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

<AUT4US>

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

<AUT4US>

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:
Camily Ribeiro, RA 23000251
Gabrielly Simão Domingos, RA 23000615
Lucas Bernardes Genari, RA 23000511
Hugo Villela Andrade, RA 23000789

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2023

SUMÁRIO

ISSN 1983-6767

1. INTRODUÇÃO

No cenário em constante evolução da Internet das Coisas (IoT), a gestão eficiente e analítica dos dados provenientes de dispositivos conectados tornou-se crucial para o sucesso operacional das organizações. Com o intuito de atender a essa demanda crescente, o projeto integrado "IoT Data Streamer" propõe uma solução inovadora para a empresa AUT4US.

O objetivo primordial do IoT Data Streamer é proporcionar uma aplicação desktop robusta e intuitiva, dedicada ao gerenciamento e análise de dados oriundos do aplicativo móvel da empresa. Esta aplicação será responsável por simular a importação de dados em formatos TXT e CSV, processando-os e armazenando-os de maneira organizada em um banco de dados centralizado.

Principais Funcionalidades:

- **Simulação de Importação de Dados:** O sistema oferece a capacidade de simular a importação de dados pré-existentes em formatos TXT ou CSV. Estes dados serão devidamente processados e armazenados, garantindo a disponibilidade para consultas futuras.
- **Armazenamento em Banco de Dados:** Os dados importados serão persistidos de forma eficiente em um banco de dados, promovendo a recuperação ágil e estruturada dessas informações valiosas.
- **Consulta e Filtros:** A partir do banco de dados, o sistema possibilita a realização de consultas com filtros específicos, como datas, dias da semana, ou outros critérios personalizados pelos usuários.
- **Visualizações:** As consultas efetuadas serão apresentadas de forma intuitiva e informativa, por meio de gráficos, tabelas e outros formatos apropriados, facilitando a interpretação dos dados coletados.
- **Cadastro Manual:** Além da importação automatizada, os usuários terão a opção de cadastrar novos dados manualmente, assegurando a flexibilidade e completude do banco de dados.
- **Recursos Avançados:** O sistema incorpora funcionalidades avançadas, como a detecção de padrões nos dados e a geração de alertas, bem como a capacidade de enviar notificações por e-mail para o gerenciador quando determinadas condições forem satisfeitas.

- **Login e Segurança:** A segurança dos dados é prioridade. O sistema implementa um sistema de login para garantir o acesso exclusivo a usuários autorizados, protegendo assim a integridade das informações.

2. DESCRIÇÃO DA EMPRESA

Informações Básicas da Empresa:

- **Razão Social**: Centro Universitário Fundação de Ensino Octávio Bastos (UNIFEOB)
- **CNPJ**: 59.764.555/0001-52
- **Endereço**: Av. Dr. Otávio da Silva Bastos, 2439 - Jardim Nova São João, São João da Boa Vista - SP, 13874-149
- **Atividade**: Instituição de Ensino Superior

Contextualização do Mercado:

A UNIFEOB atua no setor educacional, mais especificamente no segmento de ensino superior. No Brasil, o setor educacional é fundamental para o desenvolvimento socioeconômico do país. Com o crescente acesso à educação superior, a demanda por cursos de graduação e pós-graduação tem se mantido em ascensão.

Principais Produtos e Serviços:

- **Cursos de Graduação**: A UNIFEOB oferece uma ampla variedade de cursos de graduação em áreas como [listar algumas áreas, por exemplo: administração, direito, engenharia, etc.].
- **Pós-Graduação**: Além dos cursos de graduação, a instituição também disponibiliza programas de pós-graduação nas áreas de [inserir áreas de pós-graduação, por exemplo: MBA, mestrado, etc.].

Metodologia de Coleta de Informações:

Para a obtenção das informações necessárias para este projeto, foram realizados encontros com empresários ligados à UNIFEOB, onde foram discutidos os aspectos fundamentais da instituição, bem como seu papel no desenvolvimento educacional e econômico da região.

Além dos encontros presenciais, também foram utilizadas outras fontes como livros, jornais, revistas e sites especializados em educação. Todas as fontes utilizadas foram devidamente referenciadas conforme a Metodologia Científica.

ISSN 1983-6767

3. PROJETO INTEGRADO

Neste estágio do Projeto Integrador, focaremos na adaptação dos conteúdos de estudo para a realidade da empresa escolhida. Isso exige não só compreensão teórica, mas também aplicação prática.

Ao alinhar o conhecimento com as necessidades da empresa, prepararemos soluções efetivas. Esta etapa une teoria e prática, traduzindo nosso esforço em resultados tangíveis.

Juntos, vamos nos assegurar que cada elemento se encaixe perfeitamente, criando um projeto que trará benefícios concretos.

3.1 PROGRAMAÇÃO ORIENTADA A OBJETO

Na etapa 3.1 do nosso Projeto Integrado, introduziremos um conceito essencial para o desenvolvimento de software moderno: a Programação Orientada a Objetos (POO). Esta abordagem revolucionária nos permite organizar e estruturar nosso código de forma mais eficiente, proporcionando uma base sólida para a construção do nosso projeto.

Ao adotar a POO, estaremos criando classes e objetos que representam entidades do mundo real, facilitando a compreensão e a manutenção do código. Além disso, a POO promove a reutilização de código, economizando tempo e recursos no desenvolvimento.

Nesta fase, exploraremos os princípios fundamentais da Programação Orientada a Objetos e como aplicá-los de maneira eficaz em nosso projeto. Segundo Ambler (1998), os conceitos fundamentais são em maior número: classes, objetos, instância, atributos, métodos, abstração,encapsulamento, herança, persistência, relacionamento entre instâncias,acoplamento, entre outros.

3.1.1 CLASSES E OBJETOS

Classes e objetos são conceitos fundamentais na programação orientada a objetos (POO).

- **Classe**: É um modelo ou um plano para criar objetos. Ela define os atributos (dados) e métodos (comportamentos) que os objetos terão.
- **Objeto**: É uma instância de uma classe. Representa uma entidade do mundo real que possui características (atributos) e pode executar ações (métodos).

  Exemplos: vide Anexos.

3.1.2 ATRIBUTOS, MÉTODOS, ENCAPSULAMENTO, HERANÇA E POLIMORFISMO.

A POO promove a reutilização de código, facilita a manutenção e melhora a organização de sistemas complexos. Ela é baseada em quatro princípios fundamentais: abstração, encapsulamento, herança e polimorfismo.

- **Atributos**

Atributos são as características que definem um objeto. Eles representam o estado de um objeto e podem incluir informações como nome, idade, valor, etc. Esses dados são armazenados como variáveis dentro da classe.

- **Métodos**

Métodos são as operações ou ações que um objeto pode realizar. Eles representam o comportamento associado a um objeto e podem manipular seus atributos, realizar cálculos ou interagir com outros objetos.

- **Encapsulamento**

O encapsulamento é um conceito que visa proteger os dados de uma classe, controlando o acesso a eles. Isso é feito através da definição de níveis de visibilidade para os atributos e métodos, como público, privado e protegido. O encapsulamento ajuda a garantir a integridade dos dados e facilita a manutenção do código.

