Universidade Federal de Goiás
Instituto de Informática - Engenharia de Software
Software Concorrente e Distribuído

Chat Online

Flavimar da Silva Almeida
Giancarlo Oliveira Silva
Luca Baccheschi Benetti
Lucas Oliveira de Souza

Goiânia
2024

1. Introdução

Uma das atividades mais recorrentes no dia a dia das pessoas é a troca de mensagem por redes sociais. Pessoas separadas por grandes distâncias geográficas, podem enviar um texto para outra, de maneira rápida, podendo também participar de grupos com familiares e amigos, de forma que todos podem mandar mensagem e todos recebem essas mensagens. Esse tipo de comunicação requer que alguns pontos sejam atendidos, como comunicação eficiente, facilidade de compartilhamento de informações e possibilidade de resolução de problemas em tempo real. Com isso, o desenvolvimento de um software que envolva chat, deve tratar para que esses pontos da comunicação sejam atendidos.

Dessa forma, o presente trabalho visa desenvolver um sistema que traga consigo um chat, permitindo que duas pessoas distintas possam conversar entre si (comunicação entre pares), além de permitir também conversar com vários ao mesmo tempo, numa forma de grupo. Para que isso aconteça da forma correta, evitando problemas de concorrência, distribuição, dificuldade no compartilhamento, entre outras coisas, o desenvolvimento desse sistema irá abordar os diferentes conteúdos da disciplina de Software Concorrente e Distribuído.

2. Requisitos funcionais

O sistema a ser desenvolvido terá como requisitos funcionais principais:

- Cadastro de usuário por email;
- Conversa usuário 1x1;
- Conversa em grupo/sala;

3. Requisitos não funcionais

O sistema a ser desenvolvido terá como requisitos não funcionais:

- Será gerada uma imagem docker da aplicação, possibilitando o uso de containers;
- O sistema será hospedado em nuvem (AWS), utilizando serviços para hospedagens de imagens docker;
- O sistema utilizará filas (serviço AWS SQS) para o envio de Email para cadastro de usuário;
- O sistema utilizará serverless (serviço lambda) para o envio de email para envio de cadastro de usuário;
- Haverá uma aplicação web ASP.NET MVC, frontend e backend na mesma solução, com Javascript, HTML e CSS no front;
- Haverá uma aplicação serverless responsável por receber o evento de cadastro e enviar o email;

- Será utilizado comunicação com o frontend por meio de websockets, para que as imagens sejam refletidas em tempo real;
- Será utilizado o banco de dados Postgres, por meio do serviço RDS da AWS.

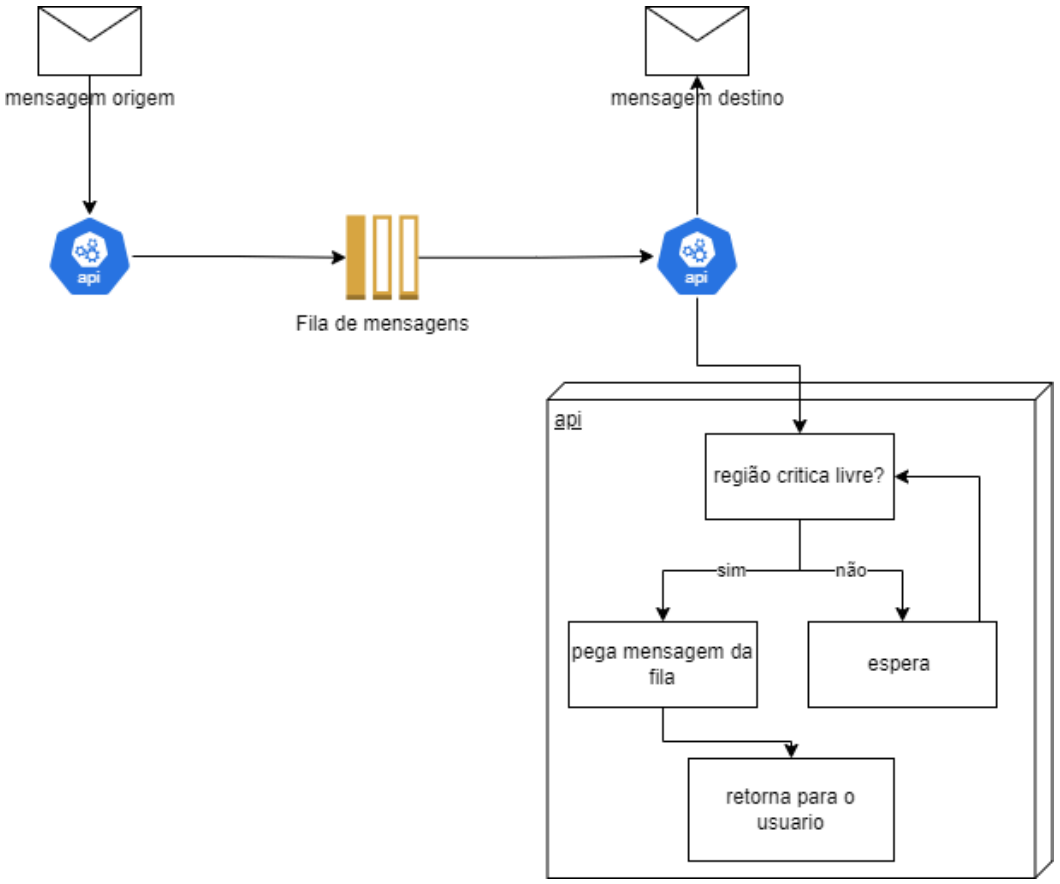
4. Relação com disciplina

Baseado nos requisitos apresentados, o sistema possui as seguintes relações com a disciplina:

