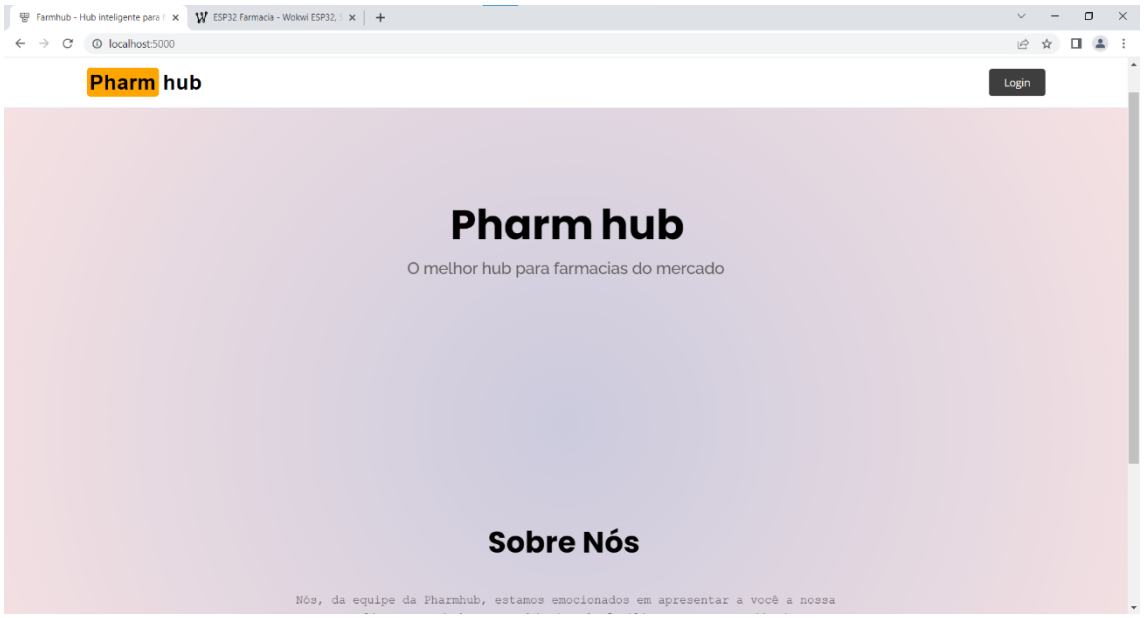
Relatório final Experiencia criativa:

O contexto da nossa aplicação é um site onde algum proprietário de farmácia possa registrar atividades básicas de sua farmácia, mas além disso também consegue monitorar as condições de umidade e temperatura em que seus produtos estão armazenados.

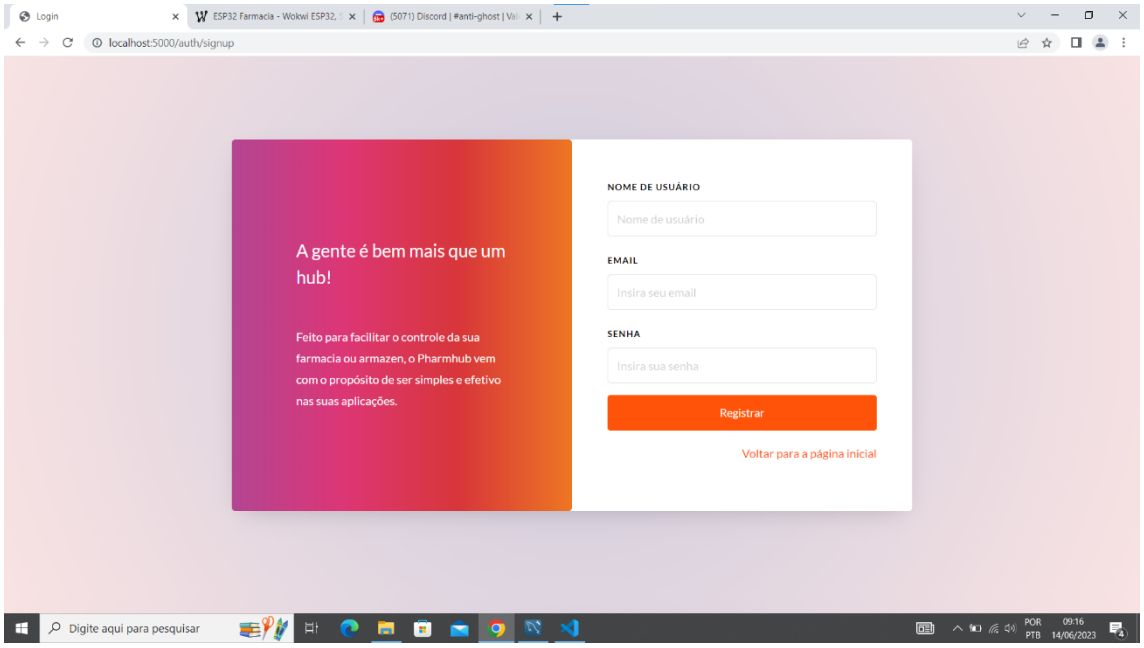
Páginas Web: No nosso projeto utilizamos o Flask para a criação das rotas de nossas páginas e utilizamos o Bootstrap para o design, entre as páginas criadas temos uma página de login e cadastro onde foi utilizado o Flask-Login, além de páginas de listagem e uma página para acessarmos os dados que vem dos sensores da parte do IOT.

