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:

Gabriel Meneghetti Zanardo, RA 23000471
Jorge Luis Santiciolli Filho, RA 23000846
Juliano Salles Amaral, RA 23000769
Kevilyn Marinho de Lima, RA 23000466
Leonardo Scatolin, RA 23000229
Wesley de Jesus Oliveira , RA 23000285

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2023

SUMÁRIO

ISSN 1983-6767

1. INTRODUÇÃO

A Sensify é uma empresa de tecnologia biométrica que se dedica ao desenvolvimento de soluções de autenticação de identidade baseadas em biometria, como reconhecimento facial, de voz e de impressão digital. Sua principal função no mercado é fornecer soluções de segurança biométrica para empresas e organizações que precisam de autenticação segura e eficiente de identidade. Hoje em dia, já sabemos que a ampliação de biometria pode ter colaborado para reduzir filas.

> O mundo desenvolvido parece estar entrando em uma nova fase das tecnologias de identificação. Trata-se agora das identificações biométricas, baseadas na coleta de informações altamente singulares dos indivíduos, como sua íris ou impressão digital. A expansão recente desse novo mercado mostra como ele está influenciando os padrões de relacionamento de diversas empresas de serviços com seus clientes, como bancos e supermercados. (Kulpas, 2006, p. 41)

Com o crescente aumento das preocupações com a segurança cibernética e a privacidade dos dados pessoais, a tecnologia biométrica tem sido cada vez mais utilizada como uma solução de autenticação segura e conveniente.

Já sabemos que o risco de vazamento de dados faz aumentar a procura por empresas de cibersegurança. Segundo dados do Jornal Nacional publicados em matéria do G1, só em 2021, milhões de brasileiros tiveram seus dados vazados ilegalmente. A preocupação com ataques cibernéticos está mexendo com o planejamento financeiro de muitas empresas, que pretendem aumentar o investimento em cibersegurança.

Oferecemos soluções de biometria que ajudam a proteger os dados e as informações sensíveis dos clientes, além de fornecer uma experiência de usuário mais eficiente e amigável. Dessa forma, a Sensify desempenha um papel importante na indústria de tecnologia, ajudando a garantir a segurança e privacidade das informações confidenciais.

> Nos Estados Unidos e na Europa, diversos varejistas testam tecnologias de reconhecimento biométrico como forma de validar pagamentos. O método evita fraudes e pode ser uma comodidade para os consumidores, mas também levanta uma polêmica sobre a invasão da vida privada que a tecnologia pode gerar. (Kulpas, 2006, p. 42)

ISSN 1983-6767

2. DESCRIÇÃO DA EMPRESA

Unifeob Fundação de Ensino Octavio Bastos, 59.764.555/0001-52, Av. Dr.Otavio da Silva Bastos,2439 - Jardim Nova São João, São João da Boa Vista - SP, 13874-149

Criada no Centro Universitário da Fundação de Ensino Octávio Bastos, UNIFEOB, a Sensify é uma empresa que tem como missão impulsionar a eficiência por meio do uso inteligente e ético da tecnologia de reconhecimento facial, óptico e biométrico. Acreditamos no potencial dessas tecnologias para simplificar processos coletivos, como chamadas em salas de aula e registros de ponto em ambientes industriais. Para tal, desenvolvemos com excelência o “Herefy”, nosso produto principal, que resolve com excelência e confiabilidade os desafios relacionados a essas áreas, otimizando e aprimorando procedimentos cotidianos.

Acreditamos firmemente que o progresso tecnológico deve andar de mãos dadas com a responsabilidade ética, buscando não apenas a eficiência operacional, mas também a segurança e privacidade dos dados. Por isso, investimos continuamente em pesquisa e desenvolvimento, assegurando que nossas soluções atendam aos mais altos padrões de qualidade e proteção aos usuários.

A Sensify, orgulhosamente fundada em uma tradição acadêmica, está comprometida em promover inovação e excelência em suas soluções, visando um impacto positivo e transformador no cotidiano de empresas e instituições que buscam aprimorar seus processos por meio da tecnologia de reconhecimento.

