Especificação de Requisitos
DS 2023/2 - Grupo 5
Mateus Cordova, Andrei Bueno, Kalani Sosa

Sistemas Integrados para Eficiência Energética em Ambientes Educacionais

1. Introdução

1.1 Propósito do Documento

O documento de arquitetura de software tem como propósito fornecer uma visão detalhada da estrutura, organização e interações entre os componentes do sistema, com foco nos três elementos principais: Arduino, Dashboard e API. Aqui estão os propósitos específicos para cada um desses componentes:

1.2 Arduino:

Visão Detalhada do Hardware e Sensores:

Descrever a arquitetura interna do Arduino, incluindo os componentes físicos e a interação com sensores (como sensores de corrente, temperatura, umidade, etc.).

Fluxo de Dados e Controle:

Ilustrar o fluxo de dados e o controle dentro do Arduino, especialmente como ele coleta informações em tempo real do ambiente.

1.3 Dashboard:

Estrutura da Interface do Usuário:

Apresentar a estrutura da interface do usuário no Dashboard, destacando os principais elementos visuais, como gráficos, controles e áreas de exibição.

Interação com o Usuário:

Detalhar como o Dashboard interage com o usuário, incluindo funcionalidades como personalização de preferências, alertas visuais e navegação.

1.4 API:

Integração entre Hardware e Dashboard:

Descrever como a API facilita a comunicação entre o hardware (Arduino) e o Dashboard, permitindo a transferência eficiente de dados em tempo real.

Segurança e Autenticação:

Abordar as práticas de segurança implementadas na API, incluindo autenticação e autorização para proteger o acesso não autorizado.

Endpoints e Funcionalidades:

Listar e detalhar os endpoints disponíveis na API, indicando suas funcionalidades específicas, como coleta de dados, exportação e notificações.

Ao cumprir esses propósitos, o documento de arquitetura de software visa criar uma referência compreensível para desenvolvedores, arquitetos e outros membros da equipe, proporcionando uma base sólida para o desenvolvimento, manutenção e evolução do sistema. Além disso, serve como uma ferramenta de comunicação eficaz para alinhar as expectativas entre as partes interessadas, garantindo uma implementação coesa e bem-sucedida.

2. Design da Arquitetura de Software

2.1 Desenvolvimento do Hardware em Arduino:

O hardware em Arduino desempenha um papel crucial na coleta de informações em tempo real do ambiente. Deve ser capaz de capturar dados relacionados ao consumo energético, iluminação, temperatura e umidade. Para isso, serão utilizados sensores específicos e módulos de medição que garantirão a precisão e a confiabilidade dos dados coletados. A integração desses componentes permitirá a criação de um sistema robusto capaz de monitorar e analisar o desempenho energético do ambiente.

2.2 API de integração entre Hardware, Dashboard e Banco de Dados

Além da coleta de dados, será implementada uma API de integração entre o hardware, o dashboard e o banco de dados. Essa API facilitará a comunicação entre esses elementos, garantindo uma transferência eficiente e segura de informações. Dessa forma, os dados coletados serão armazenados de maneira organizada no banco de dados, pronto para serem acessados e utilizados pela aplicação web.

2.3 Desenvolvimento da Aplicação Web (Dashboard) para Controle:

O painel de controle, ou dashboard, será a interface principal para o usuário final. Deverá ser projetado levando em consideração diversas características essenciais para proporcionar uma experiência eficaz e amigável. Entre essas características, destacam-se:

Sistema de Alertas e Notificações: Implementação de um sistema que alerta o usuário sobre eventos críticos relacionados à eficiência energética, permitindo uma rápida resposta a situações adversas.

Interface Intuitiva: O design do dashboard deve ser intuitivo, facilitando a compreensão e a navegação para usuários de diferentes níveis de habilidade técnica. Ícones claros, gráficos legíveis e uma disposição lógica dos elementos contribuirão para uma experiência positiva.

