

# 1 Requisitos Funcionais (RF)

## RF01 – Cadastro de usuário

- O sistema deve permitir que um usuário se cadastre fornecendo:
  - Nome
  - Email
  - Telefone
  - Senha
- O sistema deve validar se todos os campos obrigatórios foram preenchidos.
- O sistema deve armazenar a senha de forma segura usando hash (SHA-256).
- O sistema deve retornar uma mensagem de sucesso ou erro, incluindo o id do usuário cadastrado em caso de sucesso.

## RF02 – Login de usuário

- O sistema deve permitir que um usuário faça login fornecendo:
  - Email
  - Senha
- O sistema deve validar se todos os campos obrigatórios foram preenchidos.
- O sistema deve comparar a senha fornecida com a senha armazenada em hash.
- O sistema deve retornar:
  - `success: true` com dados do usuário em caso de login correto.
  - `success: false` com mensagem de erro em caso de falha (email ou senha incorretos).

## RF03 – Fale Conosco / Contato

- O sistema deve permitir que qualquer usuário envie uma mensagem de contato fornecendo:
  - Nome
  - Email
  - Telefone (opcional)
  - Assunto (opcional)
  - Mensagem
- O sistema deve validar se os campos obrigatórios (nome, email, mensagem) foram preenchidos.
- O sistema deve salvar a mensagem na tabela `clientes`.
- O sistema deve retornar uma resposta de sucesso ou erro.

## **RF04 – Listagem de contatos**

- O sistema deve permitir que um administrador consulte todas as mensagens enviadas via “Fale Conosco”.
- O sistema deve retornar os contatos em ordem decrescente de criação (data\_criacao).

## **RF05 – Verificação de saúde do servidor**

- O sistema deve fornecer um endpoint /health que retorna:
  - Status do servidor (Servidor rodando!)
  - Timestamp atual

## **RF06 – Endpoint de teste**

- O sistema deve ter um endpoint /test para verificar se o servidor está ativo, retornando uma mensagem simples.

# **2 Requisitos Não Funcionais (RNF)**

## **RNF01 – Segurança**

- Senhas devem ser armazenadas em hash (SHA-256), não em texto puro.
- O backend deve validar todos os dados recebidos para evitar inconsistências no banco de dados.

## **RNF02 – Performance**

- Conexões com o banco devem ser gerenciadas usando mysql2/promise com createPool para otimizar o desempenho.
- Operações de leitura e escrita devem ser assíncronas (async/await).

## **RNF03 – Disponibilidade**

- O servidor deve estar disponível na porta 3001.
- O endpoint /health deve permitir monitoramento do status do sistema.

## RNF04 – Escalabilidade

- O backend deve suportar múltiplas conexões simultâneas ao banco de dados.
- Estrutura modular de rotas facilita futuras expansões (ex.: adicionar mais endpoints).

## RNF05 – Usabilidade / Integração

- O backend deve permitir requisições **CORS** para integração com páginas HTML hospedadas localmente ou remotamente.
- As respostas devem estar no formato JSON para fácil consumo pelo frontend.

## RNF06 – Manutenção

- Código deve ter logs de erro (`console.log`) para facilitar a identificação de problemas.
- Estrutura organizada, com endpoints separados por funcionalidade.