UNIFEOB
ESCOLA DE NEGÓCIOS

2023

PROJETO INTEGRADO

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

A.D.S. E CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

IOT DATA STREAMER

<UNIFEOB>

SÃO JOÃO DA BOA VISTA, SP

NOVEMBRO 2023

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

A.D.S. E CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

IOT DATA STREAMER

<UNIFEOB>

MÓDULO MODELAGEM E DESENVOLVIMENTO DE SISTEMAS

Gestão Financeira – Profa. Renata Elizabeth de Alencar Marcondes

Programação Orientada a Objeto – Prof. Nivaldo Andrade

Lógica de Programação – Prof. Marcelo Ciacco de Almeida

Modelagem de Dados – Prof. Max Streicher Vallim

Projeto de Modelagem e Desenvolvimento de Sistemas – Profª. Mariângela Martimbianco Santos

Estudantes:

Igor Garcia Nunes de Araujo , RA 23000093

Igor Jose Ananias Campos, RA 23000131

João Pedro Pizoli Carvalho, RA: 23001145

João Vitor Marques, RA 23001036

Luis Fernando Menezes Ferreira Leite, RA 23000492

Vinicius Felipe Tesser, RA 23000778

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2023

SUMÁRIO

ISSN 1983-6767

1. INTRODUÇÃO

A JIVLTECH é uma empresa de tecnologia especializada em desenvolver soluções inovadoras para a gestão eficiente de dados ambientais. Composta por estudantes de Análise e Desenvolvimento de Sistemas e Ciência da Computação pela UNIFEOB, a JIVLTECH está comprometida em automatizar processos e contribuir para a sustentabilidade.

Nosso projeto central concentra-se no desenvolvimento de uma Interface de Programação de Aplicações (API) destinada a coletar e fornecer dados ambientais. Nossa API tem como principal finalidade coletar informações provenientes de sensores ambientais e disponibilizá-las aos usuários cadastrados no sistema. Este sistema é fundamentado em um sólido mecanismo de registro e autenticação, implementado em um servidor MySQL, assegurando a integridade e a confidencialidade dos dados.

Para a construção do Front-end da nossa API, adotamos uma abordagem centrada na experiência do usuário, utilizando HTML para marcação, CSS para estilização e JavaScript em conjunto com o Node.js para desenvolver funcionalidades dinâmicas e estabelecer a comunicação eficiente com o banco de dados (MySQL).

O principal objetivo da nossa API é oferecer um controle ágil e eficiente sobre os dados ambientais, direcionando informações relevantes aos usuários cadastrados. Nossa API visa otimizar a gestão de dados ambientais, contribuindo para a tomada de decisões informadas e promovendo práticas sustentáveis.

ISSN 1983-6767

2. DESCRIÇÃO DA EMPRESA

A UNIFEOB cujo CNPJ é 59.764.555/0001-52, e está localizada na Av. Dr. Otávio da Silva Bastos, 2439. Foi o local escolhido para a implementação do projeto da JIVLtech intitulado E-Trash. A UNIFEOB, sigla para Centro Universitário Fundação de Ensino Octávio Bastos, é uma instituição de ensino superior localizada na cidade de São João da Boa Vista, no estado de São Paulo, Brasil. A faculdade oferece uma variedade de cursos de graduação, pós-graduação e extensão, abrangendo áreas como ciências humanas, ciências exatas, ciências da saúde e ciências agrárias.

A UNIFEOB é reconhecida pelo Ministério da Educação (MEC) e possui uma estrutura moderna e bem equipada, com laboratórios, biblioteca, salas de aula e espaços para atividades práticas e esportivas. A instituição tem como objetivo proporcionar uma formação acadêmica de qualidade, aliando teoria e prática, para preparar os estudantes para o mercado de trabalho e para a cidadania.

Além disso, a UNIFEOB investe em projetos de pesquisa e extensão, estimulando a produção científica e o engajamento dos alunos com a comunidade. A faculdade também promove eventos, palestras e atividades culturais, visando o desenvolvimento integral dos estudantes.

Em resumo, a UNIFEOB é uma instituição de ensino superior bem estabelecida, comprometida com a excelência acadêmica, a formação profissional e o desenvolvimento social de seus alunos.

ISSN 1983-6767

3. PROJETO INTEGRADO

3.1 PROGRAMAÇÃO ORIENTADA A OBJETO

A programação orientada a objetos (POO) é um paradigma que organiza o código em objetos, encapsulando dados e comportamentos. Como mencionado no livro "Programming: Principles and Practice Using C++" Stroustrup,Bjarne, a POO visa facilitar a reutilização de código, promover a manutenção eficiente do software e aumentar a modularidade, tornando-o mais flexível.

3.1.1 CLASSES E OBJETOS

No projeto da lixeira E-trash utilizamos classes e objetos para construção do back-end de um software que liga a lixeira ao desktop. No que diz respeito a classes, foram criadas duas: Lixeira e Usuário.

A classe Lixeira foi criada como atributo os compartimentos que a lixeira contém (metal,vidro,plástico e orgânico), e com uma função denominada ‘LixeiraCheiaVazia’, que identifica se os compartimento da lixeira estão cheios ou vazios e avisam quais estão.

Juntamente com a classe Lixeira, foi criada a classe Usuario, que tem como atributos o nome, email e senha dos usuários. Na classe Usuario contamos com duas funções: Login e Registro. A função Login verificará se os emails e senhas inseridos batem com os registrados no banco de dados, já a função Registro tem como papel registrar as informações de cada novo usuário.

3.1.2 ATRIBUTOS, MÉTODOS, ENCAPSULAMENTO, HERANÇA E POLIMORFISMO.

Atributos são características de uma classe, como cor ou tamanho. Métodos são ações que a classe pode fazer, como "mover" ou "mudar de cor". Encapsulamento é proteger essas características, permitindo que só certas ações as alterem. Herança é quando uma classe pode herdar características de outra, facilitando a criação de novas classes com base em existentes.

Polimorfismo é a capacidade de tratar diferentes classes de maneira similar, simplificando o uso delas de maneira uniforme. Esses conceitos são fundamentais em programação orientada a objetos, tornando o código mais organizado, reutilizável e fácil de entender.

No projeto da lixeira, utilizamos diferentes atributos, como o nome, o email e a senha, que são os atributos relacionados a classe Usuário. E metal,plástico,orgânico e vidro, que são os atributos relacionados a classe Lixeira.

Referente aos métodos, utilizamos os métodos Registro (responsável por solicitar e armazenar o nome, email e senha dos novos usuários) e o Login (responsável por solicitar e averiguar se o email e a senha digitados pelo usuário correspondem aos registrados) na classe Usuario. E o método “LixeiraCheiaVazia”, que é responsável por monitorar e avisar quando cada compartimento da lixeira estiver cheio ou vazio.

