dados de acordo com a pergunta.

- **Terceira Etapa - Processamento dos Dados:** A partir das *flags* geradas, o sistema consulta o banco de dados e realiza os cálculos necessários. Um dos principais desafios nesse processo é que o **bitrate** é medido em rajadas, enquanto as medições de **latência** são feitas de forma contínua, mas não sincronizadas com essas rajadas. Para calcular a Qualidade de Experiência (QoE), o sistema primeiro agrupa as medições de bitrate em rajadas, calculando a média do bitrate para cada uma delas. Isso é feito criando um **dataframe** com a média do bitrate dentro de um intervalo de até 5 segundos entre as medições, agrupando os dados que estão temporalmente próximos.

Em seguida, o sistema precisa associar esses dados de bitrate com as medições de latência, que não ocorrem nos mesmos instantes que as rajadas de bitrate. Para resolver isso, o sistema cria uma segunda coluna no dataframe que contém a **média da latência** calculada em uma janela de 5 minutos centrada no **timestamp** de cada rajada de bitrate. Essa abordagem permite aproximar as medições de latência de forma a coincidir com os períodos das rajadas de bitrate, possibilitando o cálculo preciso da QoE.

- **Quarta Etapa - Geração da Resposta em Linguagem Natural:** Na última etapa, o sistema usa uma terceira instância do *GPT-4o-mini*, desta vez para gerar uma resposta em linguagem natural baseada nos resultados do processamento de dados. O modelo é orientado a construir uma resposta clara e compreensível para o usuário, utilizando os dados

processados como entrada. O *prompt* utilizado nesta instância orienta o modelo a focar na geração de texto coerente e relevante. Este uso do modelo é classificado como uma instância de **geração de texto**. Por exemplo, a resposta pode informar qual cliente teve a pior qualidade de recepção de vídeo ou como uma mudança na latência pode afetar a QoE.

4.2 Funções auxiliares

As funções auxiliares abaixo são utilizadas na terceira etapa do código principal para realizar tarefas específicas de processamento e manipulação dos dados extraídos do banco de dados. Elas facilitam o cálculo das rajadas de bitrate, a correspondência com as medições de latência e a normalização dos dados para o cálculo da Qualidade de Experiência (QoE), além de outras operações importantes para garantir que os dados sejam corretamente preparados e organizados antes de gerar a resposta final.

- **Função `salvar_dataframes_em_txt` (utilizada apenas para *debug*):** Esta função salva uma lista de `DataFrames` em sequência em um arquivo de texto. Ela é útil para armazenar e visualizar os resultados de diferentes etapas do processamento de dados, como as medições filtradas ou calculadas na etapa 3.
- **Função `salvar_variaveis_em_txt` (utilizada apenas para *debug*):** Similar à função anterior, esta função salva uma lista de variáveis (que podem ser números ou strings) em um arquivo de texto, permitindo registrar e consultar os valores calculados ao longo do processamento.
- **Função `aux_get_dataframes_from_db`:** Esta função é responsável por extrair os dados das tabelas `bitrate_train` e `rtt_train` do banco de dados `trabalho_raw.db` e convertê-los em `DataFrames` para serem manipulados durante o processamento.
- **Função `aux_convert_datahora_to_timestamp`:** Converte *strings* de data e hora no formato `YYYY-MM-DD HH:MM:SS` para o formato `timestamp` Unix, que é usado para filtrar e organizar os dados com base no tempo.
- **Função `aux_convert_timestamp_to_datahora`:** Realiza o processo inverso da função anterior, convertendo `timestamps` Unix de volta para *strings* de data e hora no formato legível, utilizado para exibir os resultados.
- **Função `aux_filter_by_time`:** Filtra as medições dentro de um intervalo de tempo específico, permitindo que o processamento dos dados se limite ao período relevante, com base em datas e horas fornecidas pelo usuário.
- **Função `aux_calculate_bitrate_bursts`:** Agrupa medições de bitrate que ocorrem em rajadas, separadas por no máximo 5 segundos. Em seguida, calcula o bitrate médio para cada rajada, permitindo identificar padrões de transmissão ao longo do tempo para cada cliente e servidor.
- **Função `aux_find_latency_for_bursts`:** Associa medições de latência (`rtt`) com rajadas de bitrate. Para cada rajada de bitrate, ela busca os dados de latência que estão dentro de uma janela de tempo em torno da rajada, e calcula a latência média correspondente.

- **Função `aux_adicionar_normalizacao`:** Normaliza os valores de bitrate e latência para garantir que os dados sejam comparáveis, permitindo que a QoE seja calculada de forma mais precisa, independentemente de variações extremas nos valores.
- **Função `aux_calcular_qoe`:** Calcula a Qualidade de Experiência (QoE) com base nos valores normalizados de bitrate e latência. A QoE é um indicador crucial para medir a qualidade de recepção de vídeo para um cliente.
- **Função `aux_simular_qoe_com_aumento_latencia`:** Simula como a QoE de um cliente seria afetada se a latência (rtt) aumentasse em um determinado percentual, permitindo prever o impacto de variações nas condições de rede.

