UNIVERSIDADE
FEDERAL DO CEARÁ
Novembro de 2025

Disciplina: Sistemas Distribuídos **Semestre:** 2025.2
Professor: Filipe Fernandes dos Santos Brasil de Matos

3ª Atividade Avaliativa Parcial (3ª AP)
Projeto Final

Contexto:

Com o crescente avanço e a ampla adoção de sistemas distribuídos em diversas áreas, como comércio eletrônico, saúde, finanças e redes sociais, torna-se cada vez mais crucial para futuros profissionais de Computação dominar as tecnologias que possibilitam a comunicação eficiente e confiável entre sistemas. A integração de serviços distribuídos exige o entendimento profundo de diferentes paradigmas de comunicação, cada um com suas particularidades, como simplicidade, desempenho, escalabilidade e interoperabilidade.

Este trabalho prático busca preparar os alunos para esses desafios ao propor o desenvolvimento de um sistema de gerenciamento de consultas médicas. Utilizando quatro abordagens distintas - Sockets, RPC/RMI, gRPC e Web Services (REST) -, os estudantes terão a oportunidade de explorar as características técnicas, vantagens e limitações de cada tecnologia, aplicando-as em um contexto realista e colaborativo.

Descrição Geral:

O trabalho propõe o desenvolvimento de um sistema voltado para o gerenciamento de consultas médicas. **O contexto de onde o sistema será aplicado (ex: clínica ou hospital) fica a critério de cada equipe.** O sistema possui um Lado Servidor e um Lado Cliente. O Lado Servidor é composto por quatro módulos principais interdependentes (além de um banco de dados para persistir informações do sistema):

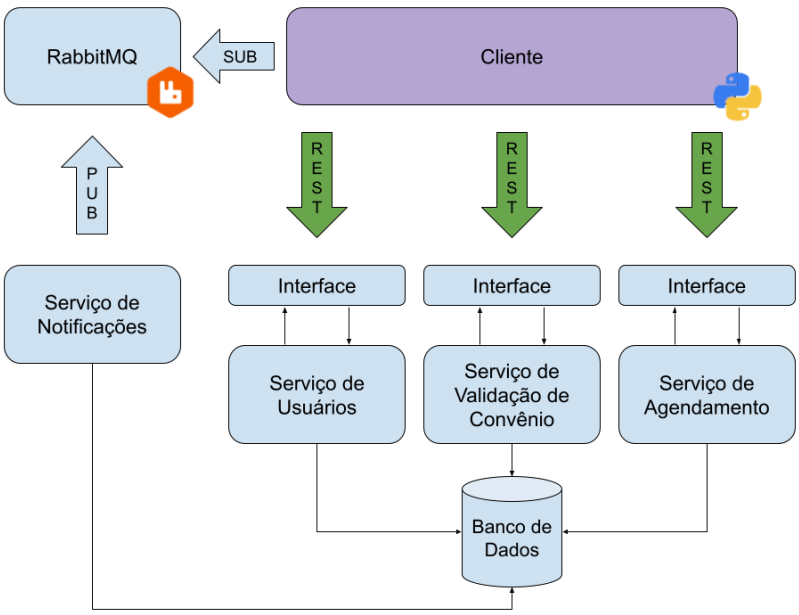
1. **Serviço de Usuários:** Gerencia todas as informações e operações relacionadas aos usuários da aplicação, controlando os dados de identidade, autenticação, autorização e perfis;
2. **Serviço de Agendamento:** Age como o coração do sistema, gerenciando os horários disponíveis, os agendamentos e coordenando as interações entre os outros módulos;

3. **Serviço de Notificações:** Informa os usuários em tempo real sobre mudanças no status das consultas (ex: confirmada e cancelada), garantindo uma experiência dinâmica.
4. **Serviço de Validação de Convênio:** Simula o processamento da elegibilidade do paciente ou pagamento, atualizando o status da consulta após a confirmação.