Quanto ao encapsulamento, herança e polimorfismo. Não achamos necessário a utilização dos mesmos para a criação do software da lixeira. Pois utilizando apenas classes, atributos e métodos, fomos capazes de construir todo o back-end.

3.1.3 MÉTODOS ESTÁTICOS, PÚBLICOS E PRIVADOS

Métodos estáticos pertencem à classe e são invocados sem criar instâncias. Métodos públicos são acessíveis externamente à classe, facilitando a interação. Métodos privados são restritos à própria classe, garantindo encapsulamento e controle interno.

No nosso código, utilizamos o atributo público para armazenar o nome, o atributo privado para armazenar o email e o protegido para armazenar as senhas, todos ligados à classe Usuário. Já na classe lixeira utilizamos o atributo público para todas as variáveis da mesma.

3.2 LÓGICA DE PROGRAMAÇÃO

A programação lógica é um paradigma de programação que se baseia na lógica matemática. “McCarthy, John [1958] foi o primeiro a publicar uma proposta de uso da lógica matemática para programação”, A lógica é a maneira como organizamos e definimos os passos a serem seguidos em cada momento para executar uma ação.

A lógica de programação é composta por um conceito crucial, denominado algoritmo que é um conjunto de pequenas etapas que devem ser seguidas a risca em prol de alcançar uma meta ou objetivo pré estipulado, por exemplo: se eu tenho como objetivo escovar os

dentes, eu devo primeiro pegar a escova, pegar a pasta, abrir a tampa da pasta, coloca-lá na escova, abrir a boca e então poderemos concluir o nosso objetivo que é escovar os dentes. O algoritmo é algo imprescindível em qualquer projeto, e é a base da criação e do controle do mesmo. Ela foi o alicerce da construção do software da E-Trash.

Após compreender como funciona a lógica por trás dos computadores, é necessário compreender os princípios fundamentais para o desenvolvimento de software. Utilizaremos a linguagem de programação JavaScript como base. um dos conceitos mais importantes que temos são as estruturas condicionais, ou seja, uma estrutura de decisão. Dependendo se a condição for verdadeira ou não, executará um código específico. Suponha que, num dia chuvoso, sua mãe solicita que, caso chova, tire as roupas do varal. Se não chover, pode deixá-las lá, essa seria uma condição.

Outro ponto crucial no desenvolvimento são as funções, que são extremamente relevantes e úteis. São como se fossem tarefas a serem cumpridas. Dentro das funções, colocamos código que executa uma ação específica e podemos acionar as funções quando necessário.

Os operadores lógicos e de comparação são muito utilizados para criar condições para tomar decisões, por exemplo: com os operadores lógicos, é possível verificar se duas condições são verdadeiras ao mesmo tempo, ou vice-versa, se apenas uma for verdadeira. Com os operadores de comparação, é possível comparar os valores passados. Por exemplo: o usuário digita o seu endereço de e-mail, com a utilização dos operadores de comparação, pode-se verificar se este endereço está cadastrado em no banco de dados.

Outro conceito fundamental para ocorrer tudo bem no final, é prototipação, é de extrema importância para que o software saia como o planejado, se consiste em criar uma versão inicial do software, geralmente de forma simplificada, para testar e validar, é uma etapa muito importante principalmente na parte das interfaces já que cada usuário acessa as interface de um dispositivo diferente.

Último dos conceitos que fundamentais para o desenvolvimento de um software, são os banco de dados, na criação do software da E-Trash foi utilizado banco de dados relacional (MySQL), que são sistemas de gerenciamento que organizam as informações em forma de tabelas, onde cada tabela representa as entidades, e as colunas as informações que contém na tabela.

Todos os conceitos abordados são fundamentais para o desenvolvimento de um bom software, e aprender de fato como funciona a lógica por trás dos computadores.

3.2.1 CONCEITOS FUNDAMENTAIS DO DESENVOLVIMENTO DE SOFTWARE

Segundo John McCarthy (1958) “A programação lógica é um paradigma de programação que se baseia na lógica matemática.” McCarthy foi o primeiro a publicar uma proposta de uso da lógica matemática para programação. A lógica é a maneira como organizamos e definimos os passos a serem seguidos em cada momento para executar uma ação. A lógica de programação é composta por um conceito crucial, denominado algoritmos. Consiste em um conjunto de instruções lógicas que permitem que o computador realize uma tarefa específica.

Após aprender a base precisa-se compreender os conceitos fundamentais no desenvolvimento de software, como base utilizaremos a linguagem JavaScript, um dos conceitos mais importantes que temos são as estruturas condicionais, ou seja, uma estrutura de decisão. Dependendo se a condição for verdadeira ou não, executará um código específico. Suponha que, num dia chuvoso, sua mãe solicita que, caso chova, tire as roupas do varal. Se não chover, pode deixá-las lá, essa seria uma condição.

Prosseguindo com os conceitos, as variáveis são como uma caixinha onde pode-se armazenar determinados valores, que podem ser usados durante a execução do programa. As variáveis possuem nomes únicos e tipos de dados, o que corresponde ao tipo de dados armazenados dentro dessa variável, como números, textos e até mesmo valores verdadeiros ou falsos.

Outro ponto crucial no desenvolvimento são as funções, que são extremamente importantes e úteis. São como se fossem tarefas a serem cumpridas. Dentro das funções, colocamos código que executa determinada ação e podemos chamá-las quando necessário.

Último dos conceitos fundamentais que são os operadores, eles funcionam como os operadores matemáticos, utilizamos ele para fazer uma soma, subtração, divisão e multiplicação são muitos úteis para o desenvolvimento de um software.

Esses princípios são fundamentais para criar uma base sólida para o desenvolvimento de um software e criar soluções para problemas computacionais.

3.2.2 DESENVOLVIMENTO DE APLICAÇÕES DESKTOP

O desenvolvimento de aplicações para desktop consiste na criação de um software que se concentra em rodar em computadores, permitindo criar softwares poderosos e versáteis,

que podem ser utilizados em diversas áreas, como jogos, aplicativos de produtividade e sistemas de gerenciamento como no caso da lixeira E-Trash.

No desenvolvimento do software da lixeira E-Trash, foi criado interfaces de usuários muito intuitivas e acessíveis para qualquer tipo de usuário, visando que as interfaces têm um papel crucial na experiência do usuário, foi priorizado a facilidade de uso, a acessibilidade e a satisfação do usuário ao desenvolver uma interface intuitiva, com descrições claras e botões indicativos.

