# Relatório da Aplicação: Lista de Tarefas (To-Do)

## 1. Visão Geral do Projeto

Este projeto consiste em uma aplicação "To-Do List" (Lista de Tarefas), desenvolvida com uma arquitetura cliente-servidor. O objetivo é permitir que o usuário gerencie suas tarefas diárias de forma eficiente através de uma interface de usuário amigável.

A aplicação é composta por três componentes principais:

1. **Front-end (Cliente):** O foco principal do projeto. Uma interface de usuário moderna e responsiva construída com o **framework Quasar (baseado em Vue.js)**, projetada para uma experiência de usuário intuitiva.
2. **Back-end (Servidor):** Uma API RESTful robusta (Node.js/Express) que serve como "cérebro" da aplicação, gerenciando as regras de negócio e a comunicação com o banco de dados.
3. **Banco de Dados (Persistência):** Um banco de dados MySQL para armazenar e persistir todas as tarefas criadas pelos usuários.

## 2. Funcionalidades do Aplicativo (Visão do Usuário)

O aplicativo Quasar foi projetado para ser um assistente de produtividade.

### 2.1. Visualização Inteligente de Tarefas

Ao abrir o aplicativo, o usuário visualiza sua lista de tarefas quer mostra todos os dados como uma breve descrição, data final (caso haja) e exibe a coloração Vermelha( De alta urgência), Amarela(De urgência média) e Verde (De urgência baixa) ao lado do check-box, além disto o usuário recebe uma notificação caso haja tarefas a serem feitas no mesmo dia. A principal funcionalidade é a **priorização automática e filtros**: o aplicativo exibe primeiro as tarefas marcadas como de **maior prioridade** e dá a possibilidade de o usuário:

Filtrar por data final;

Filtrar tarefas por: pendentes, concluídas ou ambas;

Filtrar tarefas por urgência: Baixa, Média Alta.

O filtro deve ser utilizado clicando no canto inferior direito(Botão azul com ícone de funil).

O aplicativo exibe as tarefas atrasadas com um círculo vermelho para **maior prioridade**.

Isso ajuda o usuário a focar imediatamente no que é mais importante, sem precisar reordenar a lista manualmente.

### 2.2. Adicionar Novas Tarefas

Através de um botão (Localizado no canto inferior direito verde com símbolo de +) , o usuário pode abrir um formulário para registrar uma nova tarefa. Os campos disponíveis são:

* **Descrição:** O que precisa ser feito (campo obrigatório).
* **Data Final:** Um seletor de data permite definir um prazo limite para a conclusão.
* **Nível de Prioridade:** O usuário pode definir o quão importante é a tarefa (ex: Alta, Média, Baixa).

*Aviso Integrado:* O aplicativo valida os dados e não permite a criação de uma tarefa sem uma descrição, exibindo uma mensagem de erro clara ao usuário.

### 2.3. Gerenciamento de Tarefas (Concluir e Editar)

Cada tarefa na lista possui controles interativos:

* **Concluir Tarefa:** O usuário pode marcar uma tarefa como "concluída" (Através de uma *checkbox*). Isso atualiza o status da tarefa, movendo-a para uma seção de "Concluídas".
* **Editar Tarefa:** Se os detalhes de uma tarefa precisarem ser alterados (seja a descrição, o prazo ou a prioridade), o usuário pode acionar o modo de edição (Clicando no símbolo de lápis no canto direito da tarefa para atualizar as informações).

### 2.4. Excluir Tarefas

O aplicativo permite que o usuário remova permanentemente tarefas que não são mais necessárias, mantendo a lista limpa e organizada.(Clicando sob o símbolo de lixeira no canto esquerdo da tarefa)

## 3. Arquitetura e Tecnologias Utilizadas (Contexto Técnico)

Embora o foco seja o app Quasar, ele é sustentado por uma arquitetura de back-end moderna.

| **Componente** | **Tecnologia** | **Responsabilidade** |
| --- | --- | --- |
| **Front-end** | Quasar Framework (Vue.js) | Interface do usuário, experiência visual e consumo da API. |
| **Back-end** | Node.js + Express.js | Servidor da API, lógica de negócios, validações e gerenciamento de rotas. |
| **Banco de Dados** | MySQL | Armazenamento e persistência dos dados das tarefas. |
| **Comunicação** | mysql2/promise (Node.js) | Driver de conexão assíncrona entre o Node.js e o MySQL. |
| **Middleware** | cors (Node.js) | Permite que o front-end (Quasar) faça requisições ao back-end (Express). |
| **Middleware** | express.json() (Node.js) | Realiza o "parse" do JSON enviado pelo front-end para o servidor. |