Cidadão Ativo

O cidadão participando
nas decisões da cidade.

Nome do Estudante: Guilherme Melato
Católica Joinville SC - Centro Universitário
Curso: Engenharia de Software.
Data de Entrega: 10/07/2025

Resumo

Neste documento é apresentado o projeto "Cidadão Ativo", que busca aproximar a população da câmara de vereadores. O objetivo principal é incentivar uma participação mais ativa dos cidadãos nas discussões e decisões que impactam a cidade, oferecendo acesso facilitado a informações relevantes, por meio de uma aplicação eficiente, objetiva e acessível a todas as idades.

---

1. Introdução

Contexto: Este projeto visa desenvolver uma aplicação para divulgar proposições legislativas, a atuação parlamentar e o andamento das iniciativas dos vereadores. A meta primordial é garantir que todos os cidadãos estejam cientes do que está sendo proposto, implementado ou negligenciado na Câmara Municipal, incentivando a fiscalização e a responsabilização.

Justificativa: Atualmente, há uma lacuna em plataformas digitais para conectar cidadãos aos projetos de vereadores, dificultando o acompanhamento das propostas. Esta aplicação pretende auxiliar nessa interação, tornando a gestão pública mais transparente e inteirando os cidadãos na política.

Objetivo principal: Desenvolver uma aplicação web que aproxime o cidadão das decisões legislativas municipais, facilitando o acesso a informações e promovendo a participação ativa nas decisões.
Objetivos secundários:

- Permitir visualização dos projetos em votação por bairro;
- Criar sistema de votação popular (aprovar/desaprovar);
- Fornecer painel com estatísticas e relatórios;
- Oferecer canal para propostas da comunidade;
- Garantir acessibilidade e uso para pessoas idosas.

2. Descrição do Projeto

Tema do Projeto:

- Desenvolvimento de uma aplicação web para a divulgação de ações públicas do poder legislativo

Problemas a Resolver:

- Tornar mais transparente os projetos propostos na câmara de vereadores.
- Disponibilizar de forma clara as informações dos projetos.
- Informar o andamento dos projetos propostos.
- Propor projetos à câmara com base nos interesses da comunidade.

Limitações:

- Escalabilidade: O projeto inicial será feito para apenas uma cidade, podendo ter dificuldade se pessoas de outras cidades acessem e se cadastrem.
- Inclusão e acessibilidade: Nem todas as pessoas têm acesso à internet ou são habilitadas fisicamente para o uso apropriado do software.
- Abrangência de funcionalidades: Para ser algo objetivo e disponível para todas as idades, a implementação de funções complexas pode ser complexa.
- Dependência da Adesão: O sucesso da aplicação depende da adesão de vereadores e da população de forma ativa.
- Moderação dos dados: Tornar os números de votos e projetos criados autênticos para a confiança dos usuários.

---

3. Especificação Técnica

3.1 Requisitos de Software

Requisitos Funcionais:

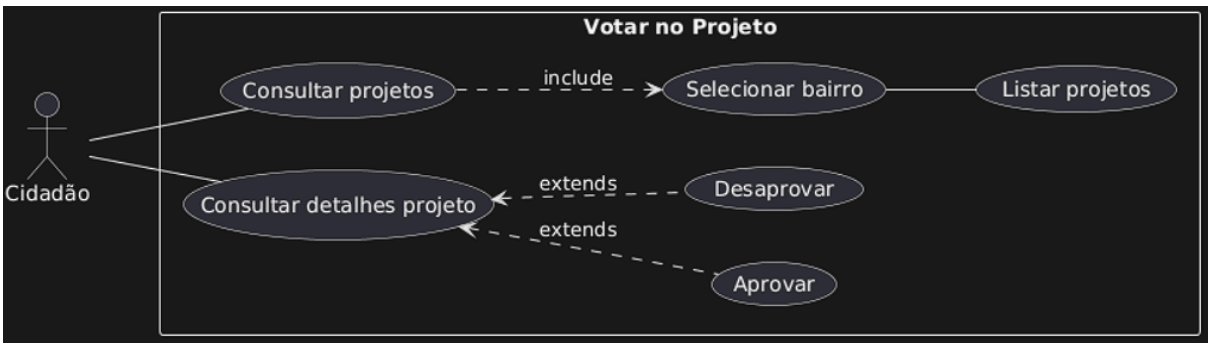
- O sistema deve permitir que o usuário se cadastre utilizando CPF, nome, data de nascimento e a definição de uma senha
- O sistema deve permitir que o usuário acesse sua conta utilizando CPF e senha.
- O sistema deve permitir que os usuários filtrem publicações por bairro desejado.
- O sistema deve permitir que os usuários votem em publicações utilizando upvote para apoiar e downvote para discordar.
- O sistema deve permitir que **o Vereador** e o **Cidadão** enviem propostas de melhorias para os Administradores.
- O sistema deve permitir que o **Administrador** possa editar publicações, incluindo exclusão das mesmas e edição de texto.
- O sistema deve permitir que o **Administrador** possa criar e editar dashboards.