Páginas web da nossa aplicação:

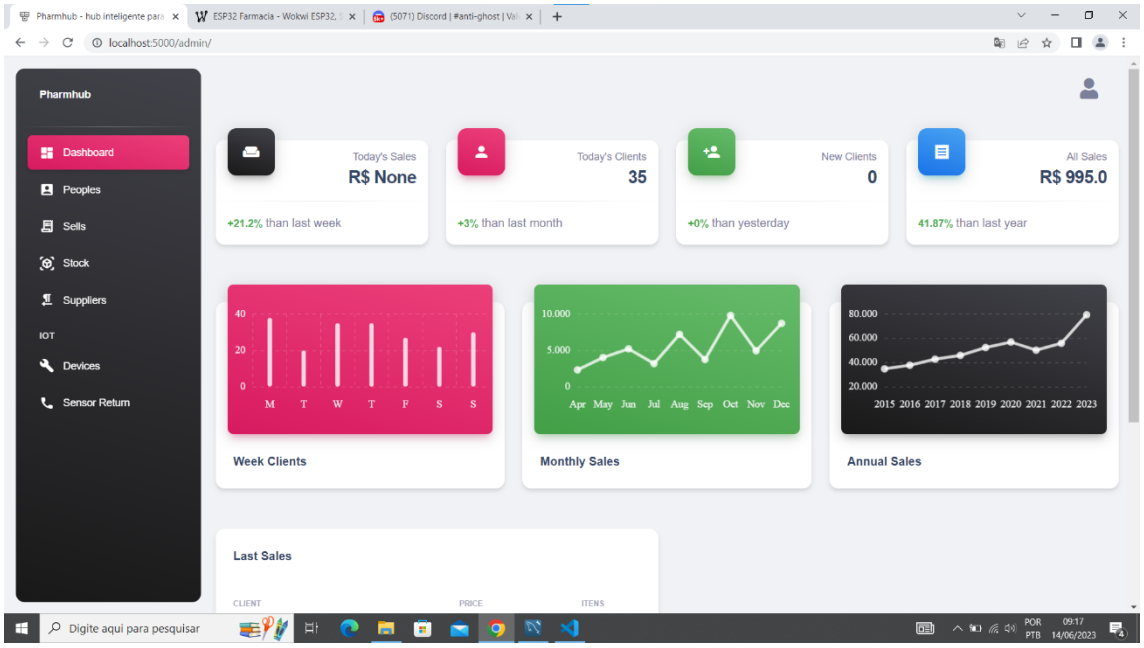
- Página principal com nossa logo e botão para login.



- Cadastro de usuário.



- Dashboard com informações e tabela com registro das vendas mais recentes.



- Index de vendas que direcionam para as páginas de registro de vendas e listagem de vendas.

- Index (people) que seleciona entre clientes e funcionários direcionando para a página de registro e listagem de clientes ou de funcionários.

- Index de Produtos (stock) que direcionam para as páginas de registro de produtos e listagem de produtos.

- Index de fornecedores (Suplies) que direcionam para as páginas de registro de fornecedores e listagem de fornecedores.

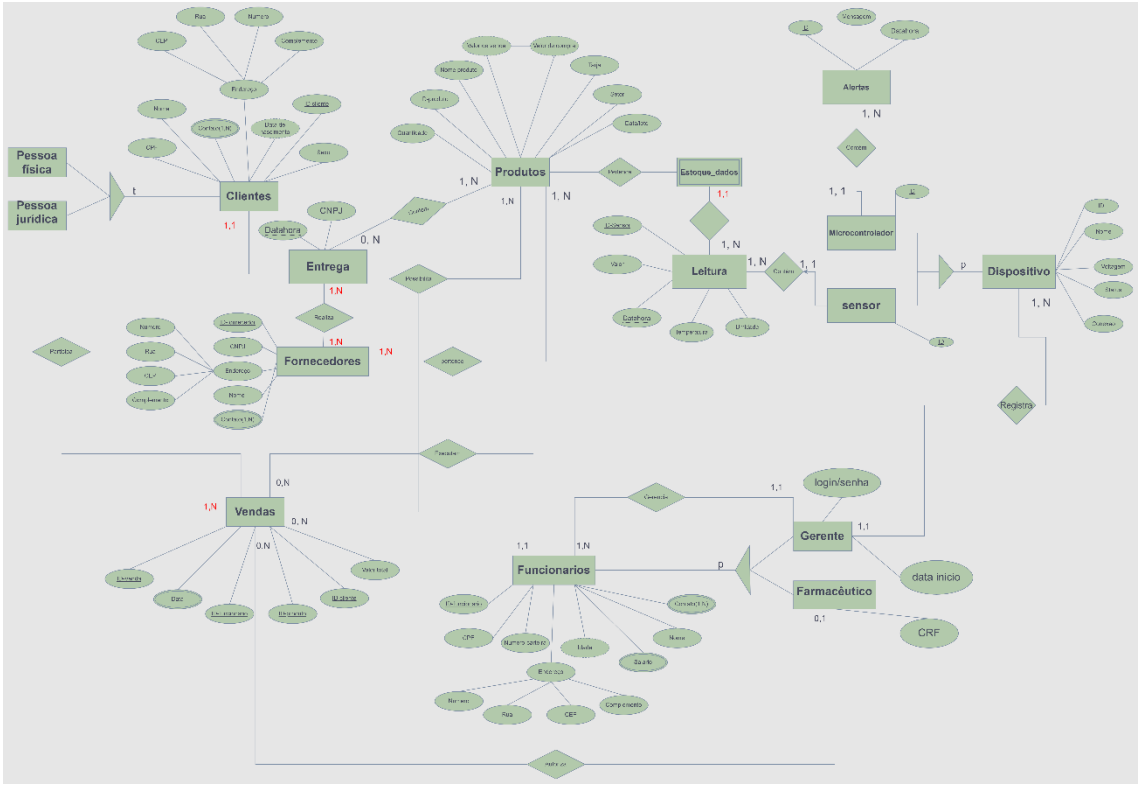
- Index de dispositivos (Dispositivos) que direcionam para as páginas de registro de sensores e microcontroladores e listagem de sensores e microcontroladores.