4.3 Interface

Para que o usuário pudesse interagir com o sistema, foi implementada uma interface *web* simples através de uma aplicação Flask (figura 18). A interface facilita o uso do modelo GPT-4o-mini, permitindo que os usuários façam perguntas relacionadas à Qualidade de Experiência (QoE) e obtenham respostas em linguagem natural.

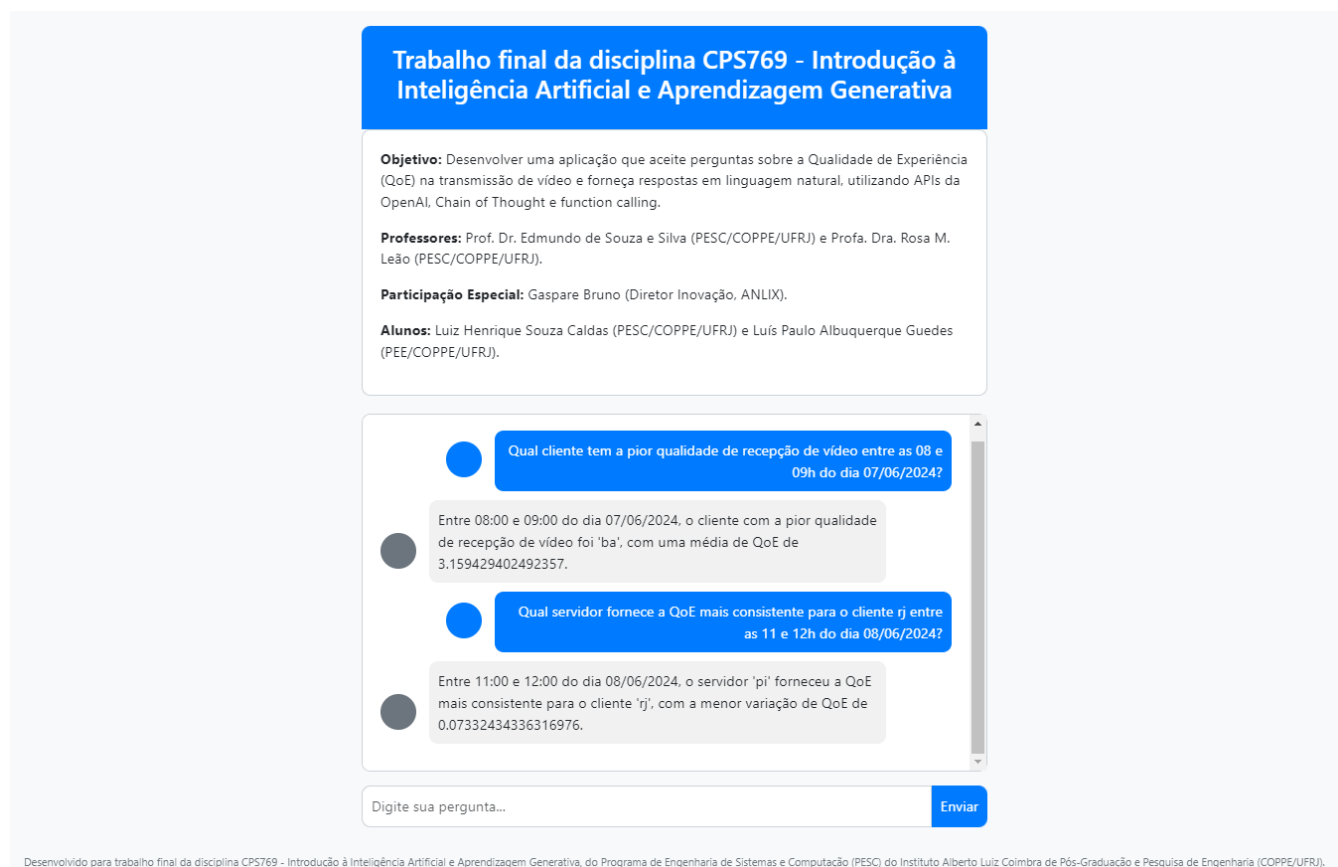


Figura 18: Interface para o usuário interagir com o sistema

4.3.1 Estrutura da Interface

- **app.py (Back-end)**: O arquivo `app.py` implementa o servidor Flask, que serve a página *web* e processa as interações do usuário. O servidor possui duas rotas principais:
 - `/`: Carrega a página HTML que exibe a interface do chatbot.
 - `/chat`: Recebe as perguntas enviadas pelo usuário e utiliza a função `responder_pergunta` (definida no arquivo `llm_model`) para gerar a resposta. A resposta é enviada de volta para a interface no formato JSON.