Cada módulo está associado a uma interface responsável pela interação com o Lado Cliente. O papel da interface é, além de criar N *endpoints*, encaminhar requisições do Lado Cliente para o serviço a ele associado, assim como retornar respostas deste serviço para o Lado Cliente. A comunicação entre a interface e o seu serviço deve ser realizada utilizando um dos seguintes mecanismos: Sockets, RPC/RMI ou gRPC. **Não pode haver repetição de mecanismo!**

Já o Lado Cliente é composto por N *scripts* em Python que simulam possíveis ações realizadas pelos clientes. A comunicação entre o Lado Cliente e o Lado Servidor poderá acontecer de maneira direta (via REST) ou indireta (via RabbitMQ). A comunicação direta será empregada entre o Cliente e as interfaces do sistema, enquanto a comunicação indireta deverá ser empregada entre o Cliente e o Serviço de Notificações. A comunicação indireta deverá ser feita através da ferramenta RabbitMQ, onde o Cliente e o Serviço de Notificações atuarão como *Subscriber* e *Publisher*, respectivamente.

A Figura abaixo apresenta uma visão geral da arquitetura planejada do sistema:



O sistema também deve ser totalmente containerizado por meio do Docker, garantindo isolamento, portabilidade e facilidade de implantação. Cada componente da Figura acima possuirá seu próprio *Dockerfile*, definindo o ambiente de execução, as dependências e os processos necessários para sua inicialização, permitindo o levantamento automatizado e padronizado do sistema em qualquer ambiente. Além disso, o Lado Servidor também deve ser

orquestrado usando Docker Compose, assegurando uma inicialização simplificada de todo o ecossistema da aplicação.

Funcionalidades Principais:

- **Agendamento de Consultas Médicas:** Permite o registro e o controle das consultas médicas no sistema. Ele possibilita que pacientes ou funcionários da clínica realizem o agendamento de atendimentos, selecionando o médico, a especialidade e o horário desejado, de acordo com a disponibilidade. O sistema deve oferecer mecanismos para evitar conflitos de horários e sobreposições de consultas, garantindo uma gestão eficiente da agenda médica;
- **Consulta de Status:** Permite o acompanhamento da situação das consultas registradas no sistema. Por meio dele, usuários podem verificar se uma consulta está **agendada**, **confirmada**, **cancelada** ou **concluída**, facilitando o monitoramento e a organização do fluxo de atendimentos. Além disso, o status pode ser atualizado automaticamente conforme o andamento do processo ou manualmente pelos profissionais autorizados;
- **Processamento de Validação:** Verifica a elegibilidade do paciente junto ao convênio médico ou a confirmação do pagamento particular da consulta. Esse serviço atua como uma etapa intermediária entre o agendamento e a confirmação definitiva do atendimento, garantindo que apenas consultas devidamente autorizadas ou quitadas sejam efetivadas. Após a validação, o sistema atualiza automaticamente o **status da consulta**, refletindo o resultado do processamento — seja a aprovação, a pendência ou a rejeição —, assegurando a consistência e a confiabilidade das informações no ambiente clínico-hospitalar;
- **Gerenciamento de Usuários do Sistema:** Controla o acesso e o perfil de todos os usuários que interagem com o sistema, incluindo **pacientes**, **médicos**, **recepcionistas** e **administradores**. Ele permite o cadastro, edição e exclusão de contas, bem como a definição de níveis de permissão e autenticação. Dessa forma, garante-se que cada usuário tenha acesso apenas às funcionalidades adequadas ao seu papel, assegurando a segurança e a integridade dos dados sensíveis do ambiente clínico-hospitalar.