(As anteriores que não possuem imagens são semelhantes, apenas páginas com inputs para cadastrar algo)

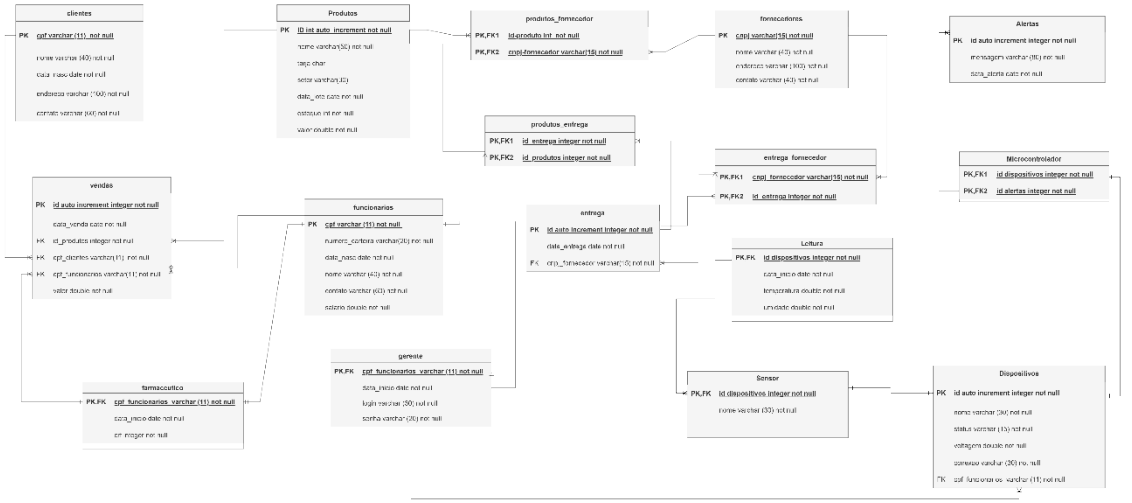
- Página para a visualização dos dados recebidos do sensor DHT22.

Banco de dados: Utilizamos em nossa aplicação o MySQL para armazenar os dados inseridos na aplicação, as tabelas foram todas criadas utilizando o Flask e a biblioteca SQLALchemy do Flask e também foram feitos todos os métodos de inserir, selecionar, editar e excluir dados do banco isso tudo tendo como base nossas modelagens conceituais e lógicas feitas anteriormente e já apresentadas em aula.

Modelagem conceitual:



Modelagem lógica:



Modelagem física:

```sql
CREATE DATABASE  IF NOT EXISTS `farmacia`;
USE `farmacia`;

DROP TABLE IF EXISTS `clientes`;

CREATE TABLE `clientes` (
  `cpf` varchar(11) NOT NULL,
  `name` varchar(50) DEFAULT NULL,
  `address` varchar(50) DEFAULT NULL,
  `contact` varchar(50) DEFAULT NULL,
  `birth_date` datetime NOT NULL,
  PRIMARY KEY (`cpf`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `devices`;

CREATE TABLE `devices` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `brand` varchar(30) DEFAULT NULL,
  `model` varchar(30) DEFAULT NULL,
  `voltage` float NOT NULL,
  `description` varchar(512) DEFAULT NULL,
  `is_active` tinyint(1) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `entrega`;

CREATE TABLE `entrega` (
  `id` int NOT NULL AUTO_INCREMENT,
  `date` datetime NOT NULL,
  `cnpj_fornecedor` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `cnpj_fornecedor` (`cnpj_fornecedor`),
  CONSTRAINT `entrega_ibfk_1` FOREIGN KEY (`cnpj_fornecedor`) REFERENCES `fornecedor` (`cnpj`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `entrega_fornecedor`;

CREATE TABLE `entrega_fornecedor` (
  `id_entrega` int NOT NULL,
  `cnpj_fornecedor` varchar(15) NOT NULL,
  PRIMARY KEY (`id_entrega`,`cnpj_fornecedor`),
  KEY `cnpj_fornecedor` (`cnpj_fornecedor`),
  CONSTRAINT `entrega_fornecedor_ibfk_1` FOREIGN KEY (`id_entrega`) REFERENCES `entrega` (`id`) ON DELETE CASCADE,
  CONSTRAINT `entrega_fornecedor_ibfk_2` FOREIGN KEY (`cnpj_fornecedor`) REFERENCES `fornecedor` (`cnpj`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