- **Herança**

A herança é um mecanismo que permite que uma classe herde os atributos e métodos de outra classe. A classe que é herdada é chamada de classe pai ou superclasse, e a classe que herda é chamada de classe filha ou subclasse. Isso promove a reutilização de código e a criação de hierarquias de classes.

- **Polimorfismo**

O polimorfismo permite que um objeto ou método possa se comportar de diferentes maneiras, dependendo do contexto em que é utilizado. Isso significa que diferentes classes podem fornecer implementações diferentes para o mesmo método. Isso promove a flexibilidade e extensibilidade do código.

Exemplos: vide Anexos.

3.1.3 MÉTODOS ESTÁTICOS, PÚBLICOS E PRIVADOS

Vamos agora abordar os conceitos de métodos estáticos, públicos e privados com exemplos práticos para cada um deles.

- **Métodos Estáticos**

Métodos estáticos pertencem à classe como um todo, não a instâncias específicas. Eles são úteis para operações que não dependem do estado de um objeto e podem ser chamados diretamente na classe, sem a necessidade de criar uma instância.

- **Métodos Públicos**

Métodos públicos são acessíveis de fora da classe e podem ser chamados por instâncias da classe ou pela própria classe. Eles representam as operações que uma classe pode realizar e são essenciais para a interação com objetos.

- **Métodos Privados**

Métodos privados são acessíveis apenas dentro da própria classe onde foram declarados. Eles não podem ser chamados de fora da classe e são utilizados para implementar funcionalidades internas, muitas vezes para auxiliar os métodos públicos.

Estes conceitos são fundamentais na POO e desempenham um papel importante na criação de código organizado, reutilizável e de fácil manutenção. Vamos agora ilustrar cada um deles com exemplos práticos.

Exemplos: vide Anexos.

3.2 LÓGICA DE PROGRAMAÇÃO

A lógica da programação é uma parte fundamental no início do desenvolvimento de sistemas. Nesse tópico, os estudantes aprenderão como funciona a lógica por trás dos computadores e como aplicá-la na prática. Serão abordados conceitos fundamentais, como algoritmos, variáveis, tipos de dados, funções, estruturas condicionais, operadores lógicos e operadores de comparação, voltados para a linguagem de programação JavaScript. Além disso, também trabalharão com prototipação, criação de aplicações desktop com o framework Electron JS e banco de dados relacional (MySQL).

3.2.1 CONCEITOS FUNDAMENTAIS DO DESENVOLVIMENTO DE SOFTWARE

"No desenvolvimento de um projeto de *software* quanto mais complexo é o *software*, maior é o empenho que o engenheiro de *software* deve fazer para desenvolver e tem que ter maior gerenciamento" (JALOTE, 2005).

A lógica de programação é o alicerce fundamental para qualquer programador, independentemente da linguagem de programação que esteja sendo utilizada. Vamos dar uma breve explicação de cada um desses conceitos:

- **Algoritmos**: São sequências de passos ou instruções que resolvem um problema ou realizam uma tarefa específica. Eles são como receitas que definem o que um programa deve fazer.
- **Variáveis**: São espaços na memória reservados para armazenar dados que podem ser utilizados e manipulados pelo programa. Cada variável possui um nome e um tipo, que determina que tipo de dados ela pode armazenar (como números inteiros, decimais, textos, etc.).
- **Tipos de Dados**: Definem o tipo de informação que uma variável pode armazenar. Alguns exemplos incluem inteiros (int), números de ponto flutuante (float), caracteres (char), strings (cadeias de caracteres), booleanos (bool), entre outros.
- **Funções**: São blocos de código que podem ser reutilizados para realizar uma tarefa específica. Elas recebem entradas (parâmetros), processam esses dados e podem retornar um resultado.
- **Estruturas Condicionais**: Permitem que um programa tome decisões com base em condições. Por exemplo, o famoso "if-else" verifica se uma condição é verdadeira e executa um bloco de código se ela for, ou outro bloco se não for.
- **Operadores**: São símbolos ou palavras-chave que indicam ações a serem realizadas em variáveis ou valores. Eles podem ser aritméticos (+, -, *, /), relacionais (==, !=, <, >, <=, >=), lógicos (&&, ||, !), entre outros.

Compreender e saber aplicar esses conceitos é essencial para escrever código de forma eficiente e resolver problemas computacionais de maneira eficaz. A prática constante e a resolução de exercícios são ótimas maneiras de consolidar esse conhecimento. Além disso, conforme o programador avança, ele pode explorar conceitos mais avançados e técnicas de otimização de código.

3.2.2 DESENVOLVIMENTO DE APLICAÇÕES

A ligação entre o front-end e o back-end em um aplicativo ou sistema de software envolve a coordenação de ações e a troca de informações para garantir que as operações CRUD (Criar, Ler, Atualizar, Excluir) sejam executadas corretamente. Aqui está como o front-end e o back-end se relacionam na execução das operações CRUD:

- **Front-End (Interface do Usuário)**:
    - A interface do usuário, que é a parte visível e interativa do aplicativo, permite que os usuários interajam com os dados e enviem solicitações ao servidor.
    - Os usuários inserem informações por meio de formulários, clicam em botões e navegam pelas telas do aplicativo por meio da interface do usuário.
    - Quando os usuários desejam criar, ler, atualizar ou excluir dados, eles acionam ações no front-end que desencadeiam solicitações HTTP (como POST, GET, PUT ou DELETE) para o back-end.
- **Back-End (Lógica de Servidor e Banco de Dados)**:
    - O back-end é a parte do aplicativo que lida com o processamento das solicitações do front-end e a manipulação dos dados no banco de dados.
    - Quando uma solicitação CRUD é recebida do front-end, o back-end realiza as seguintes ações:
        - **Create (Criar)**: O back-end recebe os dados do front-end e insere um novo registro no banco de dados.
        - **Read (Ler)**: O back-end recebe uma solicitação de leitura, consulta o banco de dados e envia os dados solicitados de volta ao front-end.
        - **Update (Atualizar)**: O back-end recebe dados atualizados do front-end, pesquisa o registro existente no banco de dados e aplica as alterações.
        - **Delete (Excluir)**: O back-end recebe uma solicitação de exclusão, localiza o registro a ser excluído no banco de dados e remove-o.
- **Comunicação e Protocolos**:
    - A comunicação entre o front-end e o back-end geralmente é feita por meio de solicitações HTTP, como requisições POST, GET, PUT e DELETE.
    - As respostas do back-end são frequentemente retornadas no formato JSON ou XML e são interpretadas pelo front-end para atualizar a interface do usuário conforme necessário.
- **Validações e Segurança**:

- Tanto o front-end quanto o back-end devem implementar verificações de segurança e validações de dados para garantir a integridade dos dados e proteger o aplicativo contra ações maliciosas.
- O back-end deve garantir que apenas os usuários autorizados tenham permissão para executar operações CRUD, e o front-end pode ajudar a fornecer uma experiência amigável ao usuário, indicando erros ou validações em tempo real.

