Relatório Trabalho Integrador
Sistema de Gerenciamento de Atas de Registro de Preços

Aysha Thayná
Erickson Müller

18 de Julho de 2025

1 Visão geral do projeto

O SIGARP - Sistema de Gerenciamento de Ata de Registros de Preços será utilizado pela Regional Oeste da Secretaria de Estado de Justiça e Reintegração Social para administrar o andamento de licitações e disponibilizar a consulta por diversos setores.

2 Objetivos do sistema

Manter um controle e histórico de licitações, possibilitando o cadastro e atualização de itens, fornecedores, lances, pedidos e demandas. As diversas licitações em processo silmutâneo podem ser verificadas no sistema.

O objetivo deste sistema não é substituir a rotina administrativa vigente, mas sim auxiliar na comunicação das situações de atas em diversos setores que estão espalhados em mais de 6 cidades.

3 Tecnologias utilizadas

- **Backend:** Node.js com Express; Passport.js, bcrypt; pg-promise.
- **Frontend:** HTML, CSS e Javascript; Axios, jQuery.
- **Banco de Dados:** PostgreSQL.

4 Arquitetura e lógica de implementação

A estrutura de repositórios dentro da pasta */src/* foi separada em */back_end/*, */front_end/* e */database/*.

4.1 Arquitetura Cliente-Servidor

Em */back_end/* estão os arquivos do servidor node iniciado com o yarn. A lógica do servidor está no arquivo *server.js*, o servidor expõe endpoints RESTful API para manipulação dos dados com o banco de dados através do *pg-promise*.

No */front_end/*, páginas em HTML são renderizadas pelo navegador usando a lógica interativa do Javascript e sendo estilizadas por CSS. Então existem 3 subpastas: */html/*, */css/* e */js/*.

Em */database/* está o arquivo de criação do banco de dados *dbSIGARP.sql*, assim como outros arquivos de inserção de tuplas no banco. Para acessar o sistema, deve-se rodar o script do arquivo $dbCREATE - Admin.sql$ que cria o usuário administrador com seu devido login e senha.

4.2 Criptografia e Autenticação

Para autenticação, o módulo Passport trabalha em harmonia com o JSON Web Token. O bcrypt faz a comparação da senha inserida com a armazenada no banco de dados e a função isAuthenticated() autoriza a rota.

5 Detalhamento de lógica por funcionalidade

Para a autenticação, o arquivo login.html coleta as credenciais do usuário, a requisição de login é enviada ao backend e este verifica as credenciais na tabela usuario do banco de dados, com a senha criptografada.

Para o cadastro de novas licitações, itens, usuários, unidades, etc o arquivo de cadastro obedece ao padrão de nome cadastro_xxxx.html e apresenta um formulário para que o usuário insira os dados.

O Javascript recebe essas informações pelo arquivo `script_cadastro_XXXX.js` através da biblioteca JQuery e chama uma função `createPost_XXXX()` que, usando a bibliotaca Axios, manda os dados para o post no backend através do link `/cadastro_XXXX_rota`.

O post fará a validação da requisição, em seguida insere a tupla no banco de acordo com os dados da rota. A situação de cada etapa do post pode ser analisada no terminal do servidor.

Para consultar as licitações cadastradas, o arquivo `consulta_licitacoes.html` exibe uma tabela e, ao carregar a página, um script `script_consulta_licitacoes.js` cria um array vazio faz uma requisição GET para o backend, via Axios. O backend consulta a tabela `licitacao` no banco de dados e retorna a lista de licitações em formato JSON. O Javascript do frontend, então, recebe esses dados e os renderiza na tabela da página usando a função renderizarLicitacoes().

Cada linha dessa tabela possui um botão **editar** e **remover**. Dentro da função renderizarLicitacoes(), o click é reconhecido e enviado para as determinadas rotas. Ao clicar em **editar**, o sistema nos direciona para a página */editar_licitacao/* com o número e ano da licitação a ser editada. Um processo similar ao de cadastro está presente nessa tela, o usuário poderá alterar a descrição da licitação e a alteração é efetuada pelo método PUT. Em seguida o usuário é redirecionado novamente para a tela de consulta de licitações.

Ao clicar em remover, a função **removerLicitacao()** é invocada, um alert aparece na tela e o *delete* é solicitado no backend com o ano e número da licitação. O método delete exclui a licitação do banco de dados.

6 Facilidades e dificuldades encontradas

6.1 Facilidades

Neste projeto, usamos a stack ensinada em aula de Programação II, isso facilitou muito no início do projeto pois usamos o template do esqueleto de um sistema para fazer o CRUD e a autenticação/autorização.

6.2 Dificuldades

O modelo de banco de dados escolhido é grande e complexo, o que acaba exigindo um grande número de telas no sistema para fazer essa integração.

7 Repositório do Github

https://github.com/erickson-cc/SIGARP.git