Esse *back-end* faz a ponte entre a interface do usuário e o modelo de linguagem, processando as perguntas e retornando as respostas.

- **index.html (Front-end)**: O arquivo `index.html` define a interface gráfica do *chatbot*. Ele usa HTML, Bootstrap para o design responsivo, e JavaScript para controlar o envio de mensagens e a exibição das respostas. A página exibe uma caixa de *chat* onde o usuário digita sua pergunta e visualiza as respostas. Há também uma área com um resumo explicando o objetivo da aplicação, o curso, e os envolvidos no projeto.
- **chatbot.js (JavaScript)**: O JavaScript controla a lógica do envio de perguntas e recebimento de respostas. A função `sendMessage()` captura o texto que o usuário digitou, exibe a pergunta na interface e envia a mensagem para o servidor Flask usando uma requisição POST para a rota `/chat`. Assim que a resposta é recebida do servidor, a função `appendMessage()` exibe a resposta do chatbot na interface.

4.3.2 Funcionamento da Interface

Quando o usuário acessa a página, ele vê uma interface simples de *chatbot*, onde pode digitar perguntas. O sistema Flask no *back-end* processa as perguntas utilizando a LLM *GPT-4o-mini*. O JavaScript no *front-end* é responsável por capturar as interações do usuário e exibir as respostas dinamicamente na página.

Fluxo de Operação:

1. O usuário digita uma pergunta na caixa de entrada e clica no botão “Enviar”.
2. A pergunta é enviada para o back-end Flask através de uma requisição POST.
3. No back-end, a pergunta é processada pela função `responder_pergunta`, que utiliza a LLM para gerar uma resposta.
4. A resposta é enviada de volta para o *front-end*, onde é exibida na caixa de *chat*.

Essa interface facilita o uso da LLM de maneira intuitiva, permitindo que qualquer usuário faça perguntas e receba respostas de forma interativa, sem precisar entender como o modelo funciona nos bastidores.

4.4 Código

Todos os códigos utilizados neste trabalho encontram-se neste repositório.

5 Resultados

Para apresentar os resultados, foi implementada uma função *debug_step-by-step*, que executa todos os passos da *Chain of Thoughts* implementada e printa, além da resposta final, os resultados intermediários. Foram realizados testes com as 8 perguntas abaixo:

Nota: todas consideram o mesmo período para facilitar o debug.

- Qual o bitrate médio dentro de cada rajada para o cliente rj e o servidor pi no período entre as 08 e 09h do dia 07/06/2024?

Passos lógicos gerados:

Step 1: 1. Filter 'bitrate_train' for client rj and server pi within the time period from 08:00 to 09:00 on 2024-06-07.

Step 2: 2. Group measurements into bursts (timestamps within 5 seconds of each other).

Step 3: 3. Calculate the average bitrate for each burst.

Flags geradas:

```
unrelated_to_db: False
bitrate_bursts: True
latency_match: False
worst_qoe_client: False
server_qoe_consistency: False
server_change_strategy: False
qoe_change: False
datahora_inicio: 2024-06-07 08:00:00
datahora_final: 2024-06-07 09:00:00
client: rj
server: pi
bitrate_delta: 0
latency_delta: 0
```

Resultado do processamento:

question: Qual o bitrate médio dentro de cada rajada para o cliente rj e o servidor pi no período entre as 08 e 09h do dia 07/06/2024?

client: rj

server: pi

```
bitrate_bursts: {"datahora_inicio":{"0":{"0\":"2024-06-07 08:03:05\","1\":"2024-06-07 08:08:05\","2\":"2024-06-07 08:13:05\","3\":"2024-06-07 08:18:05\","4\":"2024-06-07 08:23:05\","5\":"2024-06-07 08:28:05\","6\":"2024-06-07 08:33:05\","7\":"2024-06-07 08:38:05\","8\":"2024-06-07 08:43:06\","9\":"2024-06-07 08:48:06\","10\":"2024-06-07 08:53:06\","11\":"2024-06-07 08:58:06\"},"1":{"0\":"2024-06-07 08:03:05\","1\":"2024-06-07 08:08:05\","2\":"
```

[illegible]