A ligação entre o front-end e o back-end em um aplicativo é crucial para a funcionalidade e o desempenho do sistema. O front-end cuida da interação do usuário, enquanto o back-end gerencia os dados e a lógica de negócios. A implementação eficaz e a comunicação adequada entre essas duas camadas garantem que as operações CRUD sejam executadas de maneira consistente e segura.

O CRUD é fundamental na lógica de programação e no desenvolvimento de aplicativos, pois permite que os desenvolvedores executem todas as operações essenciais de manutenção de dados em um banco de dados. Essas operações são a base de muitas aplicações, desde sistemas de gerenciamento de conteúdo até aplicativos de comércio eletrônico e muito mais. Geralmente, as operações CRUD são executadas por meio de consultas SQL (Structured Query Language) em bancos de dados relacionais ou por meio de chamadas a APIs em sistemas de gerenciamento de dados não relacionais, como bancos de dados NoSQL.

Ao implementar o CRUD, os desenvolvedores podem criar aplicativos que permitem aos usuários criar, ler, atualizar e excluir dados de maneira eficaz, mantendo a integridade e a segurança dos dados.

Exemplos: vide Anexos.

3.3 MODELAGEM DE DADOS

Chegamos a uma etapa fundamental no desenvolvimento de nosso sistema: a criação do banco de dados. Inicialmente, iremos conceber o modelo lógico, uma representação abstrata e independente de plataforma dos dados e suas relações.

Em seguida, procederemos à transição para o modelo físico, considerando meticulosamente a eficiência e o desempenho do banco de dados. Esta fase é de importância crítica para assegurar que nosso sistema opere de maneira eficaz e consistente.

3.3.1 MODELO CONCEITUAL

Relacionamentos ainda podem ter atributos, quando algum dado precisa ser associado à ligação das duas instâncias envolvidas. Relacionamentos são descritos através da cardinalidade, que indica como as instâncias das entidades se relacionam. (KORTH, SILBERSCHATZ a SUDARSHAN, 2006).

- O modelo conceitual é a fase inicial de projeto de banco de dados.
- Ele descreve a estrutura de dados e as relações entre elas de uma forma independente do sistema de gerenciamento de banco de dados (SGBD) específico.
- O foco está na representação abstrata dos dados e nas entidades (tabelas) envolvidas, sem se preocupar com a implementação técnica.
- É útil para entender e comunicar as necessidades de dados entre os stakeholders do projeto.

  Exemplos: vide Anexos.

3.3.2 MODELO LÓGICO E FÍSICO

Modelo Lógico:

- O modelo lógico é uma representação de alto nível da estrutura de dados e das relações entre elas, independentemente de um sistema de gerenciamento de banco de dados específico.
- Ele se concentra nas informações, relações e restrições de integridade dos dados.
- É útil para a compreensão e comunicação dos requisitos de dados entre as partes envolvidas no projeto.

Modelo Físico:

- O modelo físico descreve a implementação concreta de como os dados serão armazenados em um sistema de gerenciamento de banco de dados específico.
- Ele detalha a estrutura das tabelas, tipos de dados, índices, restrições e otimizações de desempenho.
- É voltado para a eficiência e a otimização da manipulação dos dados, considerando fatores de hardware e consulta específicos.

  Exemplos: vide Anexos.

3.3.3 SQL

SQL (Structured Query Language) é uma linguagem de programação utilizada para gerenciar e manipular bancos de dados relacionais. É uma linguagem padronizada que permite aos desenvolvedores e administradores de banco de dados executar diversas operações em bancos de dados, como inserir, atualizar, recuperar e excluir dados. SQL é amplamente utilizado em sistemas de gerenciamento de bancos de dados relacionais (RDBMS) como MySQL, PostgreSQL, Microsoft SQL Server, Oracle, e muitos outros.

Alguns dos principais usos do SQL incluem:

1. **Consulta de Dados (SELECT)**: SQL é usado para recuperar informações específicas de um banco de dados. Os desenvolvedores podem usar instruções SQL para realizar consultas complexas que filtram, classificam e agrupam dados de acordo com critérios específicos.
2. **Inserção de Dados (INSERT)**: SQL permite a inserção de novos dados em tabelas de um banco de dados. Os desenvolvedores podem especificar os valores a serem inseridos em cada coluna.
3. **Atualização de Dados (UPDATE)**: Com SQL, é possível modificar os dados existentes em um banco de dados. Os desenvolvedores podem atualizar valores em uma ou mais linhas de uma tabela.
4. **Exclusão de Dados (DELETE)**: SQL permite a exclusão de registros específicos de uma tabela de banco de dados. Isso é útil para remover informações obsoletas ou indesejadas.
5. **Criação e Modificação de Tabelas (CREATE e ALTER)**: É possível criar novas tabelas no banco de dados e modificar a estrutura das tabelas existentes usando SQL. Isso inclui a definição de tipos de dados, chaves primárias, índices e restrições.

6. **Gerenciamento de Permissões (GRANT e REVOKE)**: SQL é usado para conceder ou revogar permissões de acesso a tabelas, visualizações e outros objetos do banco de dados, ajudando a controlar a segurança dos dados.

SQL é uma ferramenta poderosa para manipular dados em bancos de dados relacionais e é amplamente utilizado em sistemas de gerenciamento de banco de dados em todo o mundo. É uma habilidade fundamental para desenvolvedores, administradores de banco de dados e qualquer pessoa envolvida no trabalho com dados armazenados em bancos de dados relacionais.

Exemplos: vide Anexos.

3.4 GESTÃO FINANCEIRA

O custo é o montante financeiro necessário para executar um projeto, englobando todos os recursos, materiais, mão de obra e serviços envolvidos. Sua importância dentro do projeto é fundamental, pois uma gestão adequada dos custos permite o controle financeiro, evita desperdícios, auxilia na tomada de decisões e contribui para a viabilidade e o sucesso do empreendimento.

Além disso, o custo também influencia diretamente no planejamento, na definição de prazos e na qualidade final do projeto. Uma análise criteriosa dos custos possibilita o alcance dos objetivos traçados, a maximização dos recursos disponíveis e a obtenção de resultados satisfatórios. Portanto, considerar o custo como um aspecto central do projeto é essencial para sua eficiência e sustentabilidade.

3.4.1 CLASSIFICAÇÃO DOS CUSTOS

A empresa AUT4US, se dedica à automação residencial e empresarial, a fim de tornar mais prático as atividades do cotidiano. A automação envolve o controle de projetores, ar condicionado e iluminação em edifícios de A a F, bem como suas salas e laboratórios. É importante identificar todos os custos associados ao projeto AUT4US.

É importante classificar os custos em diferentes categorias, como custos diretos (aqueles diretamente relacionados ao projeto):

- Custos de desenvolvimento de software para a tela de login e cadastro.
- Custos de hardware, como sensores, controladores, dispositivos de automação e equipamentos de projeção.
- Custos de mão de obra para a instalação e configuração dos sistemas.