Foi utilizado no desenvolvimento do software, também a manipulação de arquivos, com ela, é possível ler informações de um arquivo, escrever novos dados nele, modificar seu conteúdo e até mesmo apagá-lo quando necessário, utilizamos para acessar e armazenar informações importantes para o funcionamento do software.

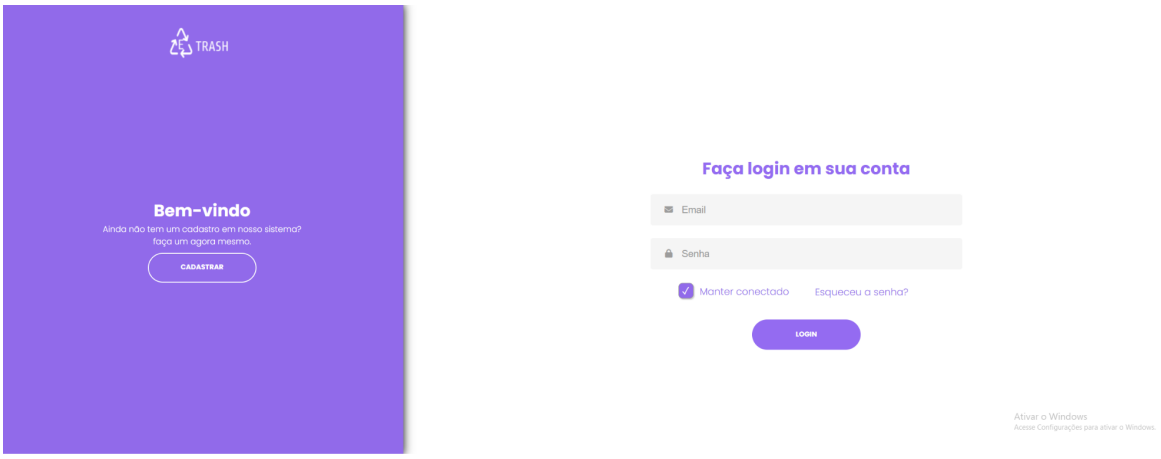
Dentro da criação do software utiliza-se um conceito que vem da programação orientada a objetos (POO), que é o empacotamento, que nos permite agrupar diversas classes e módulos num único local. É uma ótima maneira de organizar os códigos.

Um ponto crucial é a conexão e a operação com o banco de dados permitem que o software seja eficiente, uma vez que é através dele que podemos registrar informações, alterá-las e obter estas informações. No software da E-Trash, essa operação é sempre usada, desde o básico, quando o usuário se cadastra, até mesmo para verificar a quantidade de resíduos que está depositado na lixeira.

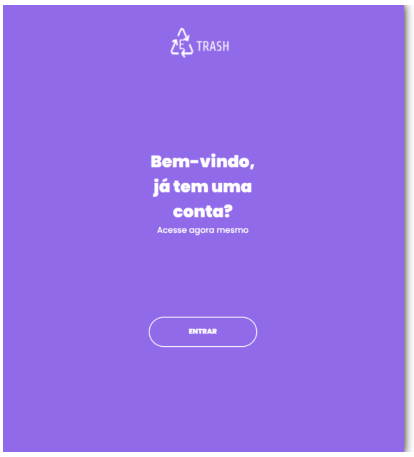
Todos estes pontos abordados são de extrema importância para criar-se uma aplicação robusta.

Segue abaixo: As interfaces que foram criadas, juntamente com suas descrições:

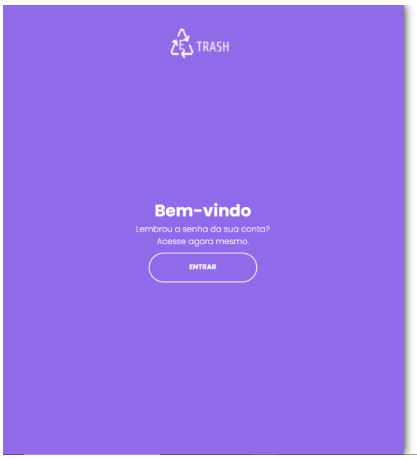
Página onde o usuário pode realizar o login:



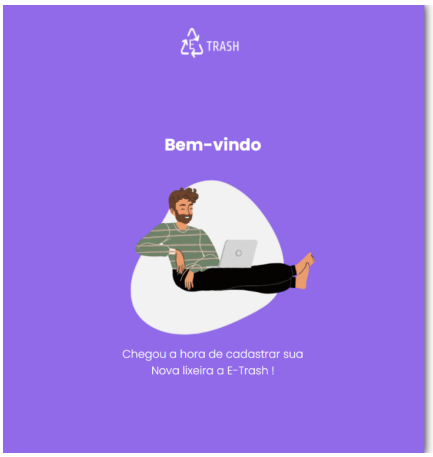
Página onde o usuário pode realizar o cadastramento:



Página de recuperação de senha:

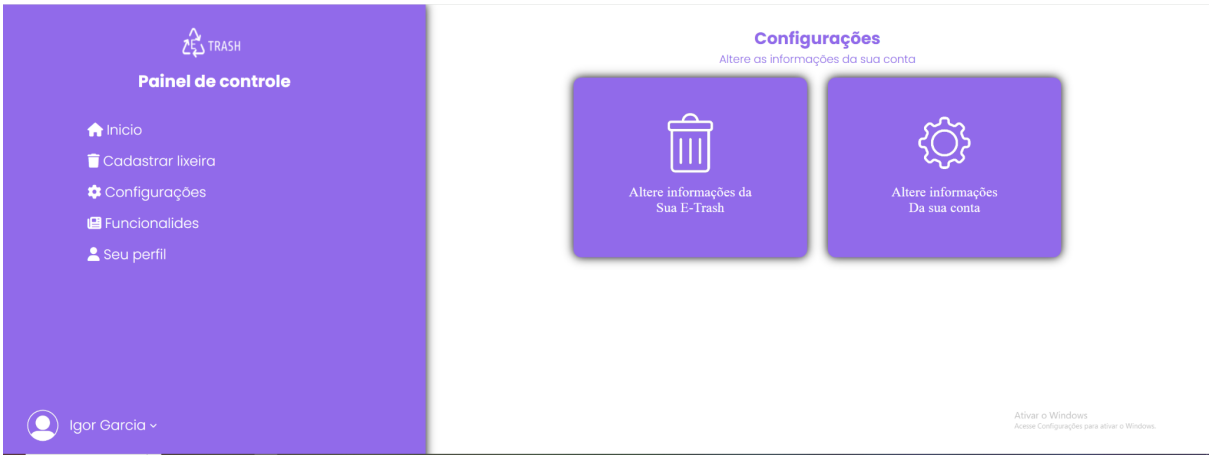


Página após o cadastramento de usuário:

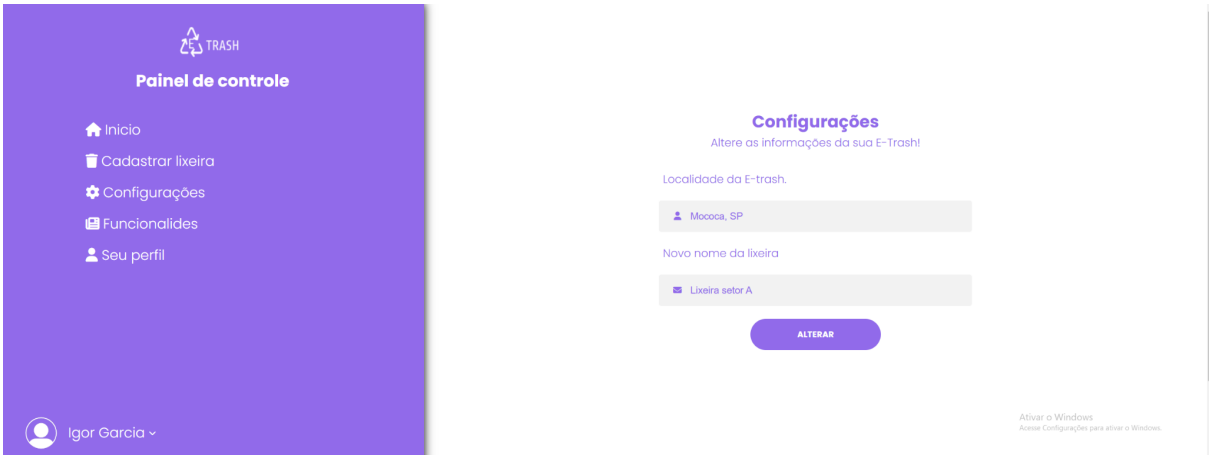


Tela de controle:

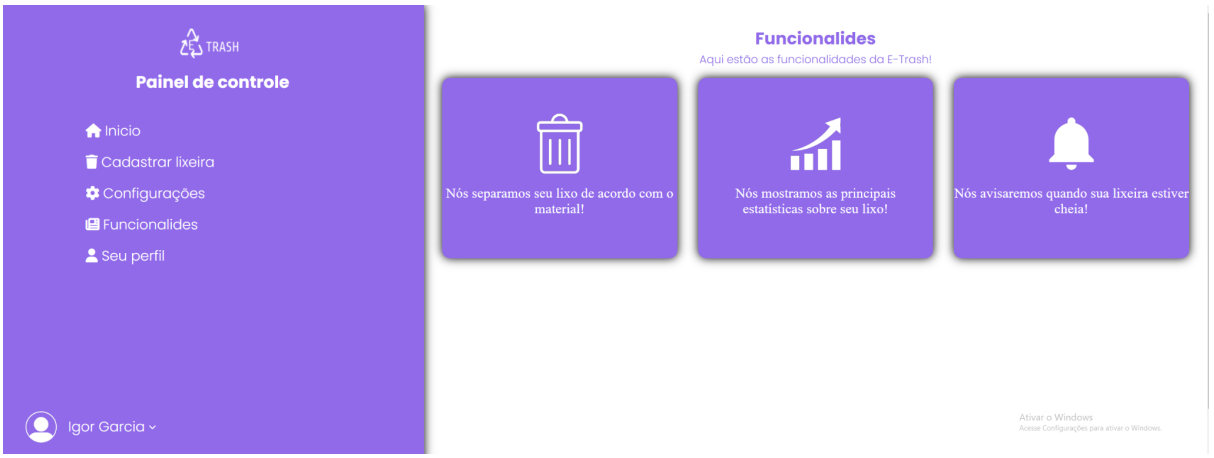
Configurações:



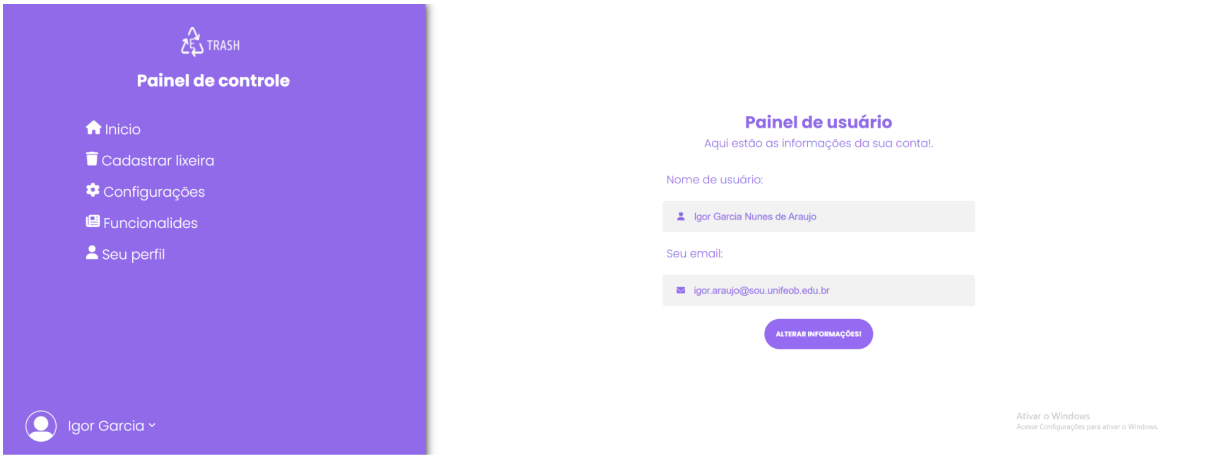
Alterar configurações da E-Trash:



Funcionalidades:



Perfil de usuário:



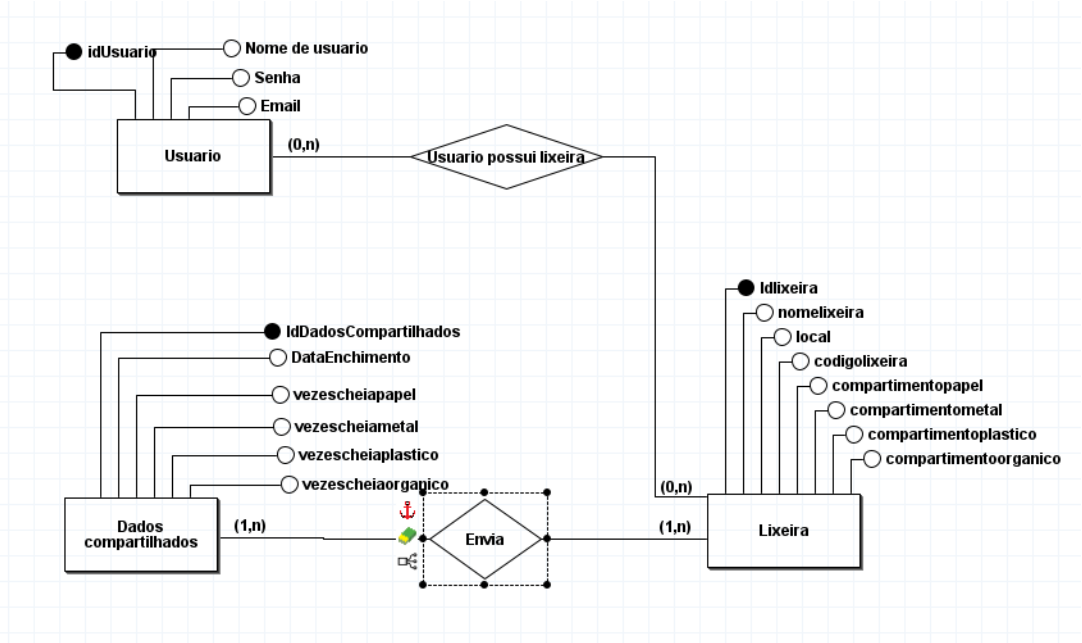
.

3.3 MODELAGEM DE DADOS

Segundo a Amazon Web Series(2023): “Modelagem de dados é o processo de criar uma representação visual, ou esquema, que define os sistemas de coleta e gerenciamento de informações de qualquer organização.” Nesta etapa do projeto, foram feitos alguns modelos para finalmente chegar em um banco de dados que pode armazenar as informações sobre a E-trash, começando com o modelo conceitual e no modelo lógico onde as informações podem ser visualizadas para que em seguida no modelo físico onde essas informações serão serão armazenadas e utilizadas em um sistema.

3.3.1 MODELO CONCEITUAL

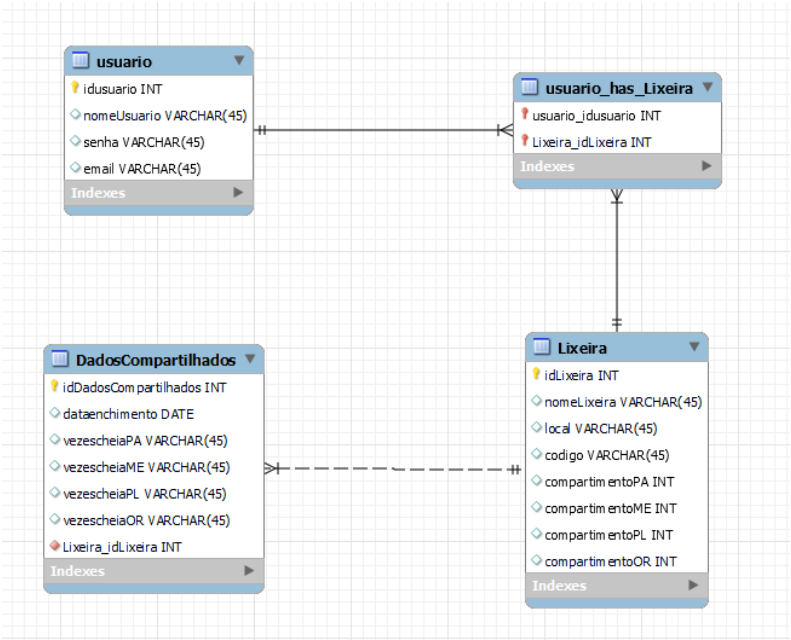
Com esse modelo conceitual, buscamos deixar bem claro as informações que serão armazenadas e suas conexões que serão feitas no aplicativo do MySQL Workbench. É possível ver 3 tables que são os objetos onde serão armazenados os dados, sendo elas "Usuário", onde será armazenado as informações sobre o usuário, "Lixeira", que irá guardar as informações sobre a lixeira e seus compartimentos e por fim "Dados Compartilhados" que será responsável por armazenar os dados dos enchimentos de cada compartimento e a data de enchimento.



3.3.2 MODELO LÓGICO E FÍSICO

Neste modelo lógico, através do MySQL fizemos uma representação gráfica dos requisitos de informação que serão coletadas através do nosso aplicativo da E-Trash, utilizamos o modelo lógico para ter uma "base" de como será o modelo físico do nosso projeto, imagine a seguinte situação, você está cadastrando suas informações pessoais em algum site ou aplicativo, essas informações ficam armazenadas em algum banco de dados e serão utilizadas posteriormente, as informações que você cadastrar em nosso aplicativo, serão armazenadas em nosso banco de dados e corretamentes divididas, como dito anteriormente, utilizamos o modelo lógico apenas para ter uma ideia. Este modelo possui 4 tabelas, sendo elas "usuários" para o cadastro de dados pessoais do usuário, como a senha, nome de usuário e e-mail, "lixeira" onde ficam armazenados o nome da lixeira, o local, o código único e o

armazenamento de cada compartimento e “dados compartilhados” sendo responsável por armazenar a data em que cada compartimento encheu e quantas vezes cada compartimento ficou cheio.



3.3.3 SQL

Utilizando os comandos no SQL, foi possível popular o banco de dados com as informações que serão importantes, primeiramente foi utilizado o comando insert para inserir os dados nas tabelas, colocando o nome do usuário, email e a senha para acessarem o aplicativo. Após os dados do usuário foram inseridas as informações que vão ser compartilhadas com o usuário sobre a lixeira armazenando a data de enchimento e quantas vezes cada compartimento ficou cheio e por fim as descrições de cada lixeira, mostrando o nome da lixeira, o seu local, o código único de cada lixeira e o preenchimento de cada um dos compartimentos.