[illegible]

```
:\"2024-06-07 08:38:08\", \"8\": \"2024-06-07 08:43:08\", \"9\": \"2024-06-07 08:48:08\", \"10\": \"2024-06-07 08:53:08\", \"11\": \"2024-06-07 08:58:08\"} \", \"9\": \"{\\"0\": \"2024-06-07 08:03:08\", \"1\": \"2024-06-07 08:08:07\", \"2\": \"2024-06-07 08:13:08\", \"3\": \"2024-06-07 08:18:08\", \"4\": \"2024-06-07 08:23:08\", \"5\": \"2024-06-07 08:28:08\", \"6\": \"2024-06-07 08:33:08\", \"7\": \"2024-06-07 08:38:08\", \"8\": \"2024-06-07 08:43:08\", \"9\": \"2024-06-07 08:48:08\", \"10\": \"2024-06-07 08:53:08\", \"11\": \"2024-06-07 08:58:08\"} \", \"10\": \"{\\"0\": \"2024-06-07 08:03:08\", \"1\": \"2024-06-07 08:08:07\", \"2\": \"2024-06-07 08:13:08\", \"3\": \"2024-06-07 08:18:08\", \"4\": \"2024-06-07 08:23:08\", \"5\": \"2024-06-07 08:28:08\", \"6\": \"2024-06-07 08:33:08\", \"7\": \"2024-06-07 08:38:08\", \"8\": \"2024-06-07 08:43:08\", \"9\": \"2024-06-07 08:48:08\", \"10\": \"2024-06-07 08:53:08\", \"11\": \"2024-06-07 08:58:08\"} \", \"11\": \"{\\"0\": \"2024-06-07 08:03:08\", \"1\": \"2024-06-07 08:08:07\", \"2\": \"2024-06-07 08:13:08\", \"3\": \"2024-06-07 08:18:08\", \"4\": \"2024-06-07 08:23:08\", \"5\": \"2024-06-07 08:28:08\", \"6\": \"2024-06-07 08:33:08\", \"7\": \"2024-06-07 08:38:08\", \"8\": \"2024-06-07 08:43:08\", \"9\": \"2024-06-07 08:48:08\", \"10\": \"2024-06-07 08:53:08\", \"11\": \"2024-06-07 08:58:08\"} \", \"bitrate_medio\": {\"0\": 281896.6666666667, \"1\": 318607.0666666667, \"2\": 271408.0, \"3\": 306599.3333333333, \"4\": 305519.5333333333, \"5\": 284570.7333333333, \"6\": 282189.4666666667, \"7\": 276509.8, \"8\": 319331.8, \"9\": 279828.5333333333, \"10\": 277090.2, \"11\": 289222.7333333333}}
```

Resposta gerada:

Entre 08:00 e 09:00 do dia 07/06/2024, o bitrate médio dentro de cada rajada para o cliente 'rj' e o servidor 'pi' foi:

- 2024-06-07 08:03:05: 281896.67 bps
- 2024-06-07 08:08:05: 318607.07 bps
- 2024-06-07 08:13:05: 271408.00 bps
- 2024-06-07 08:18:05: 306599.33 bps
- 2024-06-07 08:23:05: 305519.53 bps
- 2024-06-07 08:28:05: 284570.73 bps
- 2024-06-07 08:33:05: 282189.47 bps
- 2024-06-07 08:38:05: 276509.80 bps
- 2024-06-07 08:43:06: 319331.80 bps
- 2024-06-07 08:48:06: 279828.53 bps
- 2024-06-07 08:53:06: 277090.20 bps
- 2024-06-07 08:58:06: 289222.73 bps.

- Qual a latência nas medições que coincidem com a janela de tempo das rajadas de medição de bitrate para o cliente rj e o servidor pi no período entre as 08 e 09h do dia 07/06/2024?

Passos lógicos gerados:

Step 1: 1. Filtrar a tabela 'bitrate_train' para o cliente rj e o servidor pi dentro do período entre as 08h e 09h do dia 07/06/2024.

Step 2: 2. Agrupar as medições de bitrate em rajadas (timestamps dentro de 5 segundos um do outro).

Step 3: 3. Para cada rajada identificada, determinar a janela de tempo correspondente (5 segundos antes e 5 segundos depois do timestamp da rajada).

Step 4: 4. Filtrar a tabela 'rtt_train' para encontrar medições de latência que ocorram dentro dessas janelas de tempo das rajadas de bitrate.

Step 5: 5. Retornar os valores de latência que coincidem com as rajadas de bitrate.

Flags geradas:

```
unrelated_to_db: False
bitrate_bursts: False
latency_match: True
worst_qoe_client: False
server_qoe_consistency: False
server_change_strategy: False
qoe_change: False
datahora_inicio: 2024-06-07 08:00:00
datahora_final: 2024-06-07 09:00:00
client: rj
server: pi
bitrate_delta: 0
latency_delta: 0
```

Resultado do processamento:

question: Qual a latência nas medições que coincidem com a janela de tempo das rajadas de medição de bitrate para o cliente rj e o servidor pi no período entre as 08 e 09h do dia 07/06/2024?