ISSN 1983-6767

3. PROJETO INTEGRADO

3.1 PROGRAMAÇÃO ORIENTADA A OBJETO

A Programação Orientada a Objetos é uma abordagem de programação que se baseia na ideia de modelar objetos e ações que ocorrem no mundo real. Isso torna a compreensão e a resolução de problemas no mundo da programação mais intuitivas e eficazes, uma vez que os objetos e ações em um programa se assemelham a objetos e ações na vida real.

> De fato, algumas pessoas consideram Smalltalk o modelo base para uma linguagem de programação puramente orientada a objetos. Uma linguagem orientada a objetos deve fornecer suporte para três recursos chave de linguagem: tipos de dados abstratos, herança e vinculação dinâmica de chamadas a métodos (SEBESTA, 2018, p. 547).

Para este projeto, foi utilizada a linguagem de programação Python, que possui uma sintaxe simples e atende a todos os pilares da programação orientada a objetos. Esses pilares incluem classes, encapsulamento, polimorfismo e herança. Através do uso da linguagem Python, foi possível implementar esses conceitos de maneira eficiente e eficaz, permitindo a criação da parte lógica do projeto.

3.1.1 CLASSES E OBJETOS

Classes na programação orientada a objetos podem ser definidas como a estrutura principal antes da criação de um objeto. Dentro delas, estão presentes os atributos (variáveis) e os métodos (funções). De acordo com Booch, Rumbaugh e Jacobson (2006, p. 45), “uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica”. Por fim, os objetos são instâncias das classes e, quando inicializados, podem acessar os atributos e métodos da classe.

Para um melhor entendimento sobre classes e objetos, foi desenvolvido um exemplo prático de criação da classe “Aluno” e sua instância.

```python
class Aluno:
    def __init__(self, nome, ra):
        self.nome = nome
        self.ra = ra


aluno = Aluno( nome: "Aluno 1",  ra: "11000222")
```

A classe 'Aluno' consiste em um construtor (responsável por inicializar a classe sempre que um objeto é instanciado) e seus atributos, que, neste caso, são o nome e o RA do aluno. Após a criação da classe, um objeto foi instanciado, e seus parâmetros, incluindo o nome e o RA do aluno, foram passados para a criação do objeto.

3.1.2 ATRIBUTOS, MÉTODOS, ENCAPSULAMENTO E HERANÇA

Os atributos são responsáveis por definir o estado inicial de um objeto, podendo ser de diferentes tipos de dados, como strings, números (sejam inteiros ou não) e valores booleanos (verdadeiro ou falso). Por outro lado, os métodos são funções presentes dentro da classe que podem ser facilmente acessadas através da instância do objeto. Os métodos permitem que os objetos executem ações quando solicitados pelo usuário.

```python
class Aluno:
    def __init__(self, nome, ra):
        self.nome = nome
        self.ra = ra

    def nome_aluno_ra_aluno(self):
        print(f"O aluno se chama {self.nome} e possui RA {self.ra}")


aluno = Aluno( nome: "Aluno 1",  ra: "11000222")
aluno.nome_aluno_ra_aluno()
```

Como exemplo, criamos a classe "Aluno" que contém os atributos "nome" e "RA", juntamente com um método para exibir o nome e o RA do aluno. Em seguida, instanciamos a classe e acessamos essa instância para executar o método de exibição do nome e do RA do aluno.

Além disso, podemos 'proteger' os atributos dentro da classe utilizando o encapsulamento, o que exige que um atributo só possa ser acessado ou alterado por meio de métodos.

```python
class Aluno:
    def __init__(self, nome, ra):
        self.__nome = nome
        self.__ra = ra

    @property
    def nome(self):
        return self.__nome

    @nome.setter
    def nome(self, novo_nome):
        self.__nome = novo_nome

    @property
    def ra(self):
        return self.__ra

    @ra.setter
    def ra(self, novo_ra):
        self.__ra = novo_ra

    def nome_aluno_ra_aluno(self):
        print(f"O aluno se chama {self.nome} e possui RA {self.ra}")


aluno = Aluno( nome: "Aluno 1",  ra: "11000222")
aluno.nome_aluno_ra_aluno()
```

Na mesma classe de exemplo, os atributos “nome” e “RA” foram definidos como privados, utilizando dois underlines após o “self” ao definir os atributos. Isso significa que esses atributos só podem ser acessados e alterados por meio de seus Getters e Setters. Os Getters e Setters foram definidos usando o decorador @property, que permite o acesso aos atributos por meio de métodos para retornar seus valores, e o decorador @nome.setter, que possibilita definir novos valores para esses atributos.