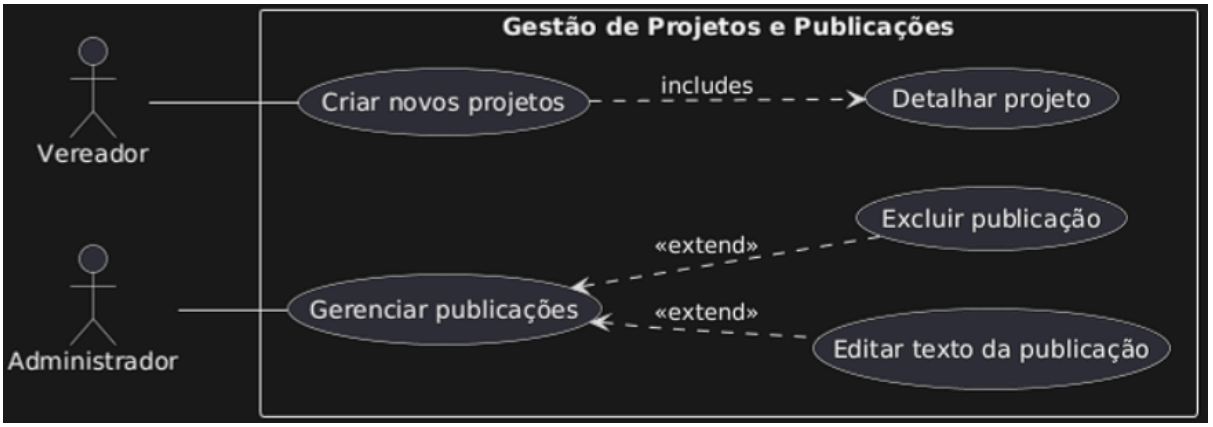
- O sistema deve permitir que o **Vereador** e o **Cidadão** tenham acesso aos dashboards.
- O sistema deve permitir que o **Vereador** possa criar novos projetos, mas sem editar ou excluir projetos já existentes.
- O sistema deve permitir que os **Cidadãos** e **Vereadores** denunciem publicações inadequadas para revisão dos Administradores.
- O sistema deve permitir que os **Administradores** gerenciem os perfis de usuários, podendo bloquear ou restringir acessos quando necessário.
- O sistema deve exibir um ranking das publicações mais votadas.
- O sistema deve permitir que os **Vereadores** justifiquem seus projetos com descrições detalhadas e anexos.
- O sistema deve gerar relatórios sobre o número de interações, votos e propostas enviadas, acessíveis para **Administradores**.