Suporte à Exportação de Dados: Possibilidade de exportar dados para análises mais detalhadas, relatórios personalizados ou compartilhamento com outras plataformas. Isso permitirá uma maior flexibilidade no uso das informações coletadas.

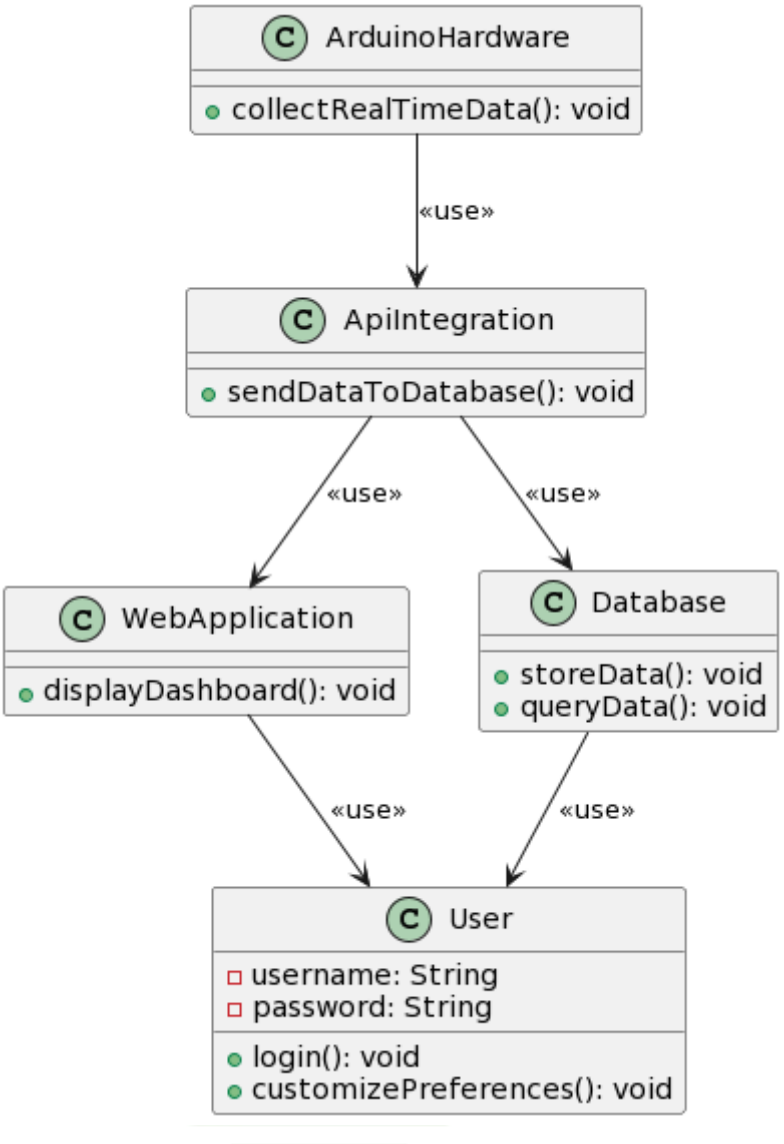
Gráficos de Consumo e Eficiência: Inclusão de gráficos visuais que apresentem de forma clara e compreensível os padrões de consumo energético e a eficiência do sistema ao longo do tempo. Esses gráficos auxiliarão o usuário na identificação de tendências e na tomada de decisões informadas.

Personalização de Preferências: Oferecimento de opções para que o usuário possa personalizar configurações e preferências, adaptando o sistema às suas necessidades específicas.

Ao unir eficientemente o hardware em Arduino, a API de integração e a aplicação web, o projeto proporcionará um sistema completo e integrado para monitoramento e controle da eficiência energética no ambiente desejado.