Outro pilar fundamental na programação orientada a objetos é a herança. Com a herança, podemos definir uma classe base e permitir que outras classes futuras herdam os atributos e métodos da classe base. Isso reduz significativamente a repetição de código e contribui para um design mais harmonioso e eficiente.

```python
class Cadastro:
    def __init__(self, nome):
        self.nome = nome


class Aluno(Cadastro):
    def __init__(self, nome, ra):
        super().__init__(nome)
        self.ra = ra


class Professor(Cadastro):
    def __init__(self, nome, especializacao):
        super().__init__(nome)
        self.especializacao = especializacao


aluno = Aluno( nome: "Aluno 1",  ra: "11000222")
professor = Professor( nome: "Professor 1",  especializacao: "POO")
```

Como exemplo, temos uma classe base que possui apenas o atributo “nome”. Posteriormente, criamos duas classes que herdam a classe base “Cadastro”. Após a herança, utilizamos o “super” para estabelecer a relação entre as classes. Isso é útil quando há a necessidade de herdar parâmetros específicos de uma classe pai, que, neste caso, se resume ao atributo “nome”. Utilizando o conceito de herança, temos também o polimorfismo. Em Python, o polimorfismo é a capacidade que uma subclasse tem de ter métodos com o mesmo nome de sua superclasse, e o programa sabe qual método deve ser invocado, especificamente (da superclasse ou da subclasse). Ou seja, o objeto tem a capacidade de assumir diferentes formas (polimorfismo).

```python
class Cadastro:
    def __init__(self, nome):
        self.nome = nome

    def obter_informacoes(self):
        return f"Nome: {self.nome}"


class Aluno(Cadastro):
    def __init__(self, nome, ra):
        super().__init__(nome)
        self.ra = ra

    def obter_informacoes(self):
        return super().obter_informacoes() + f", RA: {self.ra}"


class Professor(Cadastro):
    def __init__(self, nome, especializacao):
        super().__init__(nome)
        self.especializacao = especializacao

    def obter_informacoes(self):
        return super().obter_informacoes() + f", Especialização: {self.especializacao}"
```

Neste exemplo, temos uma classe base chamada “Cadastro”, que define um método chamado “obter_informacoes”. Em seguida, criamos duas classes, “Aluno” e “Professor”, que herdam da classe “Cadastro”. Cada uma dessas classes possui uma implementação própria do método “obter_informacoes”, que retorna informações específicas da classe.

```
aluno = Aluno( nome: "Aluno 1",  ra: "11000222")
professor = Professor( nome: "Professor 1",  especializacao: "Lógica de Programação")

cadastros = [aluno, professor]

for cadastro in cadastros:
    print(cadastro.obter_informacoes())
```

Aqui está onde o polimorfismo entra em jogo: criamos uma lista chamada “cadastros” que contém objetos de ambas as classes “Aluno” e “Professor”. Em seguida, percorremos essa lista e chamamos o método “obter_informacoes” em cada objeto, sem nos preocupar se o objeto é um aluno ou um professor. O Python automaticamente chama o método apropriado para cada objeto, graças ao polimorfismo.

3.1.3 MÉTODOS ESTÁTICOS, PÚBLICOS E PRIVADOS

Um método estático em Python é uma função que pertence a uma classe, não a uma instância específica da classe. Isso significa que você pode chamar um método estático diretamente na classe utilizando o decorador “@staticmethod”, sem a necessidade de criar um objeto (instância) dessa classe. Métodos estáticos são usados quando você tem uma funcionalidade que não depende do estado da instância, mas ainda está relacionada à classe em si.

```
class Aluno:
    @staticmethod
    def somar(a, b):
        return a + b


aluno = Aluno.somar( a: 3,  b: 5)
```