Insert:

```
insert into usuario (nomeUsuario, email, senha) values ('Luiz', 'luizcarlos@gmail.com', 'luizinho20');
insert into usuario (nomeUsuario, email, senha) values ('Ana', 'aninha2030@hotmail.com', 'aninha102030');
insert into usuario (nomeUsuario, email, senha) values ('João', 'joao.34@gmail.com', 'joao4510');

select * from usuario;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

idUsuario	nomeUsuario	email	senha
1	Luiz	luizcarlos@gmail.com	luizinho20
2	Ana	aninha2030@hotmail.com	aninha102030
3	João	joao.34@gmail.com	joao4510
NULL	NULL	NULL	NULL

```
insert into dadoscompartilhados (dataenchimento, vezescheiaPA, vezescheiaPL, vezescheiaME, vezescheiaOR, Lixeira_idLixeira)
values ('2023-10-28', '0', '1', '0', '0', 1);
insert into dadoscompartilhados (dataenchimento, vezescheiaPA, vezescheiaPL, vezescheiaME, vezescheiaOR, Lixeira_idLixeira)
values ('2023-10-28', '0', '0', '0', '0', 2);
insert into dadoscompartilhados (dataenchimento, vezescheiaPA, vezescheiaPL, vezescheiaME, vezescheiaOR, Lixeira_idLixeira)
values ('2023-10-28', '0', '0', '0', '0', 3);

select * from dadoscompartilhados
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

idDados compartilhados	dataenchimento	vezescheiaPA	vezescheiaPL	vezescheiaME	vezescheiaOR	Lixeira_idLixeira
1	2023-10-28	0	1	0	0	1
2	2023-10-28	0	0	0	0	2
3	2023-10-28	0	0	0	0	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
-- INSERTS
insert into lixeira (nomeLixeira, local, codigoLixeira, compartimentoPA, compartimentoPL, compartimentoME, compartimentoOR)
values ('Lixeira bloco A', 'Bloco A', 'UNEGX02ES', 10, 100, 80, 05);
insert into lixeira (nomeLixeira, local, codigoLixeira, compartimentoPA, compartimentoPL, compartimentoME, compartimentoOR)
values ('Lixeira bloco B', 'Bloco B', 'AI740VE77', 90, 80, 10, 55);
insert into lixeira (nomeLixeira, local, codigoLixeira, compartimentoPA, compartimentoPL, compartimentoME, compartimentoOR)
values ('Lixeira bloco C', 'Bloco C', 'BDC489EXA', 40, 20, 35, 75);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

idLixeira	nomeLixeira	local	codigoLixeira	compartimentoPA	compartimentoPL	compartimentoME	compartimentoOR
1	Lixeira bloco A	Bloco A	UNEGX02ES	10	100	80	5
2	Lixeira bloco B	Bloco B	AI740VE77	90	80	10	55
3	Lixeira bloco C	Bloco C	BDC489EXA	40	20	35	75
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Após o insert utilizamos o update para alterar alguns dados dos usuários mostrando ser possível a alteração dos mesmos com o comando Update.

Update:

```
-- UPDATE

update usuario
set nomeUsuario = "Luiz Carlos"
where idUsuario = 1;

update usuario
set nomeUsuario = "Ana Vitória"
where idUsuario = 2;

update usuario
set nomeUsuario = "João Vitor M"
where idUsuario = 3;

select * from usuario
```

Result Grid | Filter Rows: | Edit: | Export/Import:

idUsuario	nomeUsuario	email	senha
1	Luiz Carlos	luizcarlos@gmail.com	luizinho20
2	Ana Vitória	aninha2030@hotmail.com	aninha102030
3	João Vitor M	joao.34@gmail.com	joao4510
NULL	NULL	NULL	NULL

Como um teste, o comando delete foi utilizado para deletar as informações sobre uma das lixeiras, e deletou as informações da lixeira de ID 2, que posteriormente foi preenchido com diferentes informações.

Delete:

```
-- Delete

delete from dadoscompartilhados
where idDadoscompartilhados = 2;

select * from dadoscompartilhados
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

idDadoscompartilhados	dataenchimento	vezescheiaPA	vezescheiaPL	vezescheiaME	vezescheiaOR	Lixeira_idLixeira
1	2023-10-28	0	1	0	0	1
3	2023-10-28	0	0	0	0	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Por fim, foi utilizado o comando Join para mostrar na tabela o id de cada um dos responsáveis por cada lixeira, esse comando possibilita mostrar uma informação de outra tabela, nesse exemplo o usuário que pertence a outra tabela mas foi mostrado junto com a lixeira que foi associado.

Select com Join:

```
-- JOIN
50
51 •  select idLixeira as ID, usuario.nomeUsuario as nome, nomelixeira as Pertence from lixeira
52    inner join usuario on (usuario.nomeUsuario=usuario.nomeUsuario);
53
54
55
56
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	nome	Pertence
1	João Vitor M	Lixeira bloco A
1	Ana Vitória	Lixeira bloco B
1	Luiz Carlos	Lixeira bloco C

3.4 GESTÃO FINANCEIRA

A gestão financeira é uma ferramenta essencial para a estruturação e planejamento de um projeto. "Uma pequena vazão irá esvaziar uma grande fortuna, assim como um pequeno vazamento afundará um grande navio." Franklin, B. (1748)

Esta analogia exemplifica como uma pequena falha no planejamento financeiro de um projeto, pode ser o diferencial entre o sucesso do produto e a falha do mesmo.

Dentro da gestão financeira deve-se apurar diferentes valores até chegar a versão que transforme o produto da empresa lucrativo e com uma qualidade que consiga competir com o mercado.

3.4.1 CLASSIFICAÇÃO DOS CUSTOS

A classificação dos custos é separada entre o direto e o indireto. O custo direto é aquele relacionado ao valor ligado à produção. Já o custo indireto é referente aos valores que não estão diretamente ligados ao mesmo.

Um exemplo de custo direto, é a matéria prima ligada diretamente ao produto, no caso da E-trash, seus componentes eletrônicos, a parte exterior são custos ligados diretamente ao produto. Em relação aos custos indiretos do produto estão o aluguel do local de produção além da água e energia elétrica consumida pelos funcionários.

No planejamento do projeto da lixeira E-trash, fizemos o levantamento dos custos diretos (sensor capacitivo, sensor indutivo, motor de passos, 4x micro servo motor, arduino uno,jumpers, cano pvc e madeira) além do custo do salário de quem vai produzi-la e totaliza um R$ 2.364,09 em custos diretos, em custos indiretos, teremos gastos com o aluguel, energia elétrica e a água dos funcionários que totalizam R$ 1.456,80 em custos indiretos na produção.

