## 1. Identificação

Título do Projeto: Sistema Distribuído de Comunicação Inteligente para Operadora.
Autores: João Victor Fernandes Rocha, Thiago Aquino Guedes Barbosa.
Orientador: Prof. Paulo Alexandre Bressan.

## 2. Introdução e Motivação

Operadoras gerenciam um volume massivo e crescente de interações com clientes. A comunicação eficaz é vital não apenas para a operação (envio de faturas, avisos de manutenção), mas também para a retenção de clientes (suporte, marketing) e aquisição (cadastro de novos planos). A motivação deste projeto é aplicar os conceitos de sistemas distribuídos e paralelismo para resolver esse problema. Propõe-se uma arquitetura de microserviços onde cada função de comunicação (chat, e-mail em massa, marketing) é um serviço independente. Isso permite que tarefas pesadas, como o envio de e-mails, sejam executadas em paralelo por múltiplos "workers", sem impactar a latência dos serviços interativos. A utilização de filas de mensagens garante a assincronicidade e a resiliência do sistema, assegurando que nenhuma mensagem seja perdida, mesmo que um componente falhe temporariamente.

## 3. Objetivos

Nosso objetivo geral é projetar, implementar e avaliar um protótipo de sistema distribuído inteligente, baseado em microserviços, capaz de gerenciar de forma escalável, paralela e resiliente todo o fluxo de comunicação automatizada (e-mail e chat) entre uma operadora de serviços e seus clientes.

Objetivos Específicos:

**Modelar a Arquitetura:** Definir os limites e as APIs de cada microserviço.
**Implementar o Gerenciamento de Tarefas:** Desenvolver o núcleo do sistema) e uma fila de tarefas para orquestrar tarefas assíncronas.
**Desenvolver Workers Paralelos:** Criar works de e-mail que possam ser instanciados e escalados horizontalmente (ex: via Docker) para processar lotes de envio em paralelo.
**Construir o Bot de Atendimento:** Implementar o serviço de chat interativo, capaz de lidar

com requisições de baixa latência simultaneamente às tarefas em lote.
**Validar a Escalabilidade:** Realizar testes de carga para medir (e-mails/hora) do sistema ao adicionar mais instâncias de workers.
**Integrar Serviços:** Garantir a comunicação e a coleta de dados entre os serviços de automação (marketing, cadastro) e os serviços de execução (workers).

**4. Descrição Técnica**

A solução proposta utiliza obrigatoriamente microserviços como estilo arquitetural, com comunicação assíncrona para garantir o desacoplamento e a distribuição de tarefas. O sistema será dividido nos seguintes microserviços, comunicando-se primariamente via mensageria (ex: RabbitMQ ou Kafka) e APIs REST/gRPC:

1. **Bot Central (Interface e Configuração):**
    - **Função:** Ponto de entrada administrativo (operadora) para configurar mensagens, campanhas de e-mail e consultar relatórios consolidados (novos clientes, inadimplentes, faturamento).
    - **Tecnologias que podem ser usadas:** FastAPI (Python) para a API.
2. **Bot de Atendimento ao Cliente:**
    - **Função:** Contato direto com  cliente. Oferece respostas automáticas , identifica o problema e encaminha para atendimento humano quando necessário.
    - **Tecnologias que podem ser usadas:** FastAPI (Python) e MongoDB (ideal para armazenar fluxos de conversa e logs de chat).
3. **Gerenciador de Tarefas (Orquestrador):**
    - **Função:** O cérebro da distribuição. Recebe solicitações (ex: "enviar X cobranças") ou de outros serviços, divide-as em lotes menores e as enfileira.
    - **Tecnologias que podem ser usadas:** Celery (Python) como task queue e RabbitMQ como broker.
4. **Worker de E-mail (Execução Paralela):**
    - **Função:** Executa o envio real dos e-mails. É um serviço  que apenas consome tarefas da fila, gera o conteúdo dinâmico e envia via.
    - **Tecnologias que podem ser usadas:** Celery Workers (Python), Docker.
5. **Serviço de Automação de Marketing:**
    - **Função:** Analisa interações do Bot de Atendimento e agenda o envio automático de ofertas relacionadas, publicando tarefas no Gerenciador.

- **Tecnologias que podem ser usadas:** Estamos pesquisando sobre essa parte ainda.

6. **Serviço de Cadastro por E-mail:**
    - **Função:** Monitora uma caixa de entrada. Quando um cliente responde a um e-mail de cadastro, o serviço processa a resposta e cadastra o cliente no banco.
    - **Tecnologias que podem ser usadas:** Python (libs IMAP/SMTP).
7. **Serviço de Monitoramento e Relatórios:**
    - **Função:** Coleta métricas (logs, status de envio, erros) de todos os outros serviços e os consolida para exibição no Bot Central.

## 5. Cronograma

| Semanas de novembro | Atividades Principais |
|---|---|
| SEMANA 1 | Detalhamento de requisitos, definir APIs e contratos de dados. |
| SEMANA 2 | Implementação do Gerenciador de Tarefas e dos workers. |
| SEMANA 3 | Implementação do Bot de Atendimento e Bot Central. |
| SEMANA 4 | Implementação dos Serviços Auxiliares (Marketing, Cadastro) e Monitoramento. |

## 6. Resultados Esperados

Enviar milhares de e-mails em paralelo, reduzindo o tempo de execução em até 80%;
Atender múltiplos clientes simultaneamente via bot automatizado.Gerar relatórios consolidados com métricas de desempenho, engajamento e faturamento;
Demonstrar claramente os benefícios do paralelismo, evidenciando o ganho de eficiência frente a um modelo sequencial.
Como métrica de avaliação, será medido o tempo total de processamento e a taxa de sucesso dos envios, comparando a execução paralela com uma versão sequencial.

## 7. Referências


- Documentação Oficial do FastAPI. Disponível em: https://fastapi.tiangolo.com/
- Documentação Oficial do Celery Project. Disponível em: https://docs.celeryq.dev/
- Documentação Oficial do RabbitMQ. https://www.rabbitmq.com/documentation.htm