Universidade do Oeste de Santa Catarina
Disciplina: Unidade 1 - Prática Extensionista IV
Alunos: Natália Carolina Dilli, Giovani Pedro Zanatta, Luiz F. D. Demarck e Gabriela A. Z. Nunez

ENTREGA FINAL PRÁTICA EXTENSIONISTA IV

Link do GitHub do projeto: https://github.com/nataliadilli/Hidrogestor

Link do Site: https://hidrogestor.onrender.com/Pagina_inicial/index_inicial.html

Objetivos do Sistema

O sistema tem como propósito central registrar, organizar e armazenar de forma estruturada os dados referentes aos moradores e às respectivas unidades residenciais. Além disso, possibilita a realização e o controle das leituras mensais dos hidrômetros. Com base nessas leituras, o sistema calcula automaticamente o consumo individual de cada unidade, garantindo precisão nos resultados e reduzindo significativamente a probabilidade de erros manuais.

Outra funcionalidade essencial consiste na geração automática de relatórios e faturas, permitindo que administradores e usuários tenham acesso rápido, padronizado e organizado às informações de consumo. O sistema também incorpora mecanismos de auditoria e manutenção de histórico, possibilitando análises comparativas e acompanhamento da evolução do consumo ao longo do tempo. Todas essas funcionalidades operam de maneira integrada, contribuindo para a centralização, confiabilidade e eficiência na gestão dos dados relacionados ao uso da água.

Escolha da Empresa de hospedagem

Para a implantação do Hidrogestor, optou-se pela plataforma Render em razão de sua praticidade, suporte a múltiplos serviços e facilidade de integração entre frontend, backend e banco de dados. A decisão baseou-se principalmente na necessidade de centralizar toda a infraestrutura do projeto em um único ambiente, permitindo que o site estático, a API e o banco PostgreSQL operassem de forma integrada, com comunicação estável, implantação simplificada e gerenciamento unificado.

A plataforma também oferece deploy automatizado a partir do GitHub, o que facilita atualizações, correções e disponibilização de novas versões sem exigir configurações complexas de servidor. Além disso, o Render disponibiliza um banco PostgreSQL totalmente gerenciado, reduzindo a necessidade de manutenção manual, garantindo maior confiabilidade no armazenamento de leituras, moradores e faturas, e permitindo escalabilidade conforme a demanda.

O frontend estático é hospedado gratuitamente, com entrega otimizada ao usuário final, enquanto a API em Node.js é configurada como um serviço web independente, responsável por receber e processar requisições. Para viabilizar a infraestrutura, utilizou-se o plano gratuito disponibilizado pela plataforma: o banco de dados conta com 256 MB de RAM, 0,1 CPU e 1 GB de armazenamento, enquanto a API opera com 512 MB de RAM e 0,1 CPU, ambos com possibilidade de expansão conforme a necessidade do sistema. Essa abordagem centralizada garante organização, desempenho e facilidade de manutenção ao projeto como um todo.

Descrição Técnica da API de Exportação de Dados

O processo de exportação inicia-se quando o cliente acessa uma das rotas específicas por meio do botão Exportar CSV. A partir do ID da unidade consumidora, o servidor realiza consultas ao banco de dados utilizando o módulo pg, recuperando

as informações pertinentes. Nas rotas de leitura, são obtidos dados como consumo registrado, mês de referência e data da leitura. Já nas rotas de faturas, retornam valores cobrados, mês faturado e a leitura correspondente, esses resultados são recebidos pelo servidor em formato JSON.

Para converter esses dados em um arquivo CSV, utiliza-se o json2csv, por meio da classe Parser, responsável por estruturar as colunas, gerar cabeçalhos automaticamente e montar um conteúdo organizado, compatível com ferramentas como Excel.

Com o conteúdo gerado, o servidor configura a resposta HTTP definindo o tipo de arquivo como CSV e atribui nomes padrão como *leituras_ID.csv* ou *faturas_ID.csv*, permitindo que o navegador interprete corretamente o conteúdo para download, sem necessidade de criação de arquivos temporários no servidor.

A API também inclui mecanismos de tratamento de erro: quando não há registros para exportação, retorna-se o status 204 (No Content), indicando ausência de dados; em caso de falhas na consulta ou no processo de conversão, é enviado um 500 (Internal Server Error) acompanhado de mensagem explicativa. Essa abordagem garante um processo de exportação eficiente, direto, padronizado e compatível com diferentes ferramentas externas, oferecendo aos administradores e usuários uma forma prática de obter e analisar informações.

Diagrama de arquitetura DevOPs