```sql

  DROP TABLE IF EXISTS `farmaceutico`;

CREATE TABLE `farmaceutico` (
  `cpf` varchar(11) NOT NULL,
  `crf` varchar(20) DEFAULT NULL,
  `data_inicio` datetime NOT NULL,
  PRIMARY KEY (`cpf`),
  CONSTRAINT `farmaceutico_ibfk_1` FOREIGN KEY (`cpf`) REFERENCES `funcionarios` (`cpf`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


  DROP TABLE IF EXISTS `fornecedor`;

CREATE TABLE `fornecedor` (
  `cnpj` varchar(15) NOT NULL,
  `nome` varchar(40) DEFAULT NULL,
  `endereÃ§o` varchar(100) DEFAULT NULL,
  `contato` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`cnpj`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


  DROP TABLE IF EXISTS `funcionarios`;

CREATE TABLE `funcionarios` (
  `cpf` varchar(11) NOT NULL,
  `name` varchar(50) DEFAULT NULL,
  `numero_carteira` varchar(20) DEFAULT NULL,
  `contato` varchar(50) DEFAULT NULL,
  `salario` float DEFAULT NULL,
  `birth_date` datetime NOT NULL,
  PRIMARY KEY (`cpf`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

  DROP TABLE IF EXISTS `gerente`;

CREATE TABLE `gerente` (
  `cpf_funcionario` varchar(11) NOT NULL,
  `login` varchar(30) DEFAULT NULL,
  `senha` varchar(20) DEFAULT NULL,
  `data_inicio` datetime NOT NULL,
  PRIMARY KEY (`cpf_funcionario`),
  CONSTRAINT `gerente_ibfk_1` FOREIGN KEY (`cpf_funcionario`) REFERENCES `funcionarios` (`cpf`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

```sql

  DROP TABLE IF EXISTS `microcontrollers`;

CREATE TABLE `microcontrollers` (
  `id` int NOT NULL,
  `ports` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `microcontrollers_ibfk_1` FOREIGN KEY (`id`) REFERENCES `devices` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


  DROP TABLE IF EXISTS `produtos`;

CREATE TABLE `produtos` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(50) DEFAULT NULL,
  `type` varchar(50) DEFAULT NULL,
  `sector` varchar(50) DEFAULT NULL,
  `current_price` float DEFAULT NULL,
  `available_quantity` int DEFAULT NULL,
  `batch_date` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `produtos_entrega`;

CREATE TABLE `produtos_entrega` (
  `id_produto` int NOT NULL,
  `cnpj_fornecedor` varchar(15) NOT NULL,
  PRIMARY KEY (`id_produto`,`cnpj_fornecedor`),
  KEY `cnpj_fornecedor` (`cnpj_fornecedor`),
  CONSTRAINT `produtos_entrega_ibfk_1` FOREIGN KEY (`id_produto`) REFERENCES `produtos` (`id`) ON DELETE CASCADE,
  CONSTRAINT `produtos_entrega_ibfk_2` FOREIGN KEY (`cnpj_fornecedor`) REFERENCES `fornecedor` (`cnpj`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `produtos_fornecedor`;

CREATE TABLE `produtos_fornecedor` (
  `id_produto` int NOT NULL,
  `cnpj_fornecedor` varchar(15) NOT NULL,
  PRIMARY KEY (`id_produto`,`cnpj_fornecedor`),
  KEY `cnpj_fornecedor` (`cnpj_fornecedor`),
  CONSTRAINT `produtos_fornecedor_ibfk_1` FOREIGN KEY (`id_produto`) REFERENCES `produtos` (`id`) ON DELETE CASCADE,
  CONSTRAINT `produtos_fornecedor_ibfk_2` FOREIGN KEY (`cnpj_fornecedor`) REFERENCES `fornecedor` (`cnpj`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

```sql

DROP TABLE IF EXISTS `reads`;

CREATE TABLE `reads` (
  `id` int NOT NULL AUTO_INCREMENT,
  `temperatura` float NOT NULL,
  `humidade` float NOT NULL,
  `date_time` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=39 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `roles`;

CREATE TABLE `roles` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(30) DEFAULT NULL,
  `description` varchar(512) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `sensors`;

CREATE TABLE `sensors` (
  `id` int NOT NULL,
  `measure` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `sensors_ibfk_1` FOREIGN KEY (`id`) REFERENCES `devices` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `user_roles`;