Em Python, todos os métodos de uma classe são, por padrão, públicos. Isso significa que eles podem ser chamados de qualquer lugar do código, tanto de dentro da classe como de fora dela. Métodos públicos são acessíveis e utilizados para interagir com as instâncias da

classe. Eles representam a interface pública da classe e são destinados a serem chamados por outros objetos ou partes do programa.

```
class Aluno:
    def presenca(self):
        print("O aluno está presente.")


aluno = Aluno()
aluno.presenca()
```

Embora Python não tenha métodos estritamente privados como algumas outras linguagens, existe uma convenção para criar métodos privados. A convenção é usar um sublinhado duplo (por exemplo, __nome_do_metodo) no início do nome do método para indicar que ele deve ser tratado como privado e não deve ser acessado diretamente de fora da classe. Essa convenção é uma recomendação para programadores e serve para manter a integridade da classe e evitar que métodos internos sejam chamados de forma inadequada.

```
class Aluno:
    def __init__(self, nome):
        self.__nome = nome

    def __exibir_nome(self):
        return self.__nome


aluno = Aluno("Aluno 1")
nome_do_aluno = aluno.__exibir_nome()
```

No entanto, este código não irá funcionar porque o método __exibir_nome é privado, e tentar acessá-lo diretamente de fora da classe causa um erro de atributo não encontrado, portanto, é necessário criar um método para executar esse método privado:

```python
class Aluno:
    def __init__(self, nome):
        self.__nome = nome

    def __exibir_nome(self):
        return self.__nome

    def obter_nome(self):
        return self.__exibir_nome()


aluno = Aluno("Aluno 1")
nome_do_aluno = aluno.obter_nome()
```

3.2 LÓGICA DE PROGRAMAÇÃO

Na fase de implementação da lógica de programação para nosso projeto integrado, optamos por utilizar a linguagem JavaScript. JavaScript é uma linguagem de programação amplamente utilizada para desenvolvimento web, conhecida por sua versatilidade e capacidade de interação dinâmica com elementos HTML.

De acordo com Nicholas C. Zakas (2014, p.7):

> Em JavaScript você jamais irá precisar criar uma definição de classe, importar um pacote ou incluir um arquivo de cabeçalho. Basta começar a codificar com os tipos de dados que você quiser, e eles poderão ser agrupados de várias maneiras. Certamente, é possível escrever um programa JavaScript de modo procedural, mas o verdadeiro poder da linguagem surge quando você tira vantagem de sua natureza orientada a objetos.

A utilização de JavaScript na lógica de programação do nosso projeto integrado permitiu-nos explorar os princípios da programação orientada a objetos de maneira eficaz. A flexibilidade da linguagem e sua capacidade de interagir dinamicamente com elementos da web tornaram-na uma escolha adequada para a implementação de nossa lógica de programação, mantendo a coesão e a eficiência em nossas soluções.

3.2.1 CONCEITOS FUNDAMENTAIS DO DESENVOLVIMENTO DE SOFTWARE

Na lógica de programação existem alguns conceitos fundamentais para o desenvolvimento de um software, dentre eles estão os algoritmos, variáveis, tipos de dados, funções, estruturas condicionais e operadores.

Darei aqui uma breve explicação sobre cada um e de como abordamos esses conceitos em alguns trechos de códigos do tópico anterior.

Algoritmos são conjuntos de instruções bem definidas e ordenadas, usadas para resolver um problema ou executar uma tarefa. Eles consistem em uma sequência lógica de passos que devem ser seguidos para alcançar um resultado específico.

No método nome_aluno_ra_aluno(), há um algoritmo simples para exibir na tela o nome e o RA do aluno.

Variáveis são espaços de memória designados para armazenar dados. Os tipos de dados determinam o tipo de informação que uma variável pode conter, como números inteiros, decimais, strings (texto), booleanos (verdadeiro ou falso) etc.

Utilização de variáveis como self.__nome e self.__ra, armazenando informações dos alunos. São exemplos de variáveis utilizando tipos de dados como strings para nomes e números para RA.

Funções basicamente são blocos de código reutilizáveis que executam uma tarefa específica. Elas podem receber entradas (parâmetros), processar esses dados e retornar um resultado. Os métodos na classe Aluno (como nome, ra, nome_aluno_ra_aluno) representam funções que executam tarefas específicas relacionadas aos dados do aluno.