3. Técnicas de Engenharia de Software

O projeto será dividido em 3 partes distintas de desenvolvimento, Desenvolvimento do hardware em arduino, API de integração entre Hardware, Dashboard e Banco de Dados, Desenvolvimento da aplicação web (Dashboard) para controle



3.1 Arduino Controller

Arduino Board:

Um dos seguintes Arduinos poderão ser escolhidos Arduino Uno, Arduino Nano, ou Arduino Mega.

Sensores de Consumo Energético:

Sensor de corrente (ACS712) para medir a corrente elétrica. Sensor de tensão para medir a voltagem.

Sensores de Iluminação:

Fotodiodo ou Sensor de Luz (LDR - Light Dependent Resistor).

Sensores de Temperatura e Umidade:

Sensor de temperatura (por exemplo, DHT11 ou DHT22).

Sensor de umidade.

Módulos de Comunicação:

Módulo Wifi (ESP8266) para conectar o Arduino à rede.

Módulo Bluetooth, se a comunicação sem fio for necessária.

Módulos de Relé:

Relés para controle de dispositivos elétricos (por exemplo, para ligar/desligar luzes ou aparelhos).

Display:

Display LCD ou OLED para mostrar informações ao usuário.

Buzzer ou LED:

Para criar alertas sonoros ou visuais em caso de eventos críticos.

Bateria e Módulo de Gerenciamento de Energia (Opcional):

Se o projeto envolver monitoramento em locais remotos, considere a inclusão de uma fonte de energia independente, como baterias, e um módulo para gerenciar o consumo de energia.

Caixa de Projeto:

Uma caixa ou invólucro para proteger o circuito e os componentes.

Fios e Conectores:

Fios jumper e conectores para interligar os componentes.

Resistores e Capacitores:

Componentes para ajustar níveis de sinal e estabilizar a alimentação.

3.2 API em Python

Coleta de Dados:

Capacidade de receber dados provenientes do hardware em Arduino, como informações de consumo energético, iluminação, temperatura e umidade.

Implementação de rotas ou endpoints dedicados para diferentes tipos de dados.

Armazenamento em Banco de Dados:

Integração com o banco de dados para armazenar os dados coletados de maneira organizada.

Suporte para diferentes tipos de bancos de dados, dependendo das necessidades do projeto (MySQL, PostgreSQL, MongoDB).

Segurança:

Implementação de medidas de segurança, como autenticação e autorização, para proteger o acesso à API.

Atualização em Tempo Real:

Capacidade de atualizar o banco de dados em tempo real à medida que novos dados são recebidos do hardware.

API de Consulta:

Criação de endpoints que permitam a consulta dos dados armazenados no banco de dados.

Suporte para parâmetros de consulta para facilitar a obtenção de dados específicos.

Notificações e Alertas:

Implementação de um sistema de notificações para alertar o dashboard ou outros sistemas sobre eventos críticos detectados pelo hardware.

Opções de personalização para configurar preferências de notificação.

Tratamento de Erros:

Gerenciamento adequado de erros para garantir a estabilidade da API.

Retorno de códigos de status HTTP apropriados em casos de sucesso ou falha.

Documentação:

Elaboração de documentação clara e completa, descrevendo os endpoints disponíveis, os formatos de dados aceitos e retornados, além das instruções de autenticação.

Testes Automatizados:

Desenvolvimento de testes automatizados para verificar a funcionalidade da API.

Testes de integração para garantir a comunicação eficiente com o hardware e o banco de dados.

Logs e Monitoramento:

Geração de logs detalhados para rastrear atividades e facilitar a identificação de problemas.

Integração com ferramentas de monitoramento para acompanhamento em tempo real do desempenho da API.

Suporte a Formatos de Dados:

Aceitação de diferentes formatos de dados, como JSON, para facilitar a integração com o hardware e outros sistemas.

3.3 Web Interface

Página Inicial:

Apresentação de uma visão geral do estado atual do ambiente monitorado, incluindo informações de consumo energético, temperatura, umidade e iluminação.