CREATE TABLE `user_roles` (
  `user_id` int NOT NULL,
  `role_id` int NOT NULL,
  PRIMARY KEY (`user_id`,`role_id`),
  KEY `role_id` (`role_id`),
  CONSTRAINT `user_roles_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),
  CONSTRAINT `user_roles_ibfk_2` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

```sql

DROP TABLE IF EXISTS `users`;

CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(30) NOT NULL,
  `email` varchar(30) NOT NULL,
  `password` varchar(1024) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DROP TABLE IF EXISTS `vendas`;

CREATE TABLE `vendas` (
  `id` int NOT NULL AUTO_INCREMENT,
  `data_venda` date NOT NULL,
  `id_produtos` int DEFAULT NULL,
  `cpf_cliente` varchar(11) DEFAULT NULL,
  `cpf_funcionario` varchar(11) DEFAULT NULL,
  `valor` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `id_produtos` (`id_produtos`),
  KEY `cpf_cliente` (`cpf_cliente`),
  KEY `cpf_funcionario` (`cpf_funcionario`),
  CONSTRAINT `vendas_ibfk_1` FOREIGN KEY (`id_produtos`) REFERENCES `produtos` (`id`),
  CONSTRAINT `vendas_ibfk_2` FOREIGN KEY (`cpf_cliente`) REFERENCES `clientes` (`cpf`),
  CONSTRAINT `vendas_ibfk_3` FOREIGN KEY (`cpf_funcionario`) REFERENCES `funcionarios` (`cpf`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Banco de dados da aplicação:

- Tabelas para armazenar todos os dados de clientes, funcionários, vendas, produtos, fornecedores, dispositivos.

- Tabela para registro das informações recebidas do sensor DHT22 em nossa sessão de IOT.

- Tabela de vendas se relacionam com as tabelas de produtos, clientes e funcionários fornecendo em nossa aplicação web detalhes destas 3 tabelas nos registros de vendas.

- Todas as tabelas armazenam dados para fazer a listagem nas páginas especificas.

Hardware: Na parte de hardware durante as aulas utilizamos os sensores de temperatura e umidade DHT11 e o microcontrolador ESP32 para conseguirmos realizar as leituras e trata-las em nossa aplicação Flask, além disso utilizamos também o sensor DHT22 simulado no site Wokwi, pois quando não tínhamos o acesso aos sensores o simulador era nossa única opção para testes, portanto nossa aplicação pode receber leituras vindas do sensor físico conectado ao computador e também do simulador Wokwi, dependendo de qual está conectada.

Foram utilizados neste projeto um microcontrolador ESP32 e um sensor de umidade DHT11.

Tópicos MQTT utilizados "/pharmhub/leitura" URL utilizada para recuperar a informação coletada pelo sensor e enviar para a aplicação para ser tratada e armazenada no banco de dados. E "/pharmhub/botao" URL utilizada para enviar mensagem ao hardware solicitando a ação do envio de dados.

Neste link estão a montagem do circuito utilizado no nosso projeto juntamente com o código para seu funcionamento:

https://wokwi.com/projects/367348873363621889

Integração: Durante a parte final do projeto foi onde integramos todas as funcionalidades necessárias para a aplicação, as páginas web todas funcionais, cada uma com sua função e estilizadas pelo framework Bootstrap, todos os bancos de dados criados pelo Flask e armazenando as informações registradas em inputs da interface web, na parte de hardware nosso sensor DHT22 simulado no WOKWI ou o sensor DHT11 físico, está fazendo todas as leituras necessárias para o monitoramento da temperatura e umidade do ambiente e enviando os dados para aplicação Flask ao comando de um botão que solicita os dados para a aplicação e posteriormente salva no banco de dados.

Conclusão: Nossa equipe conseguiu desenvolver a maior parte do projeto durante as aulas disponibilizadas pelos professores, também utilizando as tecnologias que foram ensinadas pelos mesmos, mantivemos a base da ideia do projeto a mesma desde o início e conseguimos finalizar, dentro do prazo, todas as funcionalidades necessárias para a entrega da aplicação.

Também é importante ressaltar que este contato com IOT junto com aplicações web foi bastante desafiador, mas foi muito bem aproveitado pela minha equipe, durante o projeto conseguimos aprender muito sobre o próprio Flask, um pouco sobre hardware, bastante sobre o framework de design Bootstrap, também aprendemos bastante sobre a conexão do IOT com outras aplicações e como uma equipe aprendemos a nos organizar e dividir tarefas para consegui entregar tudo no prazo.

Relatório: Metodologia Ágil Scrum TDE – 1

Introdução: Este relatório tem como objetivo relatar a experiência da equipe ao utilizar a metodologia ágil Scrum no projeto da disciplina de Experiencia criativa. Vão ser abordados nesse relatório os principais pontos da metodologia, a aplicação prática no projeto e os motivos que levaram à escolha do Scrum como metodologia ágil.

Metodologia Ágil Scrum é uma metodologia ágil de gerenciamento de projetos que enfatiza a colaboração, transparência, adaptação e entrega contínua de atividades. Baseado em ciclos chamados de sprints, o Scrum promove uma abordagem flexível e adaptável para o desenvolvimento de software e outros projetos mais complexos.

Aplicação do Scrum no Projeto: Durante a execução do projeto, a equipe adotou o Scrum como metodologia de gerenciamento e aplicou seus principais princípios e práticas. Isso incluiu a separação de tarefas entre os membros do grupo durante nossas reuniões semanais durante as aulas de uma maneira em que todos pudessem contribuir para o projeto sabendo executar todas as partes, desde o front-end ao back-end além da gerência do banco de dados e a parte de hardware (IOT).