Já as estruturas condicionais, como 'if', 'else if' e 'else', permitem que o código tome decisões baseadas em condições. Os operadores (aritméticos, lógicos e relacionais) são usados para realizar operações em variáveis e valores. Embora não sejam explicitamente usados neste código, normalmente, estruturas condicionais e operadores seriam utilizados para validar ou processar os dados antes de serem armazenados nos atributos do aluno.

3.2.2 DESENVOLVIMENTO DE APLICAÇÕES DESKTOP

Ao desenvolver este projeto, buscamos criar uma experiência visualmente atrativa e funcional para os usuários. A estrutura do código HTML é simples, focada na melhor experiência e intuitividade do usuário. Para a estilização, utilizamos principalmente o CSS para definir o design e a disposição dos elementos na página.

A ideia principal foi criar uma interface intuitiva e limpa para facilitar o processo de cadastro dos usuários. A barra de navegação foi projetada para ser simples e clara, enquanto a seção de login destaca-se com um título marcante. O mesmo padrão foi utilizado nas demais páginas.

Os estilos são cuidadosamente ajustados para garantir uma boa legibilidade dos textos e uma fácil identificação dos campos a serem preenchidos ou alterados. As fontes utilizadas também foram escolhidas com atenção para garantir uma aparência moderna e profissional.

Ao apresentar esse projeto, meu objetivo principal seria demonstrar a capacidade de criar uma interface responsiva e estilizada para uma aplicação de desktop. Destacaria a importância de uma abordagem centrada no usuário, visando a usabilidade e a estética, elementos fundamentais para uma boa experiência do usuário.

3.3 MODELAGEM DE DADOS

A modelagem de dados foi a solução encontrada para a alta disponibilidade de dados que temos atualmente e com ela é possível receber, organizar e analisar esses dados para qualquer finalidade desejada. O projeto **IoT Data Streamer** é centrado principalmente nos conceitos de modelagem de dados como MER (Modelo Entidade Relacionamento) que possibilita trazer situações do mundo real para um banco de dados e resolver seus problemas reais no mundo digital.

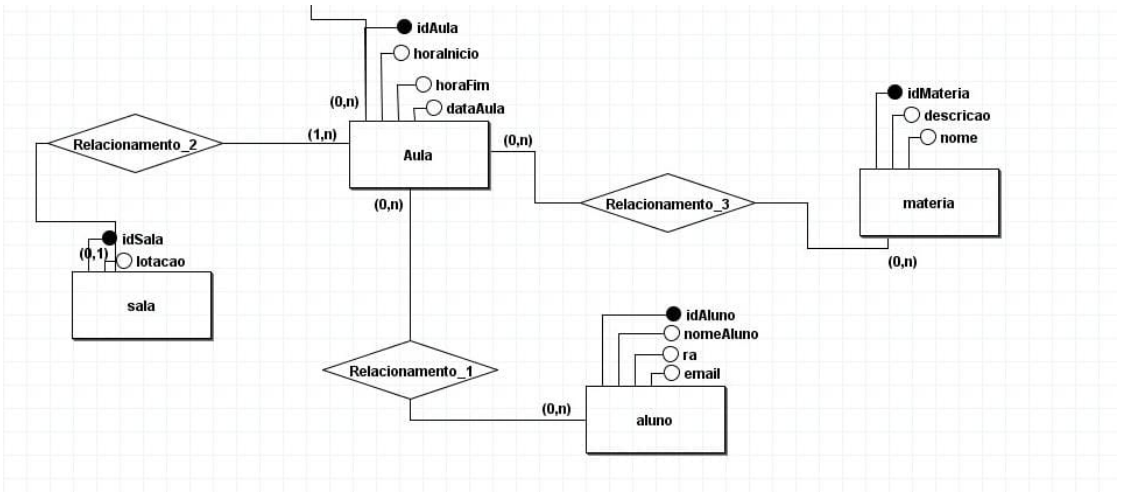
Este projeto, por exemplo, pretende resolver o atraso no processo de chamada em uma universidade e utilizando o banco de dados juntamente com um dispositivo Iot somos capazes não só de solucionar este problema como também gerenciar esses dados gerados pelo dispositivo para serem apresentados aos usuários em forma de relatórios para que eles possam fazer análise desses dados posteriormente.