3.4.2 CUSTOS DO PRODUTO

O projeto da lixeira automática desenvolvida pela JIVLTECH, denominada E-Trash, foi organizada financeiramente utilizando a estratégia de custeamento do produto

Após uma análise aprofundada do mercado, chegou-se à conclusão de que esse mercado não é muito competitivo, já que possuem poucas empresas que atuam nesse ramo, sendo nenhuma atuando em território nacional, apenas está, resultando na E-Trash ser a primeira em território nacional. A eCycle, uma empresa que atua em países estrangeiros, possui uma lixeira semelhante à E-Trash, no entanto, não fornece muitas informações sobre a lixeira em seu site oficial, apenas sabemos que é uma lixeira automatizada. Sendo assim, nosso projeto é válido e viável para o mercado, uma vez que não há outra lixeira no mercado com todas as funcionalidades -Tabela 1 - Planilha de preços projeto

Custos	Preço unitario	Preços em R$
Sensor Capacitivo	89,99	89,99
Sensor indutivo	29,9	29,9
Motor de passos	16,99	16,99
Micro servo motor 360° (4)	27,9	111,6
Arduíno UNO	104,98	104,98
Jumpers	6,99	6,99
Cano PVC(por metro)	3,64	3,64
Madeira m2	200	200
Salario		1.800
Aluguel		716,76
Energia		487,14
Agua		147,9
Internet		105

Tabela 2 - Totais de custos

Total Custos Diretos	2364,09
Total Custos indiretos	1456,8
Custo total	3820,89

3.4.3 PRECIFICAÇÃO

De acordo com a expert xp (2023) “O markup consiste em um índice que tem por objetivo estipular a precificação de determinados produtos e serviços.”

Na precificação da E-trash foi feito um levantamento dos preços e custos da E-trash e descobrimos que com um markup de 30,86% teríamos um índice de Markup de 1,31 com uma margem de lucro de 23,58% em relação ao custo de produção do produto.

Tabela 3 - Custos fixos

Custos fixos:	
Aluguel	716,76
Energia	487,14
Agua	147,9
Internet	105
Total:	1456,8

Para chegarmos a esses números utilizamos os números de uma tabela que calcula o markup automaticamente, nessa tabela incluímos o valor de custos que são fixos (Energia elétrica, Aluguel, Agua e Internet) que totalizam 1.456,80 que soma com os custos variáveis que são 564,09 da matéria prima e 1.800,00 da mão de obra que totalizam 3.456,23. Com esse valor em mente jogamos o valor da E-trash a 5.000,00 que é um valor bem considerável já que não existe um produto para competir no mercado. Com esses valores obtemos uma margem de contribuição de 1.179,11 que é um valor que consegue facilmente pagar o custo fixo.

Produto / Serviço	Custo Fixo	Custo Variável	
		Matéria Prima	Mão-de-obra
E-trash	R$ 1.456,80	R$ 564,09	R$ 1.800,00

Custo Total	Preço	Margem de Contribuição
R$ 3.820,89	R$ 5.000,00	R$1.179,11

Margem	Mark Up	Índice de Markup
23,58%	30,86%	1,31

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: GERENCIANDO FINANÇAS

A unidade de formação para a vida fornece por meio de atividades que contribuam para o desenvolvimento pessoal, profissional e social, oferecendo conhecimentos que poderão ser úteis em diversas situações futuras, assim já tendo noção e sabendo como lidar.

Tem como principal objetivo contribuir para a formação de alunos mais preparados para o mercado de trabalho e para a vida em sociedade, e sempre buscando o desenvolvimento de competências como capacidade de reflexão crítica, trabalho em equipe, comunicação e lidar com a diversidade.

3.5.1 GERENCIANDO FINANÇAS

- **Tópico 1:** O texto destaca como as finanças, economia e contabilidade estão interligadas, sendo cruciais para a gestão tanto de empresas quanto das finanças pessoais. Ele aborda temas como microeconomia e macroeconomia, explicando como a contabilidade oferece uma visão instantânea da situação financeira das empresas.

Além disso, fornece exemplos práticos sobre como classificar gastos e despesas, destacando a importância do controle financeiro. O texto ressalta a aplicabilidade dos princípios empresariais na vida pessoal, enfatizando que controlar os gastos é essencial para o planejamento financeiro. Em resumo, destaca a relevância das ciências sociais para embasar decisões financeiras e promover uma vida financeira saudável.

- **Tópico 2:** O texto destaca a importância da Administração Financeira ao enfatizar o papel essencial do gestor financeiro em maximizar a riqueza do acionista, aplicando esses princípios também na gestão de finanças pessoais. Explora fontes de renda como salários, empreendedorismo e investimentos, destacando estratégias para renda passiva. Aborda a necessidade de redução de custos para alcançar a independência financeira, propondo táticas como renegociação e controle de gastos.

  Conceitos de investimentos, como ativos financeiros e bens permanentes, são discutidos, assim como a importância do perfil do investidor na escolha de instrumentos financeiros. Em resumo, o texto destaca a aplicabilidade dos princípios de Administração Financeira na gestão pessoal, adaptando conceitos empresariais para a realidade financeira individual.Entendendo o ambiente: independência financeira, o valor da minha riqueza e o registro do dia a dia
- **Tópico 3:** A Matemática Financeira explora o valor do dinheiro no tempo, frequentemente através da análise de fluxos de caixa, representados por entradas e saídas monetárias. Seus objetivos variam entre quem aplica, buscando taxas de juros mais altas, e quem toma emprestado, visando taxas mais baixas, influenciadas pela oferta e demanda do mercado. Os juros simples, baseados apenas no capital inicial, contrastam com os juros compostos, que incorporam os juros anteriores ao capital. Ambos os sistemas têm implicações diferentes no investidor e no mercado. A concessão de crédito, sujeita a análises rigorosas, é crucial, considerando a obtenção de garantias e a necessidade de reavaliação periódica. A organização financeira, envolvendo controle de recursos, definição de prioridades e elaboração de orçamento, é fundamental para alcançar objetivos, demandando disciplina e, preferencialmente, automação de pagamentos. Investir em educação financeira e conhecer assuntos como contabilidade e matemática financeira é essencial, assim como a preparação de um plano de ação orçamentário. Relações pré-estruturadas de direitos e obrigações facilitam a alocação de recursos e evitam gastos impulsivos.

- **Tópico 4:** Imaginar um futuro promissor e uma aposentadoria tranquila, frutos dos sonhos e metas diárias, é como construir as bases para um caminho suave lá na frente. Essa jornada começa agora, no presente, demandando planejamento e ações concretas. Na gestão financeira, tanto pessoal quanto empresarial, adotar hábitos saudáveis é essencial. Manter o equilíbrio nas movimentações, realizar acompanhamentos diários do fluxo de caixa são práticas que fazem toda a diferença. Transformar sonhos em projetos é um passo importante, exigindo organização e um olhar atento para as ações futuras.

  A educação financeira desempenha um papel crucial, orientando-nos na busca do equilíbrio e na realização de nossas aspirações. No entanto, é importante desmistificar algumas ideias equivocadas, como a crença de que investir é algo reservado para grandes quantias ou que gastar apenas o que ganhamos é seguro.