Adotamos os sprints em todas as aulas que foram disponibilizadas para a execução do projeto para conseguirmos realizar o projeto nos períodos gerenciáveis das aulas, onde contaríamos com o auxílio dos professores e também fizemos algumas reuniões fora do horário de aula para dividirmos e acelerarmos certa parte do projeto. Durante os sprint, foram realizadas reuniões para discutir o progresso, identificar impedimentos e planejar o que deveria ser entregue em cada item solicitado pelos professores.

Resultados e Benefícios da Metodologia Scrum: A adoção do Scrum trouxe diversos benefícios para o projeto, incluindo:

a) Transparência e Visibilidade: As divisões de trabalho e também a colaboração entre os membros proporcionaram maior visibilidade sobre o progresso do projeto. Todos os membros da equipe tinham uma compreensão clara do trabalho em andamento e do que seria executado na próxima aula disponibilizada para realizarmos o projeto.

b) Adaptabilidade e Flexibilidade: O Scrum nos permitiu nos adaptarmos a todos os requisitos apresentados pelos professores e nos ajudou também a nos organizarmos para realizarmos as entregas de todas as etapas dos PJBL e PBL.

c) Entrega Contínua de Resultados: Ao final de cada sprint (aula), o projeto ia sendo incrementado. Isso permitiu obter uma visão sobre oque estava sendo feito e se estava se adequando aos requisitos, pois em praticamente toda aula disponibilizada para a execução do projeto uma nova funcionalidade era acrescentada.

d) Colaboração e Comunicação: O Scrum permite a colaboração e a comunicação entre os membros da equipe. As reuniões ajudaram a promover um ambiente colaborativo e a identificar e resolver problemas de forma rápida.

Motivos para a escolha do Scrum: A equipe optou pela metodologia Scrum devido a diversos fatores, entre eles:

a) Complexidade do Projeto: O projeto apresentava um certo grau de complexidade, pois era necessário muita organização e atenção para a realização de todas as partes, em especial a criação de todos os bancos de dados com o SQLALCHEMY. O Scrum, foi escolhido para lidarmos com essa complexidade e chegarmos a uma solução.

b) Facilidade para usar: A metodologia Scrum foi escolhida também pela facilidade em utilizá-la, com esta metodologia não foi necessário utilizar nada além da organização entre os membros e anotações sobre oque iria ser realizado por quem, facilitando assim nosso trabalho durante as aulas para dar continuidade ao projeto.

c) Consenso da equipe: Todos os membros da equipe concordaram na utilização do Scrum para a realização do projeto, pois concordamos que seria mais fácil utilizar esta metodologia por ser mais simples e combinar com a nossa forma de organizar os projetos que consiste em sentar juntos para discutir ideias e dividir o trabalho sempre se ajudando para que o aprendizado seja bem aproveitado por todos.

d) Flexibilidade para Mudanças: O Scrum é usado por sua flexibilidade em relação a mudanças de requisitos ou prioridades. Essa capacidade de adaptação foi considerada pela equipe, pois inicialmente estávamos em dúvidas sobre a parte do IOT e não sabíamos se futuramente teríamos que encaixar alguma mudança no projeto.

Conclusão: A experiência de utilizar a metodologia Scrum no projeto foi altamente positiva. A equipe obteve benefícios significativos, como maior visibilidade, adaptabilidade, entrega contínua de resultados e colaboração efetiva de todas as partes da equipe em diversas áreas do projeto para não deixar uma coisa na mão de cada um diminuindo assim a nossa aprendizagem durante o projeto.

A escolha do Scrum foi justificada pela complexidade do projeto, a facilidade para utilizar, o consenso da equipe e a necessidade de flexibilidade para lidar com mudanças futuras que poderiam ocorrer.

Utilizaremos futuramente o Scrum em outros projetos com tal nível de complexidade, pois para nós foi uma experiencia que nos ajudou muito a entender como projetos assim são desenvolvidos e nos mostrou a necessidade de organização para a realização de atividades deste tipo.

No início das aulas onde começamos o projeto decidimos realizar os sprints nas aulas que foram disponibilizadas para a execução dele, assim deixando as aulas mais produtivas e tentando não deixar muita coisa para realizar em casa, até mesmo por causa de algumas bibliotecas e outras coisas que não estavam rodando perfeitamente nas maquinas pessoais de alguns membros do grupo sendo melhor utilizarmos as maquinas da PUC, graças ao Scrum conseguimos nos organizar de uma maneira que a maior parte do projeto foi desenvolvida em sala e conseguimos assim finalizar o projeto com o que foi requisitado.

Relatório: Frameworks de Design e Vantagens do Framework Bootstrap TDE – 2

Introdução: Existem vários frameworks de design disponíveis para os desenvolvedores web, que fornecem componentes, estilos e padrões predefinidos para agilizar o processo de design e garantir uma aparência e experiência consistentes em todo o site.

Este relatório tem como objetivo analisar as razões pelas quais o framework Bootstrap (que foi o escolhido pela nossa equipe para o projeto) é amplamente utilizado no desenvolvimento web e destacar suas principais vantagens.

O Bootstrap é um framework de código aberto que fornece uma estrutura e conjunto de ferramentas para a criação de interfaces web responsivas e estilizadas. Lançado inicialmente pelo Twitter, o Bootstrap se tornou um dos frameworks mais populares para o desenvolvimento web, sendo adotado por uma ampla comunidade de desenvolvedores em todo o mundo.