3.3.1 MODELO CONCEITUAL

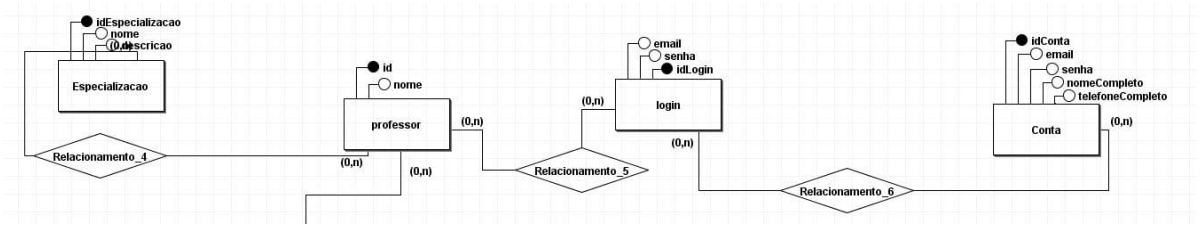
O principal problema encontrado na hora de criar o banco de dados foi entender de que forma seria feito o sistema de chamada. Poderia ser feita uma entidade com o nome “chamada” que iria se relacionar com a entidade “aluno” e “professor” ou poderíamos utilizar uma relação N:N entre “aluno” e “aula” que criaria uma tabela que teria o ID do aluno, ID da aula e um outro atributo que apenas computaria a presença do aluno. Por questões de facilidade e praticidade, e por instrução do professor Max Streicher Vallim, foi escolhida a segunda opção.

Usando essa tabela criada através da relação N:N temos mais liberdade para alocar os dados de forma mais conveniente e deixar o processo de criar os inserts mais simples.

Segue exemplo da relação de chamada no modelo conceitual estruturado através do aplicativo brModelo:

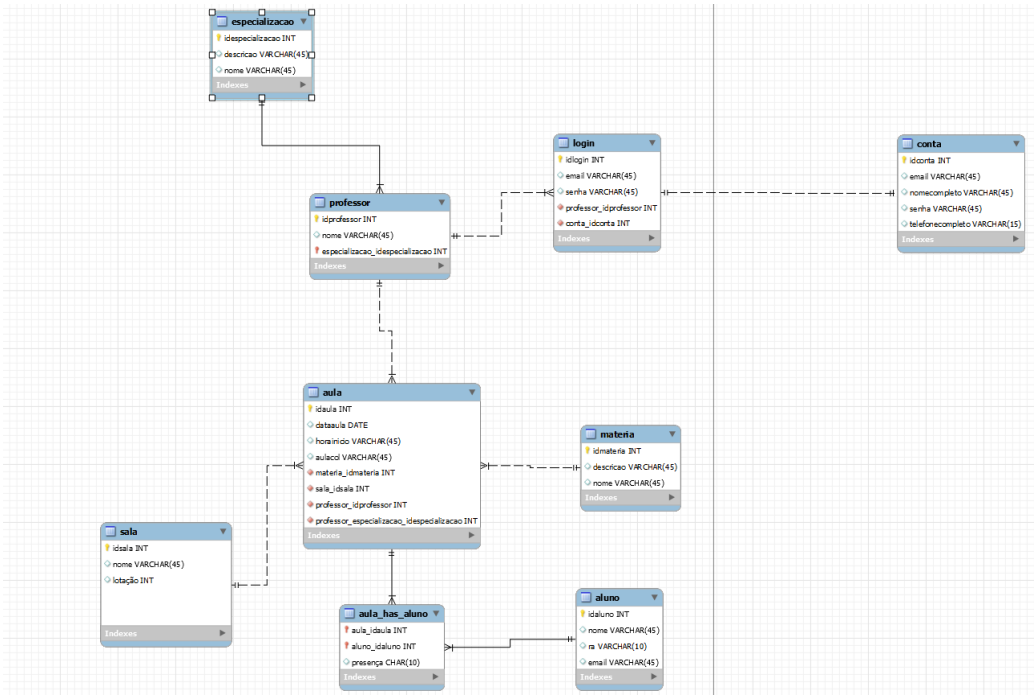


Como o projeto é a criação de um sistema desktop que recebe dados de um dispositivo IoT que auxilia no processo de chamada nas aulas, foi também necessário criar um cadastro para os alunos e professores e, por extensão, um login para que o usuário possa acessar os dados através de um app ou pelo próprio desktop.



