1. Descrição

1.1. Objetivo do Sistema

O principal objetivo deste sistema é desenvolver uma aplicação de chat distribuída que permita a comunicação entre dois ou mais usuários de forma eficiente e segura. O sistema será projetado para oferecer as seguintes funcionalidades principais:

- Registro e autenticação de usuários.
- Envio e recebimento de mensagens em tempo real.
- Armazenamento das mensagens em um banco de dados relacional para consultas futuras.

O sistema também busca garantir a integridade e segurança dos dados dos usuários, além de ser escalável para suportar múltiplos clientes simultaneamente.

1.2 Escopo

O escopo do projeto abrange o desenvolvimento de um sistema cliente-servidor com as seguintes características:

- **Servidor:** Gerencia conexões, autentica usuários, armazena mensagens no banco de dados e retransmite mensagens aos destinatários conectados.
- **Clientes:** Aplicações que se conectam ao servidor para enviar e receber mensagens.
- **Banco de Dados Relacional:** Armazena informações dos usuários, histórico de mensagens e sessões ativas.

O sistema será implementado com foco em comunicação síncrona (tempo real) utilizando WebSockets e comunicação assíncrona via APIs REST para operações como registro, login e consulta ao histórico.

1.3 Requisitos Funcionais

Necessidade 1	**Benefício**
Registro de Usuários	**Permitir que novos usuários acessem o sistema**
Id Func.	**Descrição das Funcionalidades/atores envolvidos**
F1.1	Criar conta (nome, email e senha)
	Usuário
F1.2	Validar dados do usuário antes de armazenar
	Servidor

Necessidade 2	Benefício
Login/Autenticação	Garantir acesso seguro ao sistema

Id Func.	Descrição das Funcionalidades/atores envolvidos
F1.1	Autenticar usuário com email e senha
	Usuário
F1.2	Validar credenciais no banco de dados
	Servidor
F1.3	Criar sessão autenticada após login
	Servidor

Necessidade 3	Benefício
Envio de Mensagens	Facilitar a comunicação entre usuários em tempo real

Id Func.	Descrição das Funcionalidades/atores envolvidos
F1.1	Enviar mensagem para outro usuário (chat individual) ou um grupo (chat em grupo)
	Usuário
F1.2	Armazenar mensagem no banco de dados
	Servidor
F1.3	Retransmitir mensagem ao(s) destinatário(s) conectado(s) no chat(individual ou em grupo)
	Servidor

Necessidade 4	Benefício
Recebimento de Mensagens	Garantir entrega de mensagens

Id Func.	Descrição das Funcionalidades/atores envolvidos
F1.1	Receber mensagens em tempo real via WebSocket
	Usuário
F1.2	Consultar novas mensagens no Banco
	Servidor

Necessidade 1	Benefício
Consulta ao histórico	Permitir acesso a mensagens anteriores armazenadas no banco de dados

Id Func.	Descrição das Funcionalidades/atores envolvidos
F1.1	Solicitar histórico de mensagens via API REST
	Usuário
F1.2	Retornar histórico solicitado ao cliente
	Servidor

1.4 Requisitos Não Funcionais

- Segurança: Hashing seguro das senhas e uso de protocolos seguros como HTTPS.
- Escalabilidade: Suporte a múltiplos clientes simultâneos.
- Desempenho: Respostas em tempo real para envio/recebimento de mensagens.
- Disponibilidade: Garantir que mensagens não entregues sejam armazenadas para posterior recuperação.

2. Visão

2.1 Arquitetura do Sistema

O sistema será baseado em uma arquitetura cliente-servidor com as seguintes características:

- Comunicação em tempo real entre clientes e servidor via WebSockets.
- API REST para operações como registro, login e consulta ao histórico de mensagens.
- Banco de dados relacional (MySQL) para armazenamento persistente.

2.2 Componentes Principais

- **Servidor:**
    - Linguagem: Node.js.
    - Funções:
        - Gerenciar conexões WebSocket.
        - Autenticar usuários via API REST.
        - Armazenar mensagens no banco de dados.
        - Retransmitir mensagens aos destinatários conectados.
- **Cliente:**
    - Interface gráfica simples desenvolvida com HTML/CSS/JavaScript.
    - Funções:
        - Registro e login do usuário.
        - Envio/recebimento de mensagens em tempo real.
        - Consulta ao histórico de mensagens.
- **Banco de Dados**
    - Modelo relacional com tabelas para usuários, mensagens e sessões (detalhado na seção 3.3).

2.3 Fluxo Geral do Sistema

- Registro:
    - O cliente envia os dados do usuário ao servidor via API REST.
    - O servidor valida os dados e os insere na tabela "Users".
- Login:
    - O cliente envia email e senha ao servidor via API REST.
    - O servidor verifica as credenciais na tabela "Users" e cria uma sessão na tabela "Sessions".