Gráficos e Estatísticas:

Inclusão de gráficos visuais que representem o histórico e padrões de consumo energético ao longo do tempo.

Estatísticas resumidas para fornecer uma rápida compreensão do desempenho geral.

Personalização da Interface:

Opções para personalizar a exibição dos dados de acordo com as preferências do usuário.

Escolha de temas visuais e layouts.

Alertas e Notificações:

Sistema de alertas para informar o usuário sobre eventos críticos, como picos de consumo ou falhas no sistema.

Configurações para personalizar preferências de notificação, incluindo e-mails ou mensagens push.

Exportação de Dados:

Funcionalidade para exportar dados históricos e relatórios em formatos comuns (CSV, Excel) para análises mais detalhadas.

Controle Remoto:

Se o projeto permitir, integração de controles remotos para dispositivos elétricos, como luzes, permitindo ao usuário ligar/desligar remotamente.

Interface Intuitiva:

Design intuitivo com navegação fácil para garantir uma experiência de usuário agradável.

Uso de gráficos, ícones e cores para tornar as informações mais compreensíveis.

Dashboard Responsivo:

Garantir que o dashboard seja responsivo para se adaptar a diferentes tamanhos de tela e dispositivos.

Histórico de Atividades:

Registro de atividades recentes e histórico de ações realizadas pelo usuário.

Configurações do Usuário:

Página de configurações para permitir que o usuário personalize suas preferências e ajustes do sistema.

Suporte a Múltiplos Usuários:

Implementação de autenticação segura e suporte a múltiplos usuários com diferentes níveis de acesso.

Documentação e Ajuda:

Inclusão de seção de ajuda e documentação dentro da aplicação para orientar os usuários sobre o uso e funcionalidades disponíveis.

Suporte a Navegadores:

Garantir compatibilidade com os principais navegadores para ampliar a acessibilidade.

Performance Otimizada:

Otimização de desempenho para garantir tempos de carregamento rápidos e eficiência na resposta às solicitações do usuário.

Logs e Monitoramento:

Implementação de registros de atividades para rastreamento e monitoramento contínuo do desempenho do Dashboard.

4. Especificação de tecnologias

4.1Tecnologias Envolvidas

Considerando os três componentes principais do projeto (Arduino, API e Dashboard), identificarmos as tecnologias comumente utilizadas para atender às necessidades específicas de cada parte:

4.2 Arduino (Hardware):

Linguagem de Programação:

C/C + +: É a linguagem principal para programação de Arduino.

4.3 API (Integração entre Hardware, Dashboard e Banco de Dados):

Linguagem de Programação:

Python: Uma linguagem comumente usada para desenvolver APIs devido à sua simplicidade e vasta biblioteca.

Framework Web para API:

Flask ou Django (ambos em Python): Facilitam o desenvolvimento rápido de APIs.

Banco de Dados:

SQL (PostgreSQL ou MySQL): Para armazenar dados relacionais.

MongoDB:

Se dados NoSQL forem mais adequados para o projeto.

ORM (Object-Relational Mapping):

SQLchemy (Python): Facilita a interação com bancos de dados relacionais.

Segurança:

HTTPS: Para criptografar a comunicação.

JWT (JSON Web Tokens): Para autenticação.

4.4 Dashboard (Desenvolvimento da Aplicação Web):

Linguagens de Marcação/Estilo/Script:

HTML, CSS, JavaScript: Para desenvolver a estrutura, estilo e interatividade do dashboard.

Framework Web para Frontend:

React, Angular, ou Vue.js: Para criar interfaces de usuário interativas.

Comunicação com API:

AJAX ou Fetch API: Para fazer solicitações assíncronas à API.

Gráficos e Visualizações:

Bibliotecas como Chart.js, D3.js ou Plotly para criar gráficos interativos.

Segurança Frontend:

HTTPS: Para garantir uma comunicação segura com a API.