CAGECE - Serviço de Medição de Água

Gemini

Um sistema web moderno para medição e controle de consumo de água.

Dashboard Interativo

- **Visualização de estatísticas** e medições recentes.
- Acesso rápido às principais funcionalidades.

Registro de Medições

- Formulário **intuitivo** para registrar leituras do hidrômetro.
- **Cálculo automático** do consumo.

Histórico Completo

- Visualização e gerenciamento de **todas as medições**.
- **Filtros** para busca e ordenação.
- **Exportação de dados** em formato JSON.

Relatórios e Análises

- **Gráficos e análises** de consumo por período.
- Identificação de tendências e maiores consumidores.

Stack do Projeto

- **Svelte 3**: Para um frontend reativo e performático.
- **Go**: Para um backend robusto e escalável.
- **SQLite**: Como banco de dados principal.
- **Docker**: Para orquestração e deploy.

Arquitetura Geral

O projeto CAGECE é composto por um frontend em Svelte, um backend em Go e um banco de dados SQLite, orquestrados via Docker.

Frontend com Svelte

O frontend é construído com Svelte, um framework que compila o código para JavaScript vanilla, resultando em uma aplicação rápida. Ele é responsável pela interface do usuário e interação.

```
// src/App.svelte
<script>
  import Login from "./components/Login.svelte";
  import Dashboard from "./components/Dashboard.svelte";
  import { user } from "./auth.js";
</script>

<main>
  {#if $user}
    <Dashboard />
  {:else}
```

Backend com Go

A API REST desenvolvida em Go oferece endpoints para todas as operações da aplicação, como autenticação, manipulação de dados de usuários e medições, e funcionalidades avançadas como predições de consumo e comparação.

```go
// backend/main.go
func main() {
    // ...
    mux := http.NewServeMux()
    mux.HandleFunc("/api/login", loginHandler)
    mux.Handle("/api/measurements", authMiddleware(http.HandlerFunc(
measurementsHandler)))
    mux.HandleFunc("/api/measurements/", measurementHandler)
    mux.Handle("/api/measurements/predict", authMiddleware(http.
HandlerFunc(predictionHandler)))
    mux.HandleFunc("/api/users", usersHandler)
```

Banco de Dados: SQLite

Utilizamos SQLite para um banco de dados leve e serverless, ideal para a portabilidade e facilidade de configuração que o Docker nos proporciona. Ele armazena todas as informações de usuários, medições e aparelhos.

```
CREATE TABLE IF NOT EXISTS measurements (
    "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    "user_id" INTEGER,
    "meterNumber" TEXT,
    "currentReading" REAL,
    "previousReading" REAL,
    "consumption" REAL,
    "price" REAL,
    "location" TEXT,
    "notes" TEXT,
    "timestamp" DATETIME,
```

Próximas Funcionalidades

- [ ] Sincronização com servidor remoto
- [ ] Notificações de consumo alto
- [ ] Gráficos mais avançados
- [ ] Backup automático
- [ ] Múltiplos usuários
- [ ] Relatórios em PDF

Fim

Obrigado!