- Comunicação:
    - O cliente envia uma mensagem ao servidor via WebSocket, indicando o destinatário.
    - O servidor armazena a mensagem na tabela "Messages" e retransmite ao destinatário conectado ou a deixa disponível para consulta posterior.

3. Planejamento

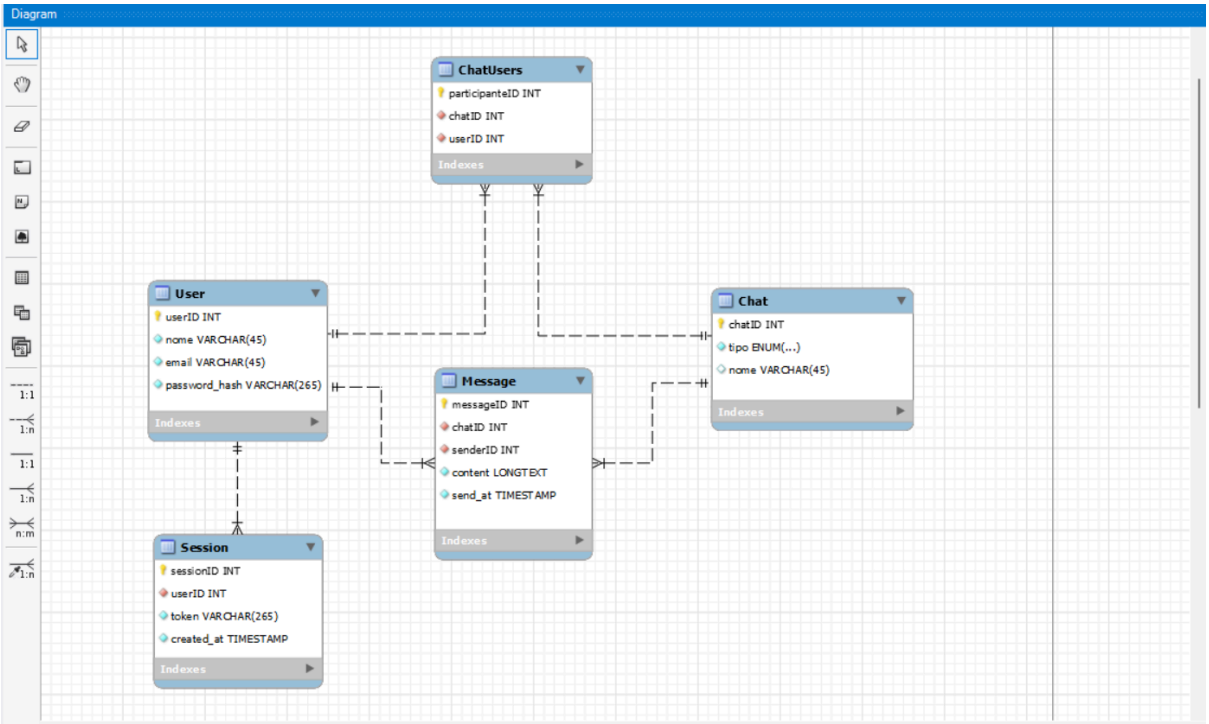
3.1 Cronograma

Fase	Atividades Principais	Duração Estimada
Planejamento Inicial	Definição dos requisitos, arquitetura e tecnologias	1 semana
Desenvolvimento BackEnd	Implementação do servidor, APIs REST e WebSocket	3 semanas
Desenvolvimento FrontEnd	Criação da interface gráfica	2 semanas
Integração	Conexão entre frontend, backend e banco	1 semana
Teste	Testes unitários, integração e carga	1 semana

3.2 Recursos Necessários

- **Tecnologias:**
    - BackEnd: Node.js.
    - FrontEnd: HTMÇ/CSS/JavaScript.
    - Banco de Dados: MySQL
- **Ferramentas:**
    - Ambiente de Desenvolvimento Integrado(IDE).
    - Ferramentas para Testes.
    - Servidor Local de Desenvolvimento.

3.3 Modelo Relacional do Banco de Dados



3.4 Riscos Potenciais

- Problemas com escalabilidade devido ao aumento no número de usuários simultâneos.
- Falhas na entrega em tempo real por instabilidade no WebSocket ou conexão dos clientes.
- Vulnerabilidades relacionadas à segurança, como ataques man-in-the-middle ou vazamento de credenciais.

3.5 Métricas de Sucesso

- Tempo médio para entrega das mensagens em tempo real (<200ms).
- Taxa de sucesso no login/autenticação (>99%).
- Capacidade do sistema suportar pelo menos 50 conexões simultâneas sem degradação significativa no desempenho.