Custos indiretos (custos gerais da empresa que contribuem para o projeto):

- Custos de marketing e vendas para a promoção dos serviços de automação.
- Custos de manutenção e suporte técnico contínuo.

Além disso, você pode categorizar outros custos como fixos (permanentes) e variáveis (que dependem da escala do projeto) e como custos de capital (investimentos iniciais) e custos operacionais (custos recorrentes).

3.4.2 CUSTOS DO PRODUTO

Abaixo estão duas tabelas com os dados relacionado aos custos da empresa AUT4US e seus concorrentes em relação ao mercado:

Tabela 1 - Valores dos produtos

AUT4US		**CONCORRÊNCIA**	
Custo do Produto	**Valores**	**Custo do Produto**	**Valores**
Desenvolvimento de software	R$ 500,00	Desenvolvimento de software	R$ 300,00
Hardware	R$ 8.000,00	Hardware	R$ 3.000,00
Mão de obra	R$ 2.000,00	Mão de obra	R$ 1.200,00
Marketing e Vendas	R$ 600,00	Marketing e Vendas	R$ 450,00
Manutenção e Suporte Técnico	R$ 300,00	Manutenção e Suporte Técnico	R$ 200,00
Total	R$ 11.400,00	**Total**	R$ 5.150,00

Fonte: Autores

A AUT4US é mais viável pois, possui preços mais elevados quando comparados aos concorrentes, porém esse valor é compensado por uma mão de obra mais qualificada, produtos de alta qualidade e com respaldo da empresa que fornece esses produtos por se tratar de uma empresa brasileira com fábricas aqui no Brasil.

Portanto garante um produto final muito superior às marcas encontradas no mercado, e com isso precisa-se de um marketing à altura para ter uma maior interação com os clientes e sempre atingindo o público alvo.

Enquanto isso, o suporte técnico por trás dessa tecnologia, é o mais qualificado do mercado e sempre está atualizado com o mercado, e presente em todos os projetos da AUT4US auxiliando-os nas tarefas diárias do aplicativo e monitorando os equipamentos remotamente.

3.4.3 PRECIFICAÇÃO

A precificação é uma prática essencial nos negócios, envolvendo a determinação do valor monetário de produtos ou serviços.Vai além de simplesmente atribuir um preço, incorporando análises de custos, concorrência, percepção de valor e estratégias de mercado. Essa abordagem equilibrada busca maximizar lucros, garantindo aceitação no mercado. A precificação é dinâmica, influenciada por fatores econômicos, tendências e inovações. Uma compreensão profunda dos custos e do valor percebido pelos consumidores é fundamental. Segundo Wernke (2005), toda a atenção deve ser dada à precificação, pois no ambiente de mercado a WERNKE, R.Análise de custos e preços de venda: ênfase em aplicações e casos nacionais. São Paulo: Saraiva, 2005. Atualmente, a concorrência é acirrada, toda a atenção deve ser dada a esta estratégia mercadológica mais crucial na formação de preços.

Tabela 2 - Formação de Preço

Custo	**Valores**
Custo Fixo	900,00
Matéria Prima	8.000,00
Mão de Obra	2.000,00
Software	500,00

Custo Total	11.400,00
Preço	16.000,00
Margem de Contribuição	4.600,00
Margem	28,75%
Mark Up	40,35%
Índice de Markup	1,4

Fonte: Autores

- Margem de contribuição:

A Margem de Contribuição é um indicador financeiro que representa a diferença entre as receitas e os custos variáveis de uma empresa, ou seja, é o valor que sobra das vendas após descontados os custos diretamente relacionados à produção ou venda de um produto ou serviço.

Na tabela o valor da margem de contribuição é de R$4.600,00 isso significa que após deduzir todos os custos associados à produção ou venda, restaram R$4.600,00 de lucro.

- Mark up:

O Mark up é um índice aplicado sobre o custo de produção ou compra de um produto ou serviço para determinar o preço de venda. Ele representa a porcentagem que você adiciona ao custo para obter o preço final.

Na tabela o Mark up é de 40,35%, isso significa que está adicionando 1,40 do custo para obter o preço de venda.

- Margem:

A margem no contexto de lucratividade se refere à porcentagem de lucro obtida a partir das vendas.

Na tabela a margem é de 28,75%, isso significa que 28,75% do valor das vendas se traduzem em lucro.

Portanto, em conjunto, esses três conceitos são cruciais para determinar os preços de venda, avaliar a rentabilidade dos produtos ou serviços e gerenciar os resultados financeiros de um negócio.

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: GERENCIANDO FINANÇAS

3.5.1 GERENCIANDO FINANÇAS

De acordo com Silva *et al.* (2017), a necessidade da educação financeira tem aumentado significativamente com a desregulamentação dos mercados financeiros; com o fácil acesso ao crédito; com a elevada emissão de cartões de crédito; e com o rápido crescimento na comercialização de produtos financeiros. Nesse cenário, o indivíduo que possui pouco conhecimento financeiro é facilmente influenciado por ganhos que estão aquém da realidade.

Tópico 1: Introdução aos conceitos econômicos e financeiros básicos;

Neste tópico, os estudantes serão introduzidos aos conceitos fundamentais de economia e finanças. Isso inclui noções sobre renda, despesas, poupança, investimento, inflação, entre outros. Exemplos práticos podem incluir:

Renda e Despesas: Compreender quanto se ganha e o quanto é gasto é essencial. Por exemplo, calcular e manter um orçamento mensal.

Poupança e Investimento: Explicar a importância de poupar e investir para o futuro. Mostrar diferentes opções de investimentos, como poupança, CDB, ações, entre outros.

Tópico 2: Entendendo o ambiente: independência financeira, o valor da minha riqueza e o registro do dia a dia;

Este tópico foca na importância de compreender a situação financeira pessoal. Exemplos práticos podem incluir:

Independência Financeira: Discutir o conceito de independência financeira e como ela é alcançada através do equilíbrio entre receitas e despesas.

Registro Financeiro: Ensinar a importância de manter um registro das finanças pessoais; Exemplo: manter um registro de gastos mensais em uma planilha.

Tópico 3: Dívidas e juros compostos, opções de empréstimo e alternativas ao endividado;

Este tópico aborda o gerenciamento de dívidas e os efeitos dos juros compostos. Exemplos práticos podem incluir:

Juros Compostos: Explicar como os juros compostos afetam dívidas e investimentos. Por exemplo, calcular o montante final de um empréstimo com juros compostos.

Alternativas ao Endividamento: Discutir alternativas ao endividamento, como criar um fundo de emergência ou buscar ajuda financeira em instituições.

Tópico 4: Estabelecer metas para a realização de seus sonhos e como envolver o grupo a que você pertence para atingir seus objetivos;

Neste tópico, o foco é estabelecer metas financeiras e envolver a comunidade para alcançá-las. Exemplos práticos podem incluir:

Estabelecimento de Metas: Orientar os estudantes a definir metas financeiras específicas, mensuráveis, alcançáveis, relevantes e com prazo determinado (metodologia SMART).-

Envolver o Grupo: Mostrar como o apoio da família, amigos e comunidade pode ser valioso para alcançar objetivos financeiros. Por exemplo, organizar grupos de economia coletiva.