client: rj

server: pi

```
latency: {"datahora":{"0":{"0":"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"1\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"2\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"3\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"4\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"5\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"6\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"7\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}},\"8\":{\"0\":\"2024-06-07 08:03:06\\",\"1\":\"2024-06-07 08:08:06\\",\"2\":\"2024-06-07 08:13:06\\",\"3\":\"2024-06-07 08:18:06\\",\"4\":\"2024-06-07 08:28:06\\",\"5\":\"2024-06-07 08:38:06\\",\"6\":\"2024-06-07 08:43:07\\",\"7\":\"2024-06-07 08:48:07\\",\"8\":\"2024-06-07 08:53:07\\\"}}}
```

```
6\","\6\":"2024-06-07 08:43:07\","\7\":"2024-06-07 08:48:07\","\8\":"2024-06-07 08:53:07\"}", "4":{"\0\":"2024-06-07 08:03:06\","\1\":"2024-06-07 08:08:06\","\2\":"2024-06-07 08:13:06\","\3\":"2024-06-07 08:18:06\","\4\":"2024-06-07 08:28:06\","\5\":"2024-06-07 08:38:06\","\6\":"2024-06-07 08:43:07\","\7\":"2024-06-07 08:48:07\","\8\":"2024-06-07 08:53:07\"}", "5":{"\0\":"2024-06-07 08:03:06\","\1\":"2024-06-07 08:08:06\","\2\":"2024-06-07 08:13:06\","\3\":"2024-06-07 08:18:06\","\4\":"2024-06-07 08:28:06\","\5\":"2024-06-07 08:38:06\","\6\":"2024-06-07 08:43:07\","\7\":"2024-06-07 08:48:07\","\8\":"2024-06-07 08:53:07\"}", "6":{"\0\":"2024-06-07 08:03:06\","\1\":"2024-06-07 08:08:06\","\2\":"2024-06-07 08:13:06\","\3\":"2024-06-07 08:18:06\","\4\":"2024-06-07 08:28:06\","\5\":"2024-06-07 08:38:06\","\6\":"2024-06-07 08:43:07\","\7\":"2024-06-07 08:48:07\","\8\":"2024-06-07 08:53:07\"}", "7":{"\0\":"2024-06-07 08:03:06\","\1\":"2024-06-07 08:08:06\","\2\":"2024-06-07 08:13:06\","\3\":"2024-06-07 08:18:06\","\4\":"2024-06-07 08:28:06\","\5\":"2024-06-07 08:38:06\","\6\":"2024-06-07 08:43:07\","\7\":"2024-06-07 08:48:07\","\8\":"2024-06-07 08:53:07\"}", "8":{"\0\":"2024-06-07 08:03:06\","\1\":"2024-06-07 08:08:06\","\2\":"2024-06-07 08:13:06\","\3\":"2024-06-07 08:18:06\","\4\":"2024-06-07 08:28:06\","\5\":"2024-06-07 08:38:06\","\6\":"2024-06-07 08:43:07\","\7\":"2024-06-07 08:48:07\","\8\":"2024-06-07 08:53:07\"}"}, "bitrate":{"0":281896.66666666667, "1":318607.06666666667, "2":271408.0, "3":306599.33333333333, "4":284570.73333333333, "5":276509.8, "6":319331.8, "7":279828.53333333333, "8":277090.2}, "rtt":{"0":41.84, "1":41.86, "2":41.83, "3":41.85, "4":38.1, "5":38.24, "6":38.2, "7":38.22, "8":41.87}}
```

Resposta gerada:

Entre 08:00 e 09:00 do dia 07/06/2024, as latências que coincidiram com as rajadas de bitrate para o cliente 'rj' e o servidor 'pi' foram as seguintes:

- 2024-06-07 08:03:06: 41.84 ms
- 2024-06-07 08:08:06: 41.86 ms
- 2024-06-07 08:13:06: 41.83 ms
- 2024-06-07 08:18:06: 41.85 ms
- 2024-06-07 08:28:06: 38.10 ms
- 2024-06-07 08:38:06: 38.24 ms
- 2024-06-07 08:43:07: 38.20 ms
- 2024-06-07 08:48:07: 38.22 ms
- 2024-06-07 08:53:07: 41.87 ms.

- Qual cliente tem a pior qualidade de recepção de vídeo entre as 08 e 09h do dia 07/06/2024?

Passos lógicos gerados:

Step 1: 1. Filter 'bitrate_train' and 'rtt_train' for all clients within the time period of 08:00 to 09:00 on 07/06/2024.

Step 2: 2. Group bitrate measurements into bursts (timestamps within 5 seconds of

f each other).

Step 3: 3. Ensure that QoE is only calculated when latency from 'rtt_train' matches a burst of bitrate from 'bitrate_train'.

Step 4: 4. Calculate QoE for each client based on the matched bitrate and latency values.

Step 5: 5. Identify the client with the lowest average QoE during this time period.