  Alcançar nossos objetivos financeiros requer estabelecer metas claras, criar estratégias e praticar diariamente o planejamento e controle financeiro. As atitudes certas são a chave para transformar sonhos em realidade, respeitando nossos limites e alinhando projetos com os recursos disponíveis. Quando falamos de aposentadoria, a antecipação, considerar diferentes opções de investimento e uma preparação adequada são passos essenciais para encarar a velhice com tranquilidade financeira, algo que, como estudantes universitários, podemos começar a moldar desde já.

3.5.2 ESTUDANTES NA PRÁTICA

Com a apostila de Gerenciando Finanças foi feito um vídeo com base no conteúdo de forma objetiva e com a participação de todos do grupo.

Link: https://www.youtube.com/watch?v=5mYgHazDCdo

4. CONCLUSÃO

Para concluir, a E-trash experimentou uma notável mudança desde seu conceito inicial, que inicialmente era apenas para dispositivos móveis e explorava a Internet das Coisas (IoT). A evolução do projeto incluiu uma migração efetiva para plataformas desktop, incorporando não apenas uma interface funcional, mas também um banco de dados dedicado. Além disso, foi introduzida uma abordagem estruturada de precificação, integrada à gestão financeira do sistema.

Inicialmente foram encontradas algumas dificuldades, como a integração do banco de dados com o código da programação que foi solucionado após algumas alterações, além disso encontramos alguns problemas na hora de calcular alguns valores para chegarmos ao preço final, mas após alguns cálculos chegamos a um resultado aceitável.

Essa jornada não apenas reflete avanços tecnológicos, mas também destaca a importância de aprender com desafios. A E-trash não é apenas um projeto tecnológico em constante evolução; é um exemplo de como enfrentar problemas, aprender com eles e seguir adiante. Essa experiência reforça a ideia de que, mesmo diante de obstáculos, é possível encontrar soluções e progredir.

REFERÊNCIAS

STROUSTRUP, B.. Programming: Principles and Practice Using C++.

EPR CONSULTORIA: Custos indiretos: entenda o que são e qual sua relação com custos diretos. Disponível em: https://eprconsultoria.com.br/custos-indiretos/#:~:text=Os%20custos%20diretos%20s%C3%A3o%20aqueles. Acesso em: 20 set. 2023.

SIMULARE. Markup: O que é, sua importância e como calcular? Disponível em: https://conteudos.xpi.com.br/aprenda-a-investir/relatorios/markup-o-que-e-como-calcular/#:~:text=O%20markup%20consiste%20em%20um. Acesso em: 2 nov. 2023.

MICROSOFT. Data Type Summary. Disponível em: https://learn.microsoft.com/pt-br/office/vba/language/reference/user-interface-help/data-type-summary. Acesso em: 23 out. 2023.

AWS. O que é modelagem de dados? – Explicação sobre modelagem de dados.Disponível em: https://aws.amazon.com/pt/what-is/data-modeling/#:~:text=Modelagem%20de%20dados%20%C3%A9%20o. Acesso em: 7 nov. 2023.

MCCARTHY, John (1958) - Wikipedia. Programação lógica. Disponível em: https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_l%C3%B3gica. Acesso em: 26 out. 2023.

TREINAWEB. Manipulando arquivos com Python. Disponível em: https://www.treinaweb.com.br/blog/manipulando-arquivos-com-python. Acesso em: 1 nov. 2023.

ANEXOS

Código JavaScript:

app.js > app.post('/registrarlixeira') callback

```
const express = require('express');
const bodyParser = require('body-parser');
const mysql = require('mysql2');
const path = require('path');
const app = express();
const port = 3000;

const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'lixeirapi',
});

db.connect((err) => {
  if (err) {
    console.error('Erro ao conectar ao banco de dados MySQL:', err);
  } else {
    console.log('Conexão bem-sucedida ao banco de dados MySQL');
  }
});
const redirecionar = require('./redirecionar');
app.use(redirecionar);

app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static(path.join(__dirname, 'public')));


app.post('/logar', (req, res) => {
  const { email, senha } = req.body;

  const loginQuery = 'SELECT * FROM usuario WHERE email = ? AND senha = ?';
  db.query(loginQuery, [email, senha], (err, result) => {
```

app.js > app.post('/registrarlixeira') callback

```
app.post('/registrar', (req, res) => {
  const { nome, email, senha } = req.body;

  const checkEmailQuery = 'SELECT * FROM usuario WHERE email = ?';
  db.query(checkEmailQuery, [email], (err, result) => {
    if (err) {
      console.error('Erro ao verificar o email no banco de dados:', err);
      res.status(500).json({ success: false, message: 'Erro interno do servidor' });
    } else if (result.length > 0) {
      res.status(401).json({ success: false, message: 'Usuário já cadastrado!' });
    } else {
    const insertUserQuery = 'INSERT INTO usuario (nome, email, senha) VALUES (?, ?, ?)';
    db.query(insertUserQuery, [nome, email, senha], (insertErr, insertResult) => {
    if (insertErr) {
    console.error('Erro ao inserir dados no banco de dados:', insertErr);
    res.status(500).json({ success: false, message: 'Erro ao registrar o usuário' });
    } else {
    console.log('Usuário registrado com sucesso!');
    res.json({ success: true, message: 'Registro bem-sucedido' });
        }
});
}
});
});

app.post('/registrarlixeira', (req, res) => {
  const { nome, localidade, codigo } = req.body;
  const checkLixeiraQuery = 'SELECT * FROM lixeira WHERE codigo = ?';
  db.query(checkLixeiraQuery, [codigo], (lixeiraErr, lixeiraResult) => {
    if (lixeiraErr) {
      console.error('Erro ao verificar o código da lixeira no banco de dados:', lixeiraErr);
      res.send('Erro ao verificar o código da lixeira');
```

```
const express = require('express');
const app = express();
app.get('/inicial', (req, res) => {
    res.sendFile(__dirname + '/login.html');
  });

  app.get('/registro', (req, res) => {
    res.sendFile(__dirname + '/registro/registro.html');
  });

  app.get('/registrolixeira', (req, res) => {
    res.sendFile(__dirname + '/registro/registrolixeira.html');
  });

  app.get('/painel', (req,res)=>{
    res.sendFile(__dirname + '/painel/painel-controle.html')
  });

  app.get('/cadastrolixeira',(req,res)=>{
    res.redirect('/registrolixeira')
  })

  app.post('/redirecionarRegistro',(req,res)=>{
    res.redirect('/registro')
  })

  app.post('/buttonClick',(req,res)=>{
    res.redirect('/inicial')
  })

  app.post('/inicio',(req,res)=>{
    res.redirect('/inicial')
  })
```