3.5.2 ESTUDANTES NA PRÁTICA

Foi feito um vídeo sobre o tema e anexado no classroom.

ISSN 1983-6767

4. CONCLUSÃO

O projeto integrado relacionado a empresa AUT4US propõe uma solução inovadora, que visa atender à crescente demanda por gestão eficiente e analítica de dados provenientes de dispositivos conectados na era da Internet das Coisas (IoT). A aplicação desktop desenvolvida terá funcionalidades robustas, como simulação de importação de dados, armazenamento em banco de dados, consultas e filtros, visualizações intuitivas, cadastro manual, detecção de padrões e alertas, além de garantir a segurança dos dados por meio de um sistema de login.

A AUT4US, inserida no setor de automação residencial e empresarial, se destaca no mercado ao oferecer produtos de alta qualidade e mão de obra qualificada. A estratégia de precificação, considerando os custos e a margem de contribuição, demonstra a viabilidade econômica do projeto.

O projeto Integrado introduz a Programação Orientada a Objetos (POO), um conceito essencial para o desenvolvimento de software moderno. Com foco em classes, objetos, atributos, métodos, encapsulamento, herança e polimorfismo, os fundamentos da POO serão aplicados de maneira eficaz no desenvolvimento do "IoT Data Streamer".

A lógica de programação é destacada como a base para o desenvolvimento de sistemas, abordando conceitos essenciais como algoritmos, variáveis, tipos de dados, funções, estruturas condicionais, operadores e a ligação entre o front-end e o back-end. A implementação eficaz da lógica de programação é fundamental para o sucesso do projeto.

A modelagem de dados assume um papel crucial na eficiência e consistência do sistema, passando da representação conceitual à implementação física do banco de dados. O uso do SQL como linguagem de gerenciamento de bancos de dados relacionais é fundamental para a manipulação eficaz dos dados.

Por fim, a gestão financeira é uma prática essencial para o sucesso do projeto, envolvendo a classificação dos custos, a análise dos custos do produto e a determinação da precificação adequada. A AUT4US se destaca no mercado ao oferecer produtos de alta qualidade e mão de obra qualificada, garantindo assim uma vantagem competitiva.

Em suma, o projeto integrado "IoT Data Streamer" representa uma solução inovadora e estrategicamente planejada para atender às demandas do mercado de IoT. A combinação de tecnologia avançada, programação orientada a objetos, lógica de programação, modelagem de dados e gestão financeira posiciona a AUT4US como uma empresa líder no setor de

automação residencial e empresarial. A abordagem abrangente e integrada deste projeto promete trazer benefícios tangíveis e duradouros para a organização.

REFERÊNCIAS

AMBLER, Scott W; JACOBSON, I.; RUMBAUGH, J. UML.**Unified Modeling language version.** September 1996.

JALOTE, P. **An Integrated Approach to Software Engineering.** New York: Springer, 2005.

KORTH, H. F.; SILBERCHATZ, A.; SUDARSHAN, S. **Sistema de banco de dados.** Rio de Janeiro: Campus, 2006.

WERNKE, R.**Análise de custos e preços de venda:** ênfase em aplicações e casos nacionais. São Paulo: Saraiva, 2005.

SILVA, Guilherme de Oliveira et al. **Alfabetização financeira versus educação financeira: um estudo do comportamento de variáveis socioeconômicas e demográficas. Revista de Gestão, Finanças e Contabilidade**, 2017.

ISSN 1983-6767

ANEXOS

Classes e Objetos:

Exemplo 1:

```
class Laboratorio:
    def __init__(self, numero):
        self.numero = numero
        self.ar_condicionado = False
        self.projetor = False
        self.iluminacao = False

    def ligar_ar_condicionado(self):
        self.ar_condicionado = True

    def ligar_projetor(self):
        self.projetor = True

    def ligar_iluminacao(self):
        self.iluminacao = True

    def desligar_ar_condicionado(self):
        self.ar_condicionado = False

    def desligar_projetor(self):
        self.projetor = False

    def desligar_iluminacao(self):
        self.iluminacao = False
```

Exemplo 2:

```
# Criando os prédios
predio_a = Predio("A")
predio_b = Predio("B")
predio_c = Predio("C")
predio_d = Predio("D")
predio_e = Predio("E")
predio_f = Predio("F")

# Criando salas no prédio A
sala_a1 = Sala(101)
sala_a2 = Sala(102)
sala_a3 = Sala(103)

# Ligando equipamentos na sala_a1
sala_a1.ligar_ar_condicionado()
sala_a1.ligar_projetor()
sala_a1.ligar_iluminacao()

# Adicionando salas ao prédio A
predio_a.adicionar_sala(sala_a1)
predio_a.adicionar_sala(sala_a2)
predio_a.adicionar_sala(sala_a3)

# Criando laboratórios no prédio B
lab_b1 = Laboratorio(201)
lab_b2 = Laboratorio(202)

# Ligando equipamentos no lab_b1
lab_b1.ligar_ar_condicionado()
lab_b1.ligar_projetor()
lab_b1.ligar_iluminacao()

# Adicionando laboratórios ao prédio B
```

```
predio_b.adicionar_laboratorio(lab_b1)
predio_b.adicionar_laboratorio(lab_b2)
```

Atributos, Métodos e Encapsulamentos:

```
class Predio:
    def __init__(self, nome):
        self.nome = nome
        self.espacos = []

    def adicionar_espaco(self, espaço):
        self.espacos.append(espaço)

    def listar_espacos(self):
        printf"Espaços no Prédio {self.nome}:")
        for espaco in self.espacos:
            print(espaco)
```

Herança:

```
class Equipamentos:
    def __init__(self, numero, tipo):
    self.numero = numero
    self.tipo = tipo
  self.ar_condicionado = False
    self.projetor = False
  self.iluminacao = False
    def ligar_ar_condicionado(self):
  self.ar_condicionado = True
    def ligar_projetor(self):
    self.projetor = True
    def ligar_iluminacao(self):
  self.iluminacao = True
    def desligar_ar_condicionado(self):
  self.ar_condicionado = False
    def desligar_projetor(self):
```

```
        self.projetor = False
        def desligar_iluminacao(self):
    self.iluminacao = False
        def __str__(self):
        return f"{self.tipo} {self.numero}"

# Subclasse que herda de Equipamentos e adiciona funcionalidades específicas
class SalaDeReuniao(Equipamentos):
        def __init__(self):
    super().__init__()
        def iniciar_reuniao(self):
    self.ligar_ar_condicionado()
    self.ligar_projetor()
    self.ligar_iluminacao()

        def encerrar_reuniao(self):
    self.desligar_ar_condicionado()
    self.desligar_projetor()
        self.desligar_iluminacao()

# Subclasse que herda de Equipamentos e adiciona funcionalidades específicas
class Escritorio(Equipamentos):
        def __init__(self):
    super().__init__()
        def iniciar_trabalho(self):
    self.ligar_ar_condicionado()
    self.ligar_iluminacao()

        def encerrar_trabalho(self):
    self.desligar_ar_condicionado()
    self.desligar_iluminacao()
```

