Relatório do Projeto A3

Sistemas Distribuídos e Mobile

Equipe:

- João Paulo dos Santos Costa (RA: 1272118412)
- Aleide Mascarenhas (RA: 1272116102)
- Gabriel da Silva Jorge Neto (RA: 1272122772)
- Jonathas de Carvalho Pereira (RA: 1272116857)

Requerimentos de Software:

Linguagem de programação: Python (versão mínima 3.8)
Bibliotecas utilizadas: random, socket, threading, sqlite3, json, os, time, datetime, typing

Instruções de Instalação e Execução:

Certifique-se de ter o Python instalado em sua máquina com a versão mínima 3.8.

Para fins de demonstração a aplicação já consta com 3 usuários e 2 lojas previamente cadastradas, o termo de identificação das lojas são **americanas** e **bompreco**.

Nome de usuário	Senha	Cargo
jess	123	gerente
carl	123	vendedor
paul	123	vendedor

Faça o download do código-fonte do projeto para o diretório de sua escolha.

1. Abra uma janela de terminal no diretório raiz do projeto.
2. Inicie o servidor executando o seguinte comando:
   python -m server
3. Abra uma nova janela de terminal no diretório raiz do projeto.
4. Inicie o cliente executando o seguinte comando:
   python -m client
5. No terminal que está executando o cliente, será exibido um prompt para inserir o nome de usuário e a senha.
6. Insira os dados de autenticação e pressione Enter. O sistema verificará no banco de dados se os dados estão corretos.

7. Após a autenticação bem-sucedida, serão exibidos os comandos disponíveis de acordo com o cargo do usuário autenticado.

Uma funcionalidade adicional implementada no servidor é a simulação de falha de conexão com o servidor principal. Ao adicionar a flag "**-f**" ao comando "**python -m server**", o servidor iniciará uma simulação de falha de conexão.

Após a falha de conexão com o servidor principal, o servidor intermediário realizará um número predefinido de tentativas de conexão em intervalos de tempo também pré configurados. Essas tentativas de conexão são realizadas na tentativa de restabelecer a conexão com o servidor principal.

Caso as tentativas de conexão com o servidor principal sejam mal sucedidas dentro do período determinado, o servidor intermediário iniciará a conexão com o servidor temporário. O servidor temporário continuará tentando reconectar-se ao servidor principal durante um período pré-determinado, buscando restabelecer a conexão.

Essa funcionalidade de simulação de falha de conexão e a alternância entre o servidor principal e o servidor temporário permite que a aplicação continue funcionando de maneira adequada mesmo em cenários em que a conexão com o servidor principal não está disponível temporariamente.

Essa funcionalidade adicional aumenta a robustez e a confiabilidade do sistema, garantindo que os clientes possam continuar se conectando e realizando suas operações, mesmo em situações de falha temporária de conexão com o servidor principal.

Arquitetura, Estratégia e Algoritmos Utilizados:

O projeto adota uma arquitetura cliente-servidor. O servidor principal e o intermediário são iniciados, e os clientes se conectam através do intermediário. O intermediário, por sua vez, se conecta ao servidor principal e faz a ligação entre as duas partes. Em caso de falha de conexão com o servidor principal, o intermediário tenta se conectar novamente e, se necessário, se conecta a um servidor temporário.

A estratégia de comunicação é baseada em sockets, utilizando a biblioteca socket do Python. O servidor estabelece conexões com os clientes por meio de sockets e responde às solicitações dos gerentes e vendedores.

O banco de dados utilizado é o SQLite3, que é uma biblioteca embutida no Python. As informações fornecidas pelos vendedores são persistidas em um arquivo de

banco de dados chamado "database" localizado na pasta “src”. Os gerentes podem consultar as informações de vendas registradas no banco de dados.

Os algoritmos utilizados no projeto incluem a autenticação dos usuários, a validação dos dados de autenticação no banco de dados, o gerenciamento das conexões de rede entre o servidor, o intermediário e os clientes, o armazenamento e consulta de informações no banco de dados SQLite3, entre outros.

Para evitar erros, foram utilizados blocos try-except em diferentes partes do código, a fim de tratar exceções e garantir um fluxo de execução correto.

A aplicação também utiliza as bibliotecas random, threading, json, os, time, datetime e typing para funcionalidades como geração de números aleatórios, execução de operações em threads separadas, manipulação de dados JSON, manipulação de arquivos e diretórios, manipulação de datas e tipos de dados, entre outras funcionalidades adicionais necessárias para o funcionamento do projeto.

O projeto foi desenvolvido com o objetivo de permitir que vendedores informem ao sistema o valor de cada venda feita por eles, enquanto os gerentes possam consultar as informações de vendas registradas. O servidor hospeda as informações fornecidas pelos vendedores e responde às consultas dos gerentes.

Conclusão:

O projeto desenvolvido pela nossa equipe foi uma oportunidade de aplicarmos nossos conhecimentos em programação para criar uma solução eficiente e funcional de registro e consulta de vendas. Utilizando uma arquitetura cliente-servidor, com o uso de sockets e o banco de dados SQLite3, conseguimos estabelecer uma comunicação robusta entre os usuários e o servidor, garantindo a persistência e recuperação dos dados de vendas.

Através da linguagem de programação Python e a utilização de bibliotecas como socket, threading, sqlite3, entre outras, conseguimos implementar as funcionalidades necessárias para atender às demandas dos vendedores e gerentes. Além disso, a inclusão de blocos try-except em pontos estratégicos do código nos permitiu tratar exceções e evitar erros indesejados.

Como equipe, acreditamos que esse projeto nos proporcionou um aprendizado significativo e uma experiência prática na implementação de sistemas cliente-servidor. Ficamos satisfeitos com o resultado alcançado e com a oportunidade de trabalhar em conjunto, aplicando nossas habilidades individuais para construir um projeto coeso.