3.3.2 MODELO LÓGICO E FÍSICO

O modelo lógico, feito no MySQL Workbench, é apenas uma tradução do modelo conceitual para um software capaz de criar o banco de dados como queremos. Todas as entidades e atributos do modelo conceitual porém mais direcionados a como eles serão no modelo físico, todos os relacionamentos já tem suas cardinalidades definidas, chaves primárias definidas e atributos tipados, nesse estado só falta aplicar o forward engineer no workshop que o modelo físico será gerado pelo próprio MySQL Workshop restando apenas popular o banco através dos comandos SQL como Insert, Update, Delete e Select. (explicar cada tabela do modelo lógico)



Começando pelas tabelas mais periféricas temos “especializacao” que contém as informações de formação e especialização dos professores; “conta” que possui as informações da conta do usuário que serão visíveis apenas para a instituição; “sala” é a tabela que receberá as informações das salas existentes na instituição e “aluno” que terá as informações de todos os alunos.

A tabela “login” será utilizada para o usuário conseguir entrar no sistema, ela recebe as chaves estrangeiras de “professor” e “conta” para relacionar todas as informações no código do aplicativo; “Professor” recebe apenas a chave estrangeira de “especialização” para que a instituição possa atribuir uma especialização a cada professor na hora do cadastro.

A entidade “Aula” é uma das mais importantes do banco pois ela é a que mais recebe chaves estrangeiras e é ela que fará a relação de todas as outras entidades e posteriormente se relacionar com “aluno” em uma relação N:N que será utilizada para fazer o controle da presença dos alunos pelo sistema.

3.3.3 SQL

Com o banco de dados ativo resta apenas popular ele com o comando insert into. Como as entidades do banco estão todas relacionadas é necessário começar os inserts nas entidades que não recebem atributos de outras entidades, nesse caso são: “sala”, “aluno”, “matéria” , “conta” e “especialização”.

Conforme o Anexo - A, foram adicionados 5 alunos, 4 salas, 4 matérias e 4 contas que posteriormente serão relacionadas com outras entidades para formar um banco de dados funcional. Com isso feito já é possível fazer os inserts das entidades que recebem atributos das outras.

Como foi possível ver nos Anexos B e C além de ser necessário informar dados que preenchem os atributos de cada entidade também foi necessário inserir o identificador das entidades que mantém uma relação, por isso foi preciso fazer os inserts das outras tabelas primeiro, isso fica muito claro nos inserts da tabela “aula” que recebe a data e horário que a aula está marcada e três identificadores diferentes: um para relacionar com qual matéria será aplicada nessa aula, outro para relacionar com qual sala será utilizada para essa aula e por fim qual professor irá lecionar.

Por último, foram feitos os inserts da tabela “aula_has_saluno” ilustrados no Anexo D, que é a tabela que contém a relação entre alunos, aula e um atributo que conta apenas a presença do aluno.

Desta forma se insere o identificador da aula e do aluno, o último atributo receberá nulo até o dia da aula que será feita a chamada e, através de um comando update, o atributo irá receber “Presente” ou “Ausente” que será posteriormente consultada pelo sistema.