Métodos Estáticos:

```
class EspacoEmpresarial:
    def __init__(self, numero, tipo):
        self.numero = numero
        self.tipo = tipo
        self.ar_condicionado = False
        self.projetor = False
        self.iluminacao = False

    @staticmethod
    def verificar_disponibilidade():
        return "Espaço disponível para uso."

    def __str__(self):
        return f"{self.tipo} {self.numero}"
```

Métodos Públicos e Privados:

```
class EspacoEmpresarial:
    def __init__(self, numero, tipo):
        self.numero = numero
        self.tipo = tipo
        self.ar_condicionado = False
        self.projetor = False
        self.iluminacao = False

    # Métodos públicos para ligar e desligar equipamentos
    def ligar_ar_condicionado(self):
        self._executar_ligar("Ar Condicionado")
        self.ar_condicionado = True

    def desligar_ar_condicionado(self):
        self._executar_desligar("Ar Condicionado")
```

```
        self.ar_condicionado = False

    def ligar_projetor(self):
        self._executar_ligar("Projetor")
        self.projetor = True

    def desligar_projetor(self):
        self._executar_desligar("Projetor")
        self.projetor = False

    def ligar_iluminacao(self):
        self._executar_ligar("Iluminação")
        self.iluminacao = True

    def desligar_iluminacao(self):
        self._executar_desligar("Iluminação")
        self.iluminacao = False

    def __str__(self):
        return f"{self.tipo} {self.numero}"

    # Métodos privados para realizar a operação de ligar
    def _executar_ligar(self, equipamento):
        print(f"{self.tipo} {self.numero} - {equipamento} ligado.")

    # Métodos privados para realizar a operação de desligar
    def _executar_desligar(self, equipamento):
        print(f"{self.tipo} {self.numero} - {equipamento} desligado.")

# Classe Sala e Laboratorio continuam como na resposta anterior
class Sala(EspacoEmpresarial):
    def __init__(self, numero):
        super().__init__(numero, "Sala")
```

```
class Laboratorio(EspacoEmpresarial):
    def __init__(self, numero):
        super().__init__(numero, "Laboratório")

# Exemplo de uso
sala_a1 = Sala(101)
sala_a1.ligar_ar_condicionado()
sala_a1.ligar_projetor()
sala_a1.ligar_iluminacao()
sala_a1.desligar_projetor()
```

Desenvolvimento do Front End:

- **Tela agendamento:**

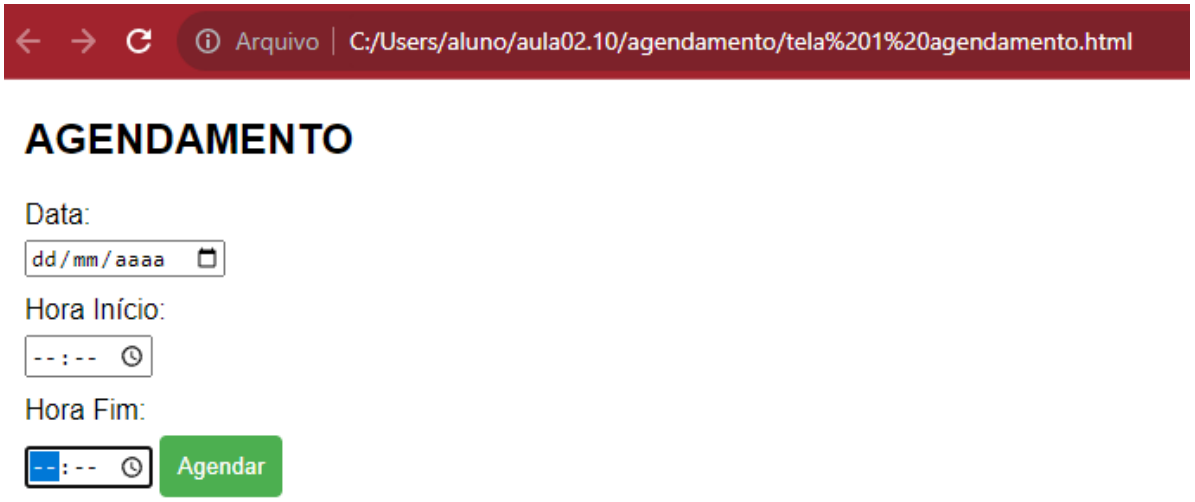
```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AGENDAMENTO</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        label {
            display: block;
            margin-bottom: 5px;
        }
        input, select {
            margin-bottom: 10px;
        }
        button {
```

```
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h2>AGENDAMENTO</h2>
    <form>
        <label for="data">Data:</label>
        <input type="date" id="data" name="data" required>

        <label for="horaInicio">Hora Início:</label>
        <input type="time" id="horaInicio" name="horaInicio" required>

        <label for="horaFim">Hora Fim:</label>
        <input type="time" id="horaFim" name="horaFim" required>
        <button type="button" onclick="agendar()">Agendar</button>
    </form>
    <script>
        function agendar() {
            // Aqui você pode adicionar lógica para processar o agendamento
                // Por exemplo, pode enviar os dados para um servidor ou realizar validações
adicionais.
            alert('Agendamento realizado com sucesso!');
        }
    </script>
</body>
```

```
</html>
```



- **Tela Ar Condicionado:**

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AGENDAMENTO</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        label {
            display: block;
            margin-bottom: 5px;
        }
        input, select {
            margin-bottom: 10px;
        }
        button {
            padding: 10px;
            background-color: #4CAF50;
```

```
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h2>AGENDAMENTO</h2>
    <form>
        <label for="data">Data:</label>
        <input type="date" id="data" name="data" required>
        <label for="horaInicio">Hora Início:</label>
        <input type="time" id="horaInicio" name="horaInicio" required>
        <label for="horaFim">Hora Fim:</label>
        <input type="time" id="horaFim" name="horaFim" required>
        <button type="button" onclick="agendar()">Agendar</button>
    </form>
    <script>
        function agendar() {
            // Aqui você pode adicionar lógica para processar o agendamento
              // Por exemplo, pode enviar os dados para um servidor ou realizar validações
adicionais.
            alert('Agendamento realizado com sucesso!');
        }
    </script>
</body>
</html>
```

- **Tela Projetor:**

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projetor</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background-color: #f4f4f4;
        }
        h2 {
            color: #333;
        }
        label {
            display: block;
            margin-bottom: 5px;
            color: #333;
```

```
        }
        input, select {
            width: 100%;
            padding: 8px;
            margin-bottom: 10px;
            box-sizing: border-box;
        }
        button {
            padding: 10px;
            margin-top: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h2>PROJETOR</h2>
    <form>
        <label for="acao">Ação:</label>
        <select id="acao" name="acao" required>
            <option value="ligar">Ligar</option>
            <option value="desligar">Desligar</option>
        </select>
        <label for="horaLigar">Ligar ás:</label>
        <input type="time" id="horaLigar" name="horaLigar" required>

        <label for="horaDesligar">Desligar ás:</label>
        <input type="time" id="horaDesligar" name="horaDesligar" required>
```

```
    <button type="button" onclick="agendar()">Agendar</button>
  </form>
  <script>
    function agendar() {
      var acao = document.getElementById('acao').value;
      var horaLigar = document.getElementById('horaLigar').value;
      var horaDesligar = document.getElementById('horaDesligar').value;
      // Adicione aqui a lógica para processar os agendamentos
      // Por exemplo, pode enviar os dados para um servidor ou realizar outras ações.
        alert('Agendamentos realizados com sucesso:\nAção: ' + acao + '\nHorário de
Ligar: ' + horaLigar + '\nHorário de Desligar: ' + horaDesligar);
    }
  </script>
</body>
</html>
```