Requisitos Técnicos:

1. Sockets (TCP/UDP):
    - Implementar a comunicação direta entre os módulos com mensagens binárias ou em texto.
    - Criar um protocolo simples para estruturar os dados trocados.
2. RPC/RMI:
    - Utilizar interfaces remotas para definir operações acessíveis entre os módulos.
    - Simular chamadas de métodos distribuídos.
3. gRPC:
    - Usar Protobuffer para definição das interfaces e mensagens entre os serviços;
    - Implementar comunicação eficiente baseada em HTTP/2, usando comunicação *streaming* (se necessário);
    - Adotar duas linguagens de programação distintas entre os módulos envolvidos.

Sobre a Entrega:

Serão permitidas equipes de até cinco alunos que devem participar ativamente da programação do trabalho. O trabalho deve ser entregue até o dia 06 de Janeiro de 2026 (Terça-Feira). No ato da entrega, um membro da equipe deverá subir a imagem do *container* Docker referente ao Lado Cliente no Docker Hub (https://hub.docker.com/) e criar um arquivo *zip* com todo o projeto do Lado Servidor. Como todo o projeto, entenda todos os recursos base para que eu consiga levantar o Lado Servidor na minha máquina usando o Docker Compose. A imagem do Lado Cliente deve estar preparada para que eu a execute no modo interativo, visando testes e análise de código. O arquivo *zip* e o endereço da imagem devem ser enviados para o meu e-mail institucional (filipe.fernandes@crateus.ufc.br). O e-mail deverá ter como título **[2025.2] 3ª Atividade Avaliativa de Sistemas Distribuídos** e como corpo ele deverá conter **I) Os nomes e matrículas dos integrantes da equipe, II) o *link* do container Cliente no Docker Hub; III) o arquivo zip do Lado Servidor; IV) um vídeo de 10 à 15 minutos apresentando as principais operações do projeto; e V) Uma planilha mapeando as ações dos usuários aos scripts cliente desenvolvidos e como executar o referido script para reproduzir a ação.** A Tabela abaixo ilustra uma sugestão de planilha. Fiquem livres para adicionar novas colunas, caso julguem necessário. **ATENÇÃO: Serão observadas a maneira como os arquivos enviados via e-mail foram organizados e o capricho/facilidade do mini-tutorial para rodar cada uma das versões do Trabalho Prático.**

AÇÃO	SCRIPT	COMO EXECUTAR?
Criar um usuário novo	users.py	1) cd /home/client/ 2) python users.py “criar” “Filipe” “123” “Admin”

Todos os trabalhos serão analisados e comparados. **Caso seja identificada cópia de trabalhos, todos os trabalhos envolvidos receberão nota ZERO. Caso o trabalho não execute ou seja submetido com arquivos faltantes, em um primeiro momento, será atribuída nota ZERO.**

Além disso, cada grupo deve apresentar seus respectivos trabalhos no meu gabinete. Nesse dia, **todos os membros do grupo DEVEM estar presentes (alunos ausentes ficarão sem nota do trabalho)**. O grupo deverá realizar alguns testes instruídos por mim. O grupo também deve fazer uma apresentação geral sobre os códigos e, eventualmente, responder questionamentos meus sobre a implementação. **É proibida a leitura de papéis ou textos durante a apresentação.** O apresentador deverá mostrar domínio do conteúdo e não apenas ler textos prontos. **Caso o apresentador apenas leia textos sua nota será zero.** Planeja-se que **a apresentação dos trabalhos ocorra entre os dias 13 e 15 de Janeiro de 2026 em um horário a ser definido.**

Qualquer modificação na especificação desse trabalho, os alunos serão notificados via SIGAA. Então, atenção ao sistema. Qualquer dúvida me procurem no Gabinete 10 do Bloco Didático 2, na Sala da Coordenação de Ciência da Computação do Bloco Administrativo ou me contactem via e-mail institucional (filipe.fernandes@crateus.ufc.br).

Equipes:

EQUIPE 1	EQUIPE 2
EQUIPE 3	**EQUIPE 4**
EQUIPE 5	**EQUIPE 6**
EQUIPE 7	**EQUIPE 8**
EQUIPE 9	**EQUIPE 10**