Flags geradas:

```
unrelated_to_db: False
bitrate_bursts: False
latency_match: False
worst_qoe_client: True
server_qoe_consistency: False
server_change_strategy: False
qoe_change: False
datahora_inicio: 2024-06-07 08:00:00
datahora_final: 2024-06-07 09:00:00
client:
server:
bitrate_delta: 0
latency_delta: 0
```

Resultado do processamento:

```
question: Qual cliente tem a pior qualidade de recepção de vídeo entre as 08 e
09h do dia 07/06/2024?
worst_client: ba
worst_client_mean_qoe: 3.159429402492357
```

Resposta gerada:

Entre 08:00 e 09:00 do dia 07/06/2024, o cliente com a pior qualidade de recepção de vídeo foi 'ba', com uma média de QoE de 3.159429402492357.

- Qual servidor fornece a QoE mais consistente para o cliente rj entre as 08 e 09h do dia 07/06/2024?

Passos lógicos gerados:

Step 1: 1. Filter 'bitrate_train' for client rj and the relevant servers within the time period of 08:00 to 09:00 on 07/06/2024.

Step 2: 2. Filter 'rtt_train' for client rj within the same time period.

Step 3: 3. Ensure that QoE is only calculated when latency from 'rtt_train' matches a burst of bitrate from 'bitrate_train'.

Step 4: 4. Calculate QoE for each server that client rj is connected to during this time frame.

Step 5: 5. Calculate the variance of QoE for each server.

Step 6: 6. Identify the server with the lowest variance, indicating the most consistent QoE.

Flags geradas:

```
unrelated_to_db: False
bitrate_bursts: False
latency_match: False
worst_qoe_client: False
server_qoe_consistency: True
server_change_strategy: False
qoe_change: False
datahora_inicio: 2024-06-07 08:00:00
datahora_final: 2024-06-07 09:00:00
client: rj
server:
bitrate_delta: 0
latency_delta: 0
```

Resultado do processamento:

```
question: Qual servidor fornece a QoE mais consistente para o cliente rj entre
as 08 e 09h do dia 07/06/2024?
qoe_variance: {'ce': 0.03180962402563327, 'df': 0.028072240357539146, 'es': 10
67459804.530213, 'pi': 0.013866611458790644}
best_qoe_variance: {'pi', 0.013866611458790644}
client: rj
```

Resposta gerada:

Entre 08:00 e 09:00 do dia 07/06/2024, o servidor 'pi' forneceu a QoE mais consistente para o cliente 'rj', com a menor variação de QoE de 0.013866611458790644.

- Qual é a melhor estratégia de troca de servidor para maximizar a qualidade de experiência do cliente rj entre as 08 e 09h do dia 07/06/2024?

Passos lógicos gerados:

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]


```

:{"2024-06-07 08:48:07\\",\\"26\\":\\"2024-06-07 08:49:06\\",\\"27\\":\\"2024-06-07 08:52:36\\",\\"28\\":\\"2024-06-07 08:53:07\\",\\"29\\":\\"2024-06-07 08:53:37\\",\\"30\\":\\"2024-06-07 08:57:37\\",\\"31\\":\\"2024-06-07 08:59:06\\"}","30":{"\\"0\\":\\"2024-06-07 08:02:36\\",\\"1\\":\\"2024-06-07 08:03:06\\",\\"2\\":\\"2024-06-07 08:03:36\\",\\"3\\":\\"2024-06-07 08:08:06\\",\\"4\\":\\"2024-06-07 08:09:06\\",\\"5\\":\\"2024-06-07 08:12:36\\",\\"6\\":\\"2024-06-07 08:13:06\\",\\"7\\":\\"2024-06-07 08:13:36\\",\\"8\\":\\"2024-06-07 08:14:06\\",\\"9\\":\\"2024-06-07 08:18:06\\",\\"10\\":\\"2024-06-07 08:23:36\\",\\"11\\":\\"2024-06-07 08:24:06\\",\\"12\\":\\"2024-06-07 08:27:36\\",\\"13\\":\\"2024-06-07 08:28:06\\",\\"14\\":\\"2024-06-07 08:28:36\\",\\"15\\":\\"2024-06-07 08:32:36\\",\\"16\\":\\"2024-06-07 08:33:36\\",\\"17\\":\\"2024-06-07 08:34:06\\",\\"18\\":\\"2024-06-07 08:37:36\\",\\"19\\":\\"2024-06-07 08:38:06\\",\\"20\\":\\"2024-06-07 08:38:36\\",\\"21\\":\\"2024-06-07 08:39:07\\",\\"22\\":\\"2024-06-07 08:43:07\\",\\"23\\":\\"2024-06-07 08:43:37\\",\\"24\\":\\"2024-06-07 08:47:36\\",\\"25\\":\\"2024-06-07 08:48:07\\",\\"26\\":\\"2024-06-07 08:49:06\\",\\"27\\":\\"2024-06-07 08:52:36\\",\\"28\\":\\"2024-06-07 08:53:07\\",\\"29\\":\\"2024-06-07 08:53:37\\",\\"30\\":\\"2024-06-07 08:57:37\\",\\"31\\":\\"2024-06-07 08:59:06\\"}"},"31":{"\\"0\\":\\"2024-06-07 08:02:36\\",\\"1\\":\\"2024-06-07 08:03:06\\",\\"2\\":\\"2024-06-07 08:03:36\\",\\"3\\":\\"2024-06-07 08:08:06\\",\\"4\\":\\"2024-06-07 08:09:06\\",\\"5\\":\\"2024-06-07 08:12:36\\",\\"6\\":\\"2024-06-07 08:13:06\\",\\"7\\":\\"2024-06-07 08:13:36\\",\\"8\\":\\"2024-06-07 08:14:06\\",\\"9\\":\\"2024-06-07 08:18:06\\",\\"10\\":\\"2024-06-07 08:23:36\\",\\"11\\":\\"2024-06-07 08:24:06\\",\\"12\\":\\"2024-06-07 08:27:36\\",\\"13\\":\\"2024-06-07 08:28:06\\",\\"14\\":\\"2024-06-07 08:28:36\\",\\"15\\":\\"2024-06-07 08:32:36\\",\\"16\\":\\"2024-06-07 08:33:36\\",\\"17\\":\\"2024-06-07 08:34:06\\",\\"18\\":\\"2024-06-07 08:37:36\\",\\"19\\":\\"2024-06-07 08:38:06\\",\\"20\\":\\"2024-06-07 08:38:36\\",\\"21\\":\\"2024-06-07 08:39:07\\",\\"22\\":\\"2024-06-07 08:43:07\\",\\"23\\":\\"2024-06-07 08:43:37\\",\\"24\\":\\"2024-06-07 08:47:36\\",\\"25\\":\\"2024-06-07 08:48:07\\",\\"26\\":\\"2024-06-07 08:49:06\\",\\"27\\":\\"2024-06-07 08:52:36\\",\\"28\\":\\"2024-06-07 08:53:07\\",\\"29\\":\\"2024-06-07 08:53:37\\",\\"30\\":\\"2024-06-07 08:57:37\\",\\"31\\":\\"2024-06-07 08:59:06\\"}"}

```

```

QoE: {"0":1.8722568683,"1":0.4448322638,"2":1.0846052575,"3":0.6571791761,"4":161.743378668,"5":1.849747368,"6":0.3841643022,"7":1.2055086484,"8":237.5829099711,"9":0.5878069003,"10":1.4507685916,"11":941.6779696344,"12":2.1669718582,"13":0.5155963812,"14":1.1930813364,"15":1.8472891664,"16":0.9289224668,"17":87033.3530953859,"18":2.0159258488,"19":0.4612051002,"20":1.0156137038,"21":0.0,"22":0.7388367374,"23":1.2143298267,"24":2.0503723819,"25":0.4829572338,"26":1730.8593343275,"27":2.3166950791,"28":0.4166222792,"29":0.9119155498,"30":2.0952771518,"31":560.5211665899}

```

client: rj

Resposta gerada:

Entre 08:00 e 09:00 do dia 07/06/2024, a melhor estratégia de troca de servidor para maximizar a qualidade de experiência do cliente 'rj' é a seguinte:

- 08:02:36: servidor 'df'
- 08:03:06: servidor 'pi'
- 08:03:36: servidor 'ce'

- 08:08:06: servidor 'pi'
- 08:09:06: servidor 'es'
- 08:12:36: servidor 'df'
- 08:13:06: servidor 'pi'
- 08:13:36: servidor 'ce'
- 08:14:06: servidor 'es'
- 08:18:06: servidor 'pi'
- 08:23:36: servidor 'ce'
- 08:24:06: servidor 'es'
- 08:27:36: servidor 'df'
- 08:28:06: servidor 'pi'
- 08:28:36: servidor 'ce'
- 08:32:36: servidor 'df'
- 08:33:36: servidor 'ce'
- 08:34:06: servidor 'es'
- 08:37:36: servidor 'df'
- 08:38:06: servidor 'pi'
- 08:38:36: servidor 'ce'
- 08:39:07: servidor 'es'
- 08:43:07: servidor 'pi'
- 08:43:37: servidor 'ce'
- 08:47:36: servidor 'df'
- 08:48:07: servidor 'pi'
- 08:49:06: servidor 'es'
- 08:52:36: servidor 'df'
- 08:53:07: servidor 'pi'
- 08:53:37: servidor 'ce'
- 08:57:37: servidor 'df'
- 08:59:06: servidor 'es'.