- **Tela Luz:**

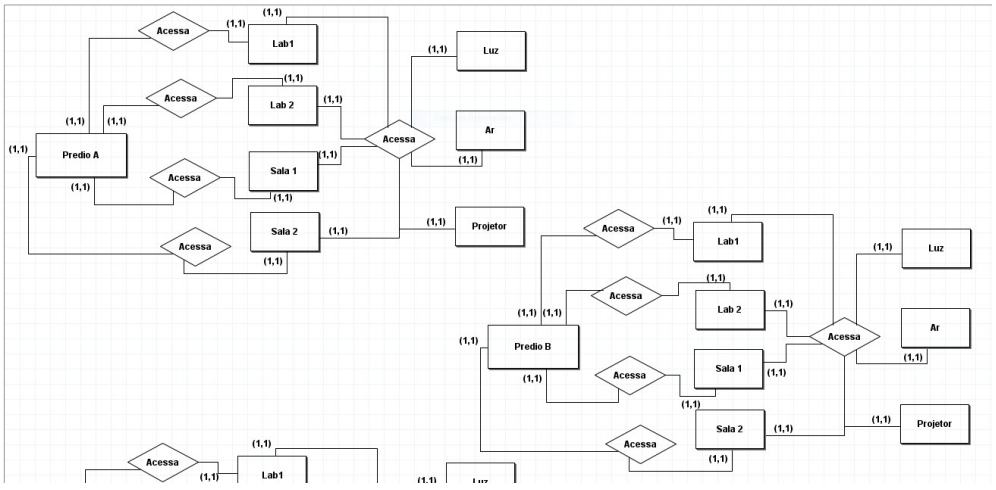
```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Luz</title>
```

```
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background-color: #f4f4f4;
        }
        h2 {
            color: #333;
        }
        label {
            display: block;
            margin-bottom: 5px;
            color: #333;
        }
        input, select {
            width: 100%;
            padding: 8px;
            margin-bottom: 10px;
            box-sizing: border-box;
        }
        button {
            padding: 10px;
            margin-top: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
```

```
<body>
    <h2>LUZ</h2>
    <form>
        <label for="acao">Ação:</label>
        <select id="acao" name="acao" required>
            <option value="ligar">Ligar</option>
            <option value="desligar">Desligar</option>
        </select>
        <label for="horaLigar">Ligar ás:</label>
        <input type="time" id="horaLigar" name="horaLigar" required>
        <label for="horaDesligar">Desligar ás:</label>
        <input type="time" id="horaDesligar" name="horaDesligar" required>
        <button type="button" onclick="agendar()">Agendar</button>
    </form>
    <script>
        function agendar() {
            var acao = document.getElementById('acao').value;
            var horaLigar = document.getElementById('horaLigar').value;
            var horaDesligar = document.getElementById('horaDesligar').value;
            // Adicione aqui a lógica para processar os agendamentos
            // Por exemplo, pode enviar os dados para um servidor ou realizar outras ações.
              alert('Agendamentos realizados com sucesso:\nAção: ' + acao + '\nHorário de
Ligação: ' + horaLigar + '\nHorário de Desligamento: ' + horaDesligar);
        }
    </script>
</body>
</html>
```

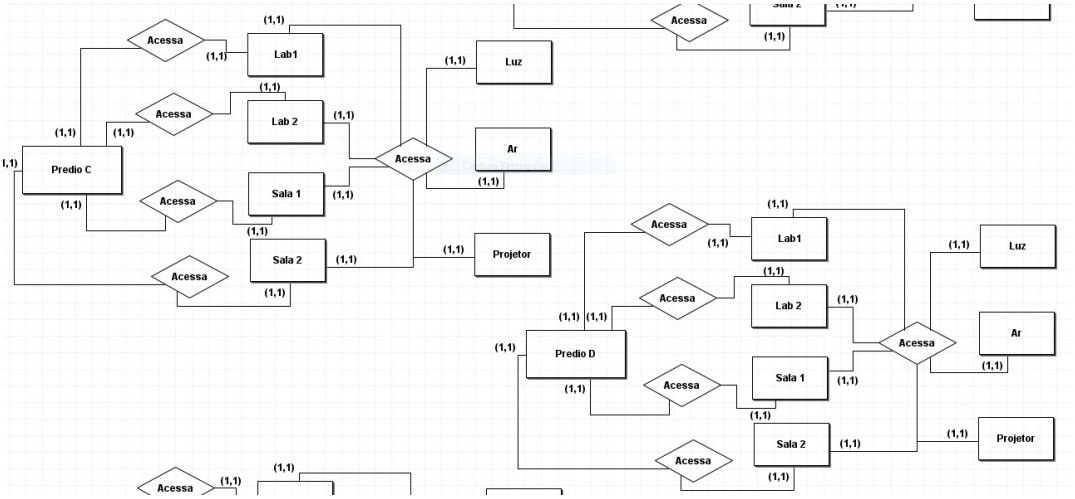
Modelo Conceitual:

Imagem 1:



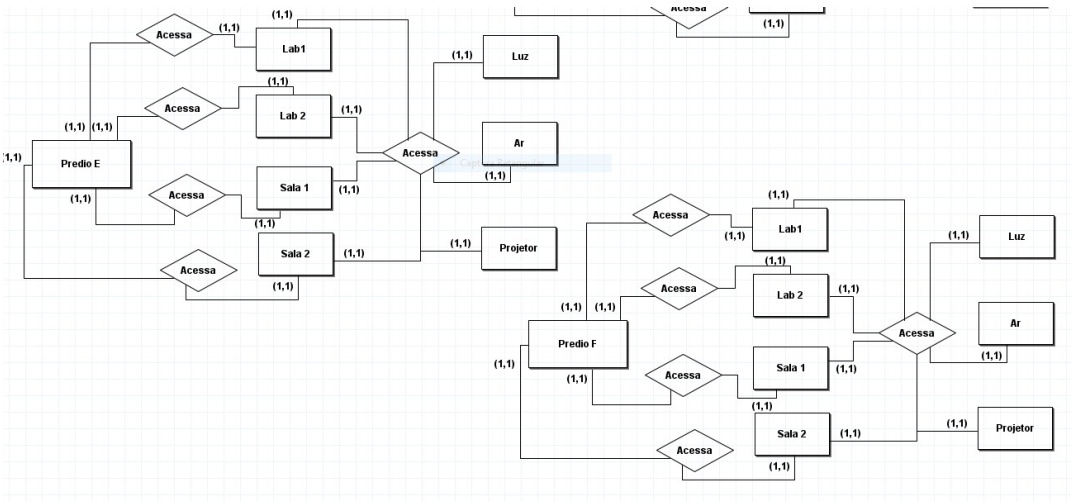
Fonte: autores.