Requisitos Não-Funcionais:

- O sistema deve ser capaz de aguentar uma quantidade considerável de usuários (Com base na porcentagem de cidadãos numa cidade, já que não será 100% da cidade que utilizará a aplicação).
- O sistema deve **processar uma votação** em no máximo 2 segundos
- Todas as senhas dos usuários devem ser armazenadas utilizando **criptografia SHA-256** ou superior.
- O sistema deve estar em conformidade com a **LGPD (Lei Geral de Proteção de Dados)**, garantindo o consentimento do usuário para coleta de dados, caso haja.
- As ações administrativas (edição, exclusão, bloqueio de usuários) devem ser **registradas em logs de auditoria**.
- As cores e contrastes do sistema devem permitir **leitura confortável** para usuários com baixa visão.
- O backend do sistema deve utilizar **NestJs**, e o banco de dados deve ser **MySQL**

Diagramas de casos de uso:





3.2 Considerações de Design

Escolhas de Design:

- Foi selecionada uma aplicação PWA à uma aplicação mobile, pois o custo aumentaria muito para disponibilizar para os diferentes SO mobile existentes, podendo alcançar mais usuários.
- A interface será minimalista e com filtros claros para disponibilizar o conteúdo de forma direta, facilitando o uso para todas as idades, especialmente para usuários de mais idade.

Visão Inicial de Arquitetura:

- frontend: Interface web para os cidadãos (visualizar projetos, votar, filtrar), vereadores (cadastrar projetos) e admins (criar e editar dashboards, editar os projetos, aprovar projetos), além de enviar requisições HTTP para o backend
- backend: API RESTful que gerencia solicitações dos usuários (cadastros, votações e filtros), e irá validar as requisições enviadas pelo front através dos módulos de autenticação e moderação
- Banco de Dados: Banco relacional para guardar as informações de usuários, projetos e dashboards

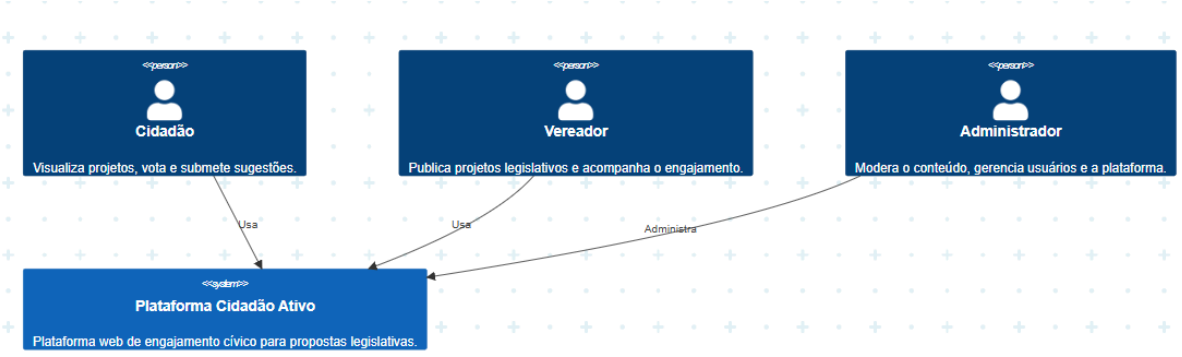
- Módulo de Autenticação: Gerencia o login e o tempo de sessão dos usuários no geral
- Módulo de Moderação: Permite que os moderadores aprovem projetos, criem e editem dashboards e editem os projetos

Padrões de Arquitetura:

- O método de Cliente-Servidor será utilizado, permitindo uma boa, eficiente e simples comunicação entre o cliente (frontend) e o servidor (backend).
- O padrão MVC será adotado para organizar a lógica de negócios facilitando a manutenção.