Os selects vêm após os inserts pois eles servem para gerar uma consulta dos dados inseridos, através deles é possível criar diversas tabelas para conferir se os dados estão corretos e posteriormente exibir esses dados para o usuário.

```
84 •  select idaula as ID, dataaula as "Data", horainicio as Inico, horafim as Fim, materia.nome as Materia, sala.nome as Sala, professor.nome as Professor from aula
85    inner join materia on ( aula.materia_idmateria = materia.idmateria)
86    inner join sala on (aula.sala_idsala = sala.idsala)
87    inner join professor on (aula.professor_idprofessor = professor.idprofessor)
88    where professor.nome = "Max Streicher Vallim";
```

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

ID	Data	Inico	Fim	Materia	Sala	Professor
1	2023-10-01	19:40	22:20	Modelagem de dados	Lab. informática 1	Max Streicher Vallim
2	2023-10-02	19:40	22:20	Modelagem de dados	Lab. informática 2	Max Streicher Vallim
3	2023-10-03	19:40	22:20	Modelagem de dados	Lab. informática 3	Max Streicher Vallim
4	2023-10-04	19:40	22:20	Modelagem de dados	Lab. informática 4	Max Streicher Vallim
5	2023-10-05	19:40	22:20	Modelagem de dados	Lab. informática 2	Max Streicher Vallim

É possível criar uma tabela feita através do comando select que mostre os dados relacionados de maneira conveniente para o usuário. O comando inner join foi usado juntamente ao select para exibir atributos de diversas entidades para formar uma tabela.

Por último, falta apenas a utilização do comando update que irá atualizar os dados do banco conforme o usuário utilize o sistema e haja a necessidade de atualizar algum valor quando é feita a chamada em uma aula o aluno irá receber a presença ou falta.

```
130
131     -- aulas após dia 01-10 Jorge:
132 •   update aula_has_aluno
133     set presença="Presente"
134     where aluno_idaluno = 1 and aula_idaula = 1;
```

No exemplo acima foi alterado apenas o atributo presença do aluno de id 1 da aula de id 1 porém todos os alunos que estão designados para aquela aula receberão um novo valor de presença e agora quando for feita outra consulta no banco através de um select todos os alunos que foram na aula estarão com presença e os que não foram receberão falta.

	IDAula	aluno	Presença
▶	1	Jorge Luis	Presente
	6	Wesley	Presente
	11	Gabi	Presente
	16	Kevin	Presente
	1	Leonardo	Ausente

3.4 GESTÃO FINANCEIRA

A Gestão Financeira é o processo de administrar o dinheiro de uma empresa de forma eficiente, abrangendo atividades como orçamento, planejamento, controle de gastos, investimentos e obtenção de financiamento. Ela desempenha um papel crucial nas tomadas de decisões dentro da empresa, uma vez que uma boa gestão financeira permite o crescimento sustentável e a estabilidade financeira de uma organização, contribuindo para seu sucesso de longo prazo.

De acordo com Schier (2006, p.25):

> Contabilidade de custos é uma técnica utilizada para identificação e mensuração dos custos dos produtos em todo o processo produtivo, aquisição de mercadorias para revenda de custos para prestação de serviços, além de uma forma para proporcionar seu controle.

Além disso, a gestão financeira, segundo Oliveira (2003), é o ato de conduzir e obter resultados satisfatórios na empresa através de objetivos propostos. Para que estes sejam alcançados, pode-se contar com algumas práticas que a grande maioria dos gestores já utiliza nos setores financeiros, como: planejamento, orçamento, fluxo de caixa, etc.

Dessa forma, resolvemos fazer a gestão financeira de nosso produto, o Herefy, que, de forma simplificada, é um método de chamada automática com cadastro de digitais e recognição facial. Coletando todas as informações necessárias para tal gestão, como, por exemplo, seus custos diretos, como os materiais para sua construção, e seus custos indiretos, como a quantidade de energia elétrica, fizemos todas as relações de preço e classificamos como possível uma mudança de preço nos produtos quando vendidos, uma vez que alguns estão um pouco abaixo do preço hoje fornecido.

3.4.1 CLASSIFICAÇÃO DOS CUSTOS

Custo é todo o gasto utilizado na produção de um produto ou serviço e todo gasto relacionado com a produção pode ser considerado como gasto. Dito isso, eles podem ser diferenciados em indireto, direto e variável ou fixo.

Custo fixo é todo gasto que independente da venda continuará o mesmo, como o aluguel e conta internet. Estes gastos serão diluídos no valor do produto mas as vendas não irão de maneira alguma influenciar no valor final.

Custo variável é o gasto que irá variar de acordo com as vendas ou imprevistos ocorridos na produção do produto. Utilizando o produto da Sensify como exemplo, é possível citar os componentes que compõem um m o