- Se a latência aumentar 20

Passos lógicos gerados:

Step 1: 1. Filter 'rtt_train' for client rj and server pi within the time period of 08:00 to 09:00 on 2024-06-07 to retrieve current latency values.

Step 2: 2. Increase the retrieved latency values by 20%.

Step 3: 3. Filter 'bitrate_train' for client rj and server pi within the same time period to get the corresponding bitrate data.

Step 4: 4. Match the modified latency values with the bitrate bursts from 'bitrate_train' to calculate the new QoE.

Step 5: 5. Compare the new QoE values with the original QoE values calculated using the unmodified latency values to explain the impact of the latency change.

Flags geradas:

```
unrelated_to_db: False
bitrate_bursts: False
latency_match: False
worst_qoe_client: False
server_qoe_consistency: False
server_change_strategy: False
qoe_change: True
datahora_inicio: 2024-06-07 08:00:00
datahora_final: 2024-06-07 09:00:00
client: rj
server: pi
bitrate_delta: 0
latency_delta: 20
```

Resultado do processamento:

question: Se a latência aumentar 20%, como isso afeta a QoE do cliente rj e servidor pi entre as 08 e 09h do dia 07/06/2024?

QoE: {"0":0.4448322638,"1":0.6571791761,"2":0.3841643022,"3":0.5878069003,"4":0.5155963812,"5":0.4612051002,"6":0.7388367374,"7":0.4829572338,"8":0.4166222792}

new_QoE: {"0":0.4448199839,"1":0.6571731322,"2":0.3841501582,"3":0.5877960855,"4":0.5136006779,"5":0.4594936367,"6":0.7360613912,"7":0.4811540656,"8":0.4166222792}

```
client: rj
server: pi
```

datahora: {"0":"2024-06-07 08:03:06","1":"2024-06-07 08:08:06","2":"2024-06-07 08:13:06","3":"2024-06-07 08:18:06","4":"2024-06-07 08:28:06","5":"2024-06-07 08:38:06","6":"2024-06-07 08:43:07","7":"2024-06-07 08:48:07","8":"2024-06-07 08:53:07"}

Resposta gerada:

Se a latência aumentar 20% entre 08:00 e 09:00 do dia 07/06/2024 para o cliente 'rj' e servidor 'pi', a qualidade de experiência (QoE) mudaria da seguinte maneira:

- 2024-06-07 08:03:06: De 0.4448322638 para 0.4448199839.
- 2024-06-07 08:08:06: De 0.6571791761 para 0.6571731322.
- 2024-06-07 08:13:06: De 0.3841643022 para 0.3841501582.
- 2024-06-07 08:18:06: De 0.5878069003 para 0.5877960855.
- 2024-06-07 08:28:06: De 0.5155963812 para 0.5136006779.
- 2024-06-07 08:38:06: De 0.4612051002 para 0.4594936367.
- 2024-06-07 08:43:07: De 0.7388367374 para 0.7360613912.
- 2024-06-07 08:48:07: De 0.4829572338 para 0.4811540656.

- 2024-06-07 08:53:07: De 0.4166222792 para 0.4166222792.

- Qual o endereço de IP do cliente rj na rede?

Passos lógicos gerados:

Step 1: 1. Recognize that this question is unrelated to the available network measurement data.

Flags geradas:

unrelated_to_db: True
bitrate_bursts: False
latency_match: False
worst_qoe_client: False
server_qoe_consistency: False
server_change_strategy: False
qoe_change: False
datahora_inicio: 2024-06-07 00:00:00
datahora_final: 2024-06-10 23:59:59
client:
server:
bitrate_delta: 0
latency_delta: 0

Resultado do processamento:

question: Qual o endereço de IP do cliente rj na rede?
info: A pergunta não está relacionada ao banco de dados disponível.

Resposta gerada:

Não possuo em meu banco de dados o endereço de IP do cliente rj na rede.
Por favor, entre outra pergunta relacionada à rede ou aos dados disponíveis.

- Qual é a previsão do tempo para amanhã?

Passos lógicos gerados:

Step 1: 1. Reconhecer que esta pergunta é irrelevante para os dados de medições de rede disponíveis no banco de dados.

Flags geradas:

unrelated_to_db: True
bitrate_bursts: False
latency_match: False
worst_qoe_client: False
server_qoe_consistency: False
server_change_strategy: False
qoe_change: False
datahora_inicio: 2024-06-07 00:00:00
datahora_final: 2024-06-10 23:59:59
client:
server:
bitrate_delta: 0
latency_delta: 0

Resultado do processamento:

question: Qual é a previsão do tempo para amanhã?
info: A pergunta não está relacionada ao banco de dados disponível.

Resposta gerada:

Desculpe, mas não tenho informações sobre a previsão do tempo.
Por favor, entre outra pergunta relacionada à rede ou aos dados disponíveis.

6 Conclusão