Motivos do porquê o Bootstrap foi selecionado pela nossa equipe:

a) Facilidade de uso: Uma das principais vantagens do Bootstrap é a sua facilidade de uso. Com uma documentação abrangente e clara, o Bootstrap oferece uma grande quantidade de componentes pré-projetados, como botões, formulários, tabelas e menus, que podem ser facilmente integrados às páginas web, inclusive facilitando muito na criação das tabelas de registros da nossa aplicação. Isso economizou muito do nosso tempo na hora de desenvolver o front-end.

b) Responsividade: A crescente de dispositivos móveis fez com que a responsividade se tornasse uma característica crucial para os sites modernos, por mais que nossa aplicaçao não tenha esse foco é algo importante de ressaltar. O Bootstrap é conhecido por sua capacidade de criar interfaces web responsivas de maneira eficiente. Ele possui uma grade flexível e fluida que se adapta automaticamente aos diferentes tamanhos de tela, garantindo que o site seja exibido de maneira adequada em todos os dispositivos.

c) Consistência e padronização: Outra vantagem significativa do Bootstrap é sua abordagem consistente e padronizada para o design e a estruturação de elementos da interface. O framework oferece uma aparência visualmente agradável e profissional, permitindo que os desenvolvedores criem sites com uma estética coesa e harmoniosa, o que nos permitiu deixar as tabelas de cadastro iguais umas às outras.

d) Comunidade ativa e recursos adicionais: O Bootstrap possui uma comunidade ativa de desenvolvedores e usuários que contribuem constantemente com atualizações, melhorias e recursos adicionais. Essa comunidade é uma valiosa fonte de suporte, tutoriais, exemplos e plugins personalizados que podem ser utilizados para estender as funcionalidades do Bootstrap. Através dessa vasta comunidade, os desenvolvedores têm acesso a recursos e soluções para resolver problemas comuns e melhorar suas habilidades no desenvolvimento com o framework este fato nos auxiliou muito, pois as informações sobre ele na internet facilitaram muito a correção de erros e a produção do nosso front-end.

Conclusão: O framework Bootstrap é amplamente utilizado na indústria de desenvolvimento web devido às suas vantagens significativas. Sua facilidade de uso, capacidade de resposta, consistência e apoio da comunidade são fatores que nos fizeram seleciona-lo. O Bootstrap nos ajudou a economizar tempo e forneceu uma base sólida para a criarmos uma interface web moderna e visualmente atraente.

Por ser de fácil utilização e conter uma grande quantidade de templates feitos na internet o Bootstrap é uma ferramenta que nos ajudou a economizar muito tempo durante a construção da nossa aplicação.

Relatório: Vantagens do uso de NoSQL em IOT e a escolha do MySQL TDE - 3

Introdução: Os bancos de dados NoSQL (Not Only SQL) são uma categoria de bancos de dados que diferem dos bancos de dados relacionais tradicionais. Eles foram projetados para lidar com requisitos de armazenamento e recuperação de dados em grande escala, alta disponibilidade, escalabilidade horizontal e flexibilidade de esquema.

Definição dos bancos de dados NoSQL:

Os bancos de dados NoSQL são projetados para lidar com grandes volumes de dados e cenários de alta velocidade.

Eles não seguem o modelo de tabela e esquema rígido dos bancos de dados relacionais.

Os bancos de dados NoSQL geralmente adotam modelos de dados flexíveis, como documentos, chave-valor, colunas amplas e grafos.

Eles são altamente escaláveis, oferecendo a capacidade de distribuir dados em vários servidores para suportar cargas de trabalho intensivas.

Principais diferenças entre bancos de dados NoSQL e relacionais:

a) Modelo de dados: Os bancos de dados relacionais seguem um modelo tabular com esquema fixo, enquanto os NoSQL oferecem modelos flexíveis, como documentos, chave-valor, colunas amplas e grafos.

b) Escalabilidade: Os bancos de dados relacionais são projetados principalmente para escalar verticalmente (adicionar recursos a um único servidor), enquanto os NoSQL são projetados para escalabilidade horizontal (adicionar mais servidores à infraestrutura).

c) Consistência: Os bancos de dados relacionais geralmente enfatizam a consistência forte dos dados, enquanto os NoSQL oferecem modelos com consistência eventual, permitindo maior disponibilidade e tolerância a falhas.

d) Transações: Os bancos de dados relacionais suportam transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), enquanto os NoSQL geralmente priorizam a escalabilidade e a velocidade, oferecendo garantias de consistência mais relaxadas.

Vantagens dos bancos de dados NoSQL:

a) Escalabilidade horizontal: Os bancos de dados NoSQL são altamente escaláveis, permitindo distribuir dados em vários servidores para suportar cargas de trabalho intensivas.

b) Flexibilidade de esquema: Os modelos de dados NoSQL flexíveis permitem a adaptação dinâmica dos esquemas, tornando mais fácil lidar com dados sem uma estrutura rígida.

c) Desempenho: Os bancos de dados NoSQL são otimizados para consultas rápidas e operações de gravação em grande escala.

d) Disponibilidade: A arquitetura distribuída dos bancos de dados NoSQL oferece alta disponibilidade, pois os dados são replicados entre vários servidores.

Desvantagens dos bancos de dados NoSQL:

a) Menor suporte a consultas complexas: Alguns bancos de dados NoSQL podem ter limitações na realização de consultas complexas que envolvem várias relações e agregações.

b) Menor maturidade: Comparados aos bancos de dados relacionais, os NoSQL são uma tecnologia relativamente nova, o que significa que podem ter menos ferramentas, recursos e comunidades estabelecidas.

c) Consistência eventual: Embora a consistência eventual possa ser adequada para muitos cenários, existem casos em que a consistência forte dos bancos de dados relacionais é necessária.

Tipos de bancos de dados NoSQL:

A) Bancos de dados de documentos: Armazenam dados em formato de documento, como JSON ou XML.

b) Bancos de dados de chave-valor: Armazenam dados como pares de chave e valor

c) Bancos de dados de colunas amplas: Armazenam dados em colunas, permitindo consultas eficientes em grandes conjuntos de dados.

d) Bancos de dados de grafos: Armazenam dados na forma de nós e arestas, sendo adequados para modelar relacionamentos complexos.

Vantagens do uso de NoSQL em IOT:

O Internet of Things (IOT) é um conceito que se refere à conexão e comunicação entre dispositivos físicos, objetos e sistemas de informação por meio da internet. Com o rápido crescimento do IOT, há uma necessidade crescente de gerenciar e processar grandes volumes de dados gerados por esses dispositivos. Nesse contexto, os bancos de dados NoSQL surgiram como uma solução alternativa ao modelo relacional tradicional, como o MySQL.

As vantagens da utilização deste modelo são: A escalabilidade horizontal, flexibilidade de modelo de dados, alta velocidade de leitura e gravação e tolerância a falhas.

Principais desafios na utilização do NoSQL em IOT:

a) Modelagem de dados: A escolha e a estruturação adequada dos dados em um modelo flexível é essencial para atender aos requisitos específicos da aplicação IOT.

B) Consistência: A garantia da consistência dos dados em tempo real pode ser um desafio, pois os bancos de dados NoSQL oferecem diferentes níveis de consistência.

C) Escalabilidade: O gerenciamento eficiente do grande volume de dados gerados pelos dispositivos IOT é fundamental para garantir a escalabilidade do banco de dados NoSQL.

D) Segurança: A proteção dos dados confidenciais ou críticos requer a implementação de recursos de segurança adequados, como autenticação, criptografia e controle de acesso.

E) Integração com outras tecnologias: A integração eficiente do banco de dados NoSQL com outras tecnologias e componentes, como plataformas de gerenciamento de dispositivos e serviços em nuvem, é um desafio importante a ser considerado.

Por que escolhemos utilizar o MySQL:

Facilidade de uso: escolhemos o MySQL pela sua facilidade de uso além de já estarmos familiarizado com suas funções devido as aulas do semestre passado, além de quem alguns bancos de dados NoSQL não estavam rodando com estabilidade nas máquinas da PUC então selecionamos o MySQL pela sua confiabilidade e também por nossa experiencia anterior com o uso dessa ferramenta.

Conclusão: O uso de bancos de dados NoSQL em IOT traz diversas vantagens, como escalabilidade horizontal, flexibilidade de modelo de dados, alta velocidade de leitura e gravação, e tolerância a falhas.

No entanto, utilizamos o MySQL pela sua facilidade de uso. É importante ressaltar que a escolha entre NoSQL e MySQL dependerá das características e requisitos específicos de cada projeto de IOT.

Relatório de Utilização do Flask-MQTT para Integração do Sensor DHT11 e DHT22 no Projeto TDE – 4

Introdução: Este relatório descreve a utilização do Flask-MQTT para a integração do sensor DHT22 (simulado no site https://wokwi.com/) e do sensor DHT11 (disponibilizado apenas durante as aulas) em nosso projeto. O Flask-MQTT é uma extensão do Flask que facilita a implementação de um servidor MQTT para comunicação com dispositivos IOT.

Objetivo: O objetivo principal do uso do Flask-MQTT foi possibilitar a integração do sensor ao nosso projeto, permitindo a coleta de dados de temperatura e umidade e a sua transmissão em tempo real para a nossa aplicação web onde o usuário poderia monitorar a situação em que os remédios da farmácia estão armazenados através do apertar de um botão na interface web.

Resultados: A utilização do Flask-MQTT para a integração do sensor em nosso projeto finalizou a parte de IOT. Conseguimos estabelecer a comunicação entre o sensor e a aplicação web, transmitindo os dados de temperatura e umidade em tempo real. Os resultados foram exibidos na interface web, permitindo o monitoramento contínuo das condições em que os remédios do usuário estão armazenados sempre que o usuário solicitar.

Conclusão: O uso do Flask-MQTT possibilitou a integração do sensor DHT22 e DHT11 em nosso projeto. Através dessa extensão do Flask, conseguimos implementar de forma eficiente um servidor MQTT e estabelecer a comunicação entre o sensor e a aplicação web. Com isso, alcançamos nosso objetivo de coletar e exibir os dados de temperatura e umidade em tempo real, permitindo um monitoramento eficaz das condições da área de armazenamento.

Ferramentas utilizadas: Utilizamos o site https://wokwi.com/ para simular o nosso sensor de temperatura e umidade, pois como não tínhamos acesso a eles a não ser durante as aulas utilizamos o simulador, mas o sensor físico também foi uma opção funcional do nosso projeto, coletando dados e passando-os para a nossa aplicação.

Grupo: Henrique Zan Grande, João Gabriel Pitol, Lucas Braga, Rafael Vargas.