- Utilização de containers (Docker), que possibilitará escalabilidade, tolerância a falhas, disponibilidade, desempenho.
- Aplicação distribuída.
- Aplicação baseada em eventos.
- Uso de conceitos serverless.
- Uso de filas.
- Uso de websockets.
- Uso de semáforos, controles de threads e tasks.
- A aplicação tem uma natureza distribuída e concorrente, pois vários usuários online poderão acessar a mesma sala. E também mensagens diretas.
- Backend será desenvolvido de acordo com os padrões API RestFUL.
- Deverão ser aplicadas lógicas inerentes à concorrência e distribuição, em relação aos vários usuários publicando mensagens de forma concorrente

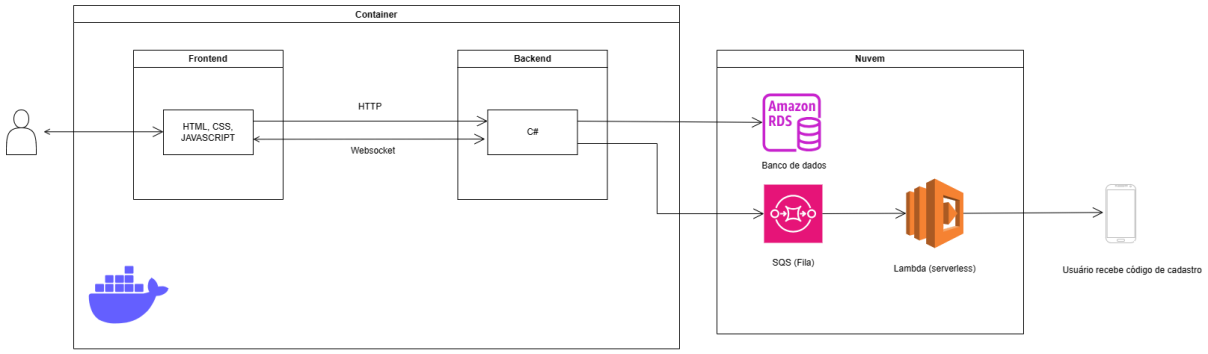
4.1 Visão de envio e recebimento das mensagens

Abaixo está o fluxo que a mensagem passará para chegar no usuário destino:



5. Arquitetura (alto nível)

A aplicação a ser desenvolvida terá a seguinte arquitetura:

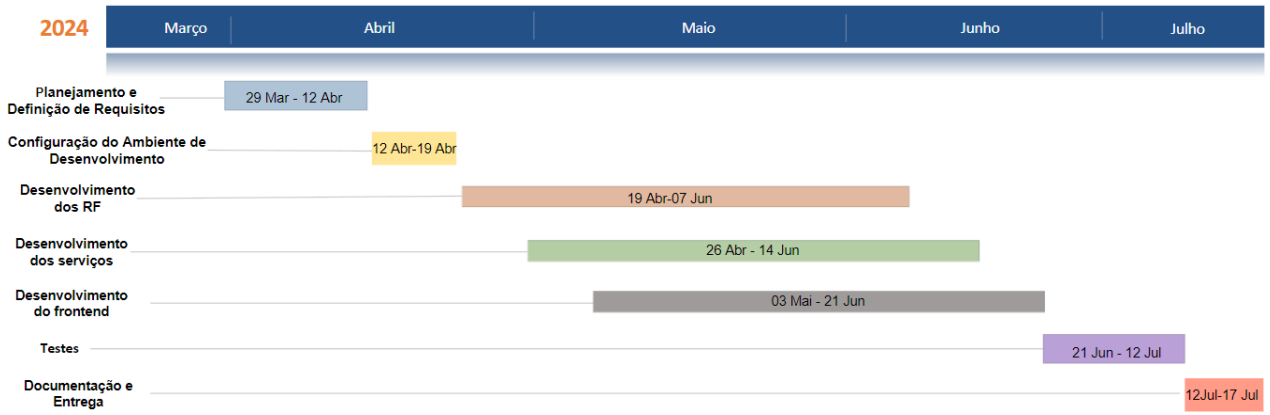


6. Papéis

Cada integrante da equipe ficará responsável por um elemento de desenvolvimento do software

- Luca (**Lider**) - Desenvolvedor Full Stack - foco em arquitetura
- Lucas - Desenvolvedor Full Stack - foco em back end
- Giancarlo - Desenvolvedor Full Stack - foco em front end
- Flavimar - Desenvolvedor Full Stack - foco em infra

7. Cronograma de desenvolvimento



7.1 Planejamento e Definição de Requisitos

- Detalhamento dos requisitos funcionais e não funcionais.
- Definição das tecnologias a serem utilizadas.
- Design preliminar da arquitetura do sistema.

7.2 Configuração do Ambiente de Desenvolvimento

- Configuração do ambiente Docker para desenvolvimento local
- Provisionamento dos serviços necessários na AWS (RDS, SQS, Lambda).
- Configuração do ambiente de desenvolvimento ASP.NET MVC.

7.3 Desenvolvimento dos RF

- Implementação do cadastro de usuário por número.
- Desenvolvimento da funcionalidade de conversa 1x1.
- Desenvolvimento das funcionalidades de envio e recebimento de mensagens.
- Implementação da conversa em grupo/sala.

7.4 Desenvolvimento dos serviços

- Integração do sistema com serviços AWS (RDS, SQS, Lambda).
- Configuração da comunicação por websockets

7.5 Desenvolvimento do frontend

- Implementação da interface de usuário para cadastro, do chat e outras funcionalidades.

7.6 Testes

- Testes unitários e de integração.
- Validação dos requisitos funcionais e não funcionais.

- Correção de bugs.

7.7 Documentação e Entrega

- Documentação técnica do projeto.
- Entrega do sistema funcional.