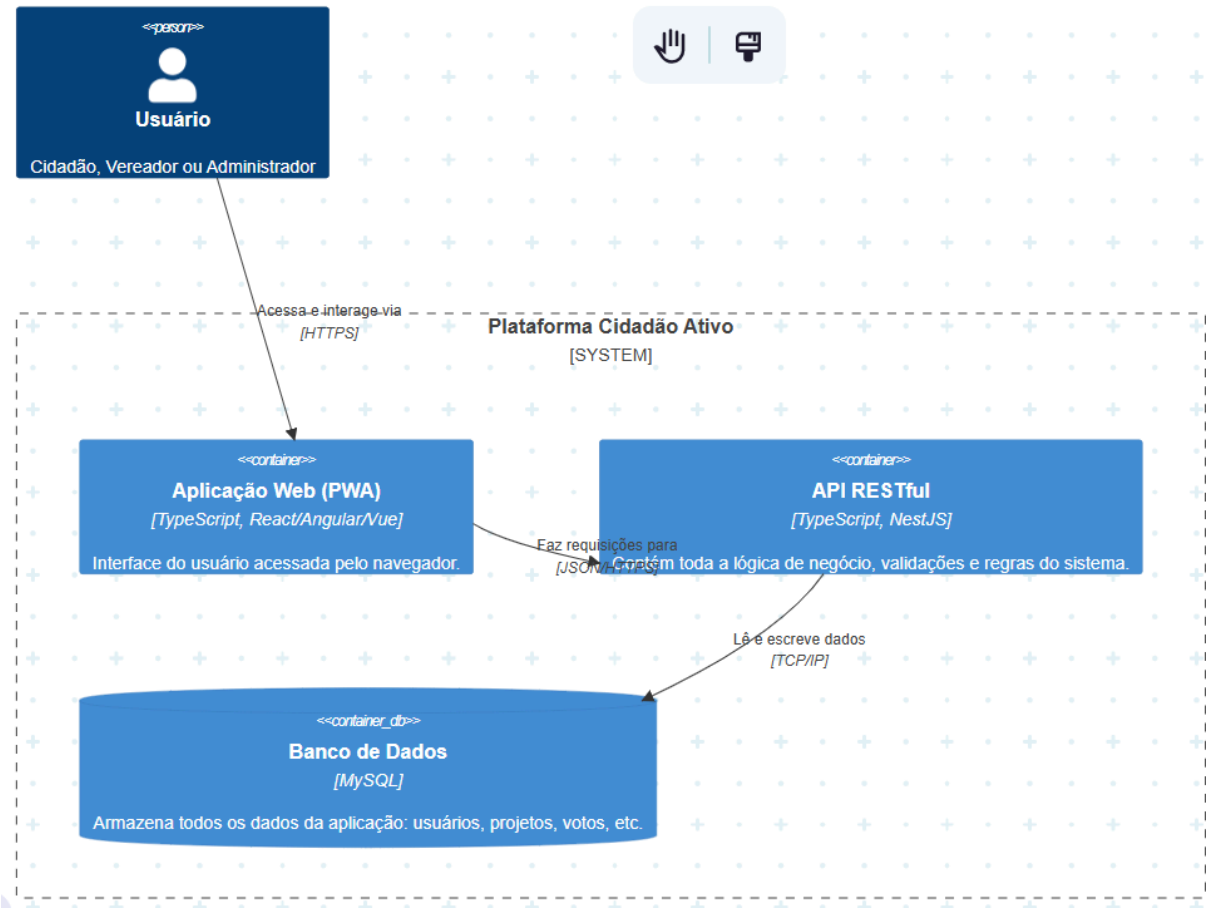
Modelo C4 (Detalhamento da Arquitetura em Níveis):

Nível 1: Contexto (Visão Geral)