Imagem2 :



Fonte: autores.

Imagem 3:

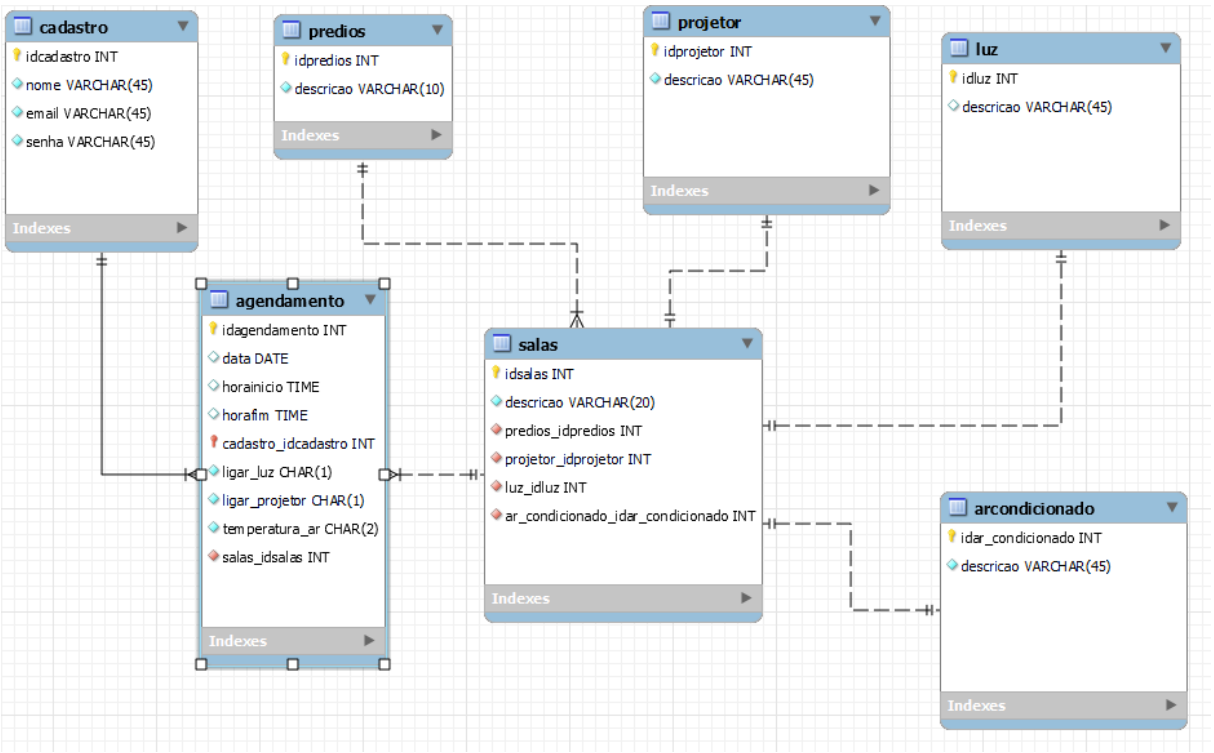


Fonte: autores.

ISSN 1983-6767

Modelo Lógico e Físico:

Imagem 4:



Fonte: autores.

Imagem 5:

	idagendamento	data	horainicio	horafim	cadastro_idcadastro	ligar_luz	ligar_projetor	temperatura_ar	salas_idsalas
▶	8	2023-10-26	19:40:00	22:30:00	1	1	1	1	1
	9	2023-10-26	19:40:00	22:30:00	1	1	1	18	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fonte: autores.

Imagem 6:

	idar_condicionado	descricao
▶	1	Ligado
	2	Ligado
	3	Ligado
	4	Ligado
*	NULL	NULL

Fonte: autores.

Imagem 7:

	idluz	descricao
▶	1	Luz
	2	Luz
	3	Luz
	4	Luz

Fonte: autores.

Imagem 8:

	idprojetor	descricao
▶	1	EPSON
	2	EPSON
	3	EPSON
	4	EPSON
	5	EPSON

Fonte: autores.

Imagem 9 :

	idsalas	descricao	predios_idpredios	projetor_idprojetor	luz_idluz	ar_condicionado_idar_condicionado
▶	1	Sala 1	1	1	1	1
	3	Sala 1	1	1	1	1
	4	Sala 1	1	1	1	1

Fonte: autores.

Imagem 10:

	idpredios	descricao
▶	1	Prédio
	2	Nome
	3	Nome
	4	Nome
	5	Nome
	6	Nome

Fonte: autores.

Imagem 11:

	idcadastro	nome	email	senha
▶	1	João da Silva	joao@example.com	senhasegura123
	2	João da Silva	joao@example.com	senhasegura123
	3	João da Silva	joao@example.com	senhasegura123
	4	João da Silva	joao@example.com	senhasegura123
	5	João da Silva	joao@example.com	senhasegura123
	6	João da Silva	joao@example.com	senhasegura123

Fonte: autores.

SQL:

```
/*Tabela Cadastro*/
INSERT INTO cadastro (nome, email, senha)
VALUES ('João da Silva', 'joao@example.com', 'senhasegura123');

/*Tabela Predios*/
INSERT INTO predios (descricao)
VALUES ('Nome');

/*Tabela Salas*/
INSERT INTO salas (idsalas, descricao, predios_idpredios, projetor_idprojetor, luz_idluz, ar_condicionado_idar_condicionado)
VALUES (1,'Sala 1', 1, 1, 1, 1);

/*Tabela Projetor*/
INSERT INTO projetor (descricao)
VALUES ('EPSON');

/*Tabela Luz*/
INSERT INTO luz (descricao)
VALUES ('Luz');

/*Tabela Ar-Condicionado*/
```

```
INSERT INTO arcondicionado (descricao)
VALUES ('LG');

/*Tabela Agendamento*/
INSERT INTO agendamento (data, horainicio, horafim, cadastro_idcadastro, ligar_luz, ligar_projetor, temperatura_ar, salas_idsalas)
VALUES ('2023-10-26', '19:40:00', '22:30:00', 1, 1, 1, 18, 1);


SELECT * FROM cadastro;
SELECT * FROM predios;
SELECT * FROM salas;
SELECT * FROM projetor;
SELECT * FROM luz;
SELECT * FROM arcondicionado;
SELECT * FROM agendamento;
```