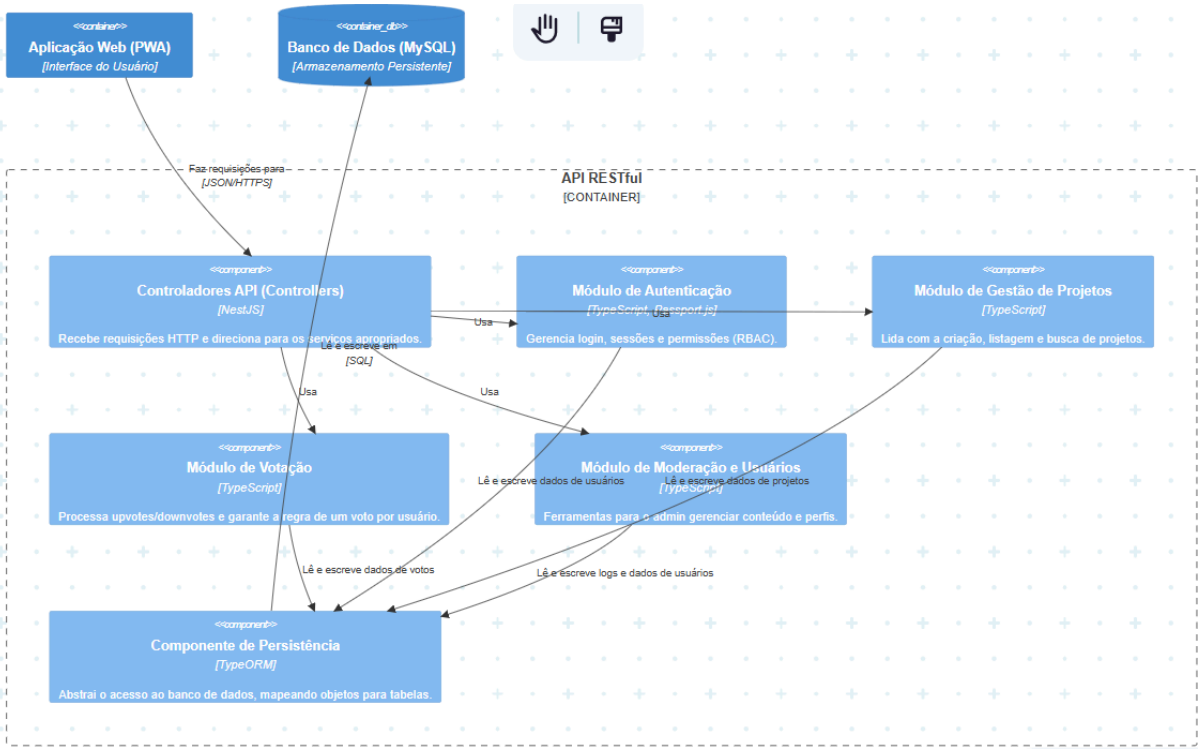
- Atores (Quem interage com o sistema?):
    1. Cidadão: O usuário principal, que utiliza a plataforma para se informar, participar de votações e fiscalizar o trabalho legislativo.
    2. Vereador: O agente público que publica propostas e utiliza a plataforma como um canal de transparência e comunicação com seus eleitores.
    3. Administrador: O operador responsável por garantir a integridade da plataforma, moderar o conteúdo e gerenciar os acessos.
- Resumo do Contexto: A plataforma "Cidadão Ativo" se posiciona como uma ponte digital entre os Atores e o processo legislativo municipal, promovendo a participação e a transparência.

Nível 2: Contêineres (Estrutura da Aplicação)



- Aplicação Web (PWA - Progressive Web App): A interface com o usuário, construída para ser acessível em qualquer dispositivo com um navegador. É a porta de entrada para todas as interações e foi escolhida por seu baixo custo e amplo alcance.
- Tecnologia: TypeScript (utilizando um framework como React, Angular ou Vue).
- API RESTful (Backend): O coração do sistema, onde toda a lógica de negócio é executada. Ele processa as ações dos usuários (votos, publicações), aplica as regras e gerencia os dados de forma centralizada e segura.
- Tecnologia: NestJS (framework Node.js com TypeScript).
- Banco de Dados (Relacional): O repositório persistente de informações. Armazena de forma estruturada todos os dados essenciais, como perfis de usuários, projetos, votos, comentários e logs de auditoria.
- Tecnologia: MySQL.
- Fluxo Principal: O Usuário utiliza a Aplicação Web, que se comunica via HTTPS com a API RESTful. A API, por sua vez, realiza as operações necessárias no Banco de Dados para ler ou salvar informações.

Nível 3: Componentes (Organização Interna da API)



- Controladores (API Controllers): Recebem as requisições da Aplicação Web e as direcionam para os serviços corretos.
- Módulo de Autenticação: Responsável por todo o fluxo de login, validação de permissões (Cidadão, Vereador, Admin) e segurança de acesso.
- Módulo de Gestão de Projetos: Contém a lógica para criar, listar, filtrar e gerenciar as propostas legislativas e sugestões dos cidadãos.
- Módulo de Votação: Garante a integridade do sistema de votos (upvote/downvote), aplicando a regra de "um voto por usuário" e calculando a popularidade dos projetos.
- Módulo de Moderação e Gestão de Usuários: Fornece as ferramentas para os Administradores gerenciarem usuários e conteúdo.
- Componente de Persistência (TypeORM): Uma camada de abstração que facilita a comunicação com o Banco de Dados, traduzindo as operações da lógica de negócio em comandos SQL.
- Colaboração: Esses componentes são desenhados para serem coesos e desacoplados, permitindo que o sistema seja mantido e expandido de forma organizada.

Nível 4: Componentes (Organização Interna da API)

- Para o "Cidadão Ativo", isso seria a implementação das classes dentro do NestJS, como ProjetosController, AuthService, VotoEntity, etc.
- Este nível é muito granular e é omitido neste resumo, sendo mais útil para os desenvolvedores durante a implementação

3.3 Stack Tecnológica

Linguagens de Programação:

- A escolha do TypeScript como linguagem principal do projeto foi estratégica pelas suas vantagens. Sua tipagem estática opcional garante robustez, confiabilidade e detecção precoce de erros, otimizando o desenvolvimento. Facilita a manutenção em larga escala, tornando o código mais legível e seguro para refatoração. Essencial para escalabilidade, permite sistemas modulares e claros. O vasto ecossistema TypeScript, com ferramentas avançadas e interoperabilidade com JavaScript, assegura uma transição suave.

Frameworks e bibliotecas:

- O NestJS será empregado como o framework principal para o desenvolvimento do sistema, devido à sua arquitetura modular e escalável, que favorece a construção de aplicações robustas e de fácil manutenção. Além disso, serão integradas bibliotecas de autenticação para garantir a segurança e a integridade dos dados, implementando mecanismos como OAuth2 e JSON Web Tokens (JWT) para um controle de acesso eficiente e seguro.
- O projeto utilizará as bibliotecas `common`, `typeorm`, `jwt`, `swagger` e `config` para garantir robustez, segurança, organização e documentação da API. **`common`** padronizará funcionalidades gerais e utilitários. **`typeorm`** será o ORM para interação com o banco de dados, facilitando operações CRUD. **`jwt`** será empregada para autenticação e autorização via JSON Web Tokens, criando uma API stateless e escalável. **`swagger`** gerará documentação interativa da API, auxiliando no desenvolvimento, testes e depuração. Por fim, **`config`** gerenciará as configurações da aplicação em diferentes ambientes, separando informações sensíveis do código-fonte.

Ferramentas de Desenvolvimento e Gestão de Projeto:

- As ferramentas a serem utilizadas serão o VSCode, MySql e ClickUp (para gerenciamento do projeto).

3.4 Considerações de Segurança

- Prevenir acessos duplicados: Implementando o cadastro através do CPF, nome e data de nascimento.
- Manipulação dos votos: Mitigada ao definir somente um voto por usuário.
- Propostas validadas: Integridade das propostas criadas validadas pelos logins dos Administradores cadastrados no banco de dados.
- Uso indevido da Plataforma: Monitorar as postagens através de logs no BD

4. Próximos passos

- **Desenvolvimento do Mockup da Aplicação Web:**
  - Criação de um protótipo visual detalhado da interface do usuário (UI).
  - Foco em usabilidade intuitiva e design atraente.
- **Implementação do Sistema de Autenticação no Backend:**
  - Priorização da segurança e controle de acesso.
  - Inclusão de mecanismos de registro, login e gerenciamento de senhas.
  - Possível integração com provedores de autenticação de terceiros.

5. Referências

- Software Engineering RFC and Design Doc Examples and Templates
- Common web application architectures - .NET | Microsoft Learn
- 10 Software Architecture Patterns You Must Know About

7. Avaliação dos Professores

Assinatura: _____

Assinatura: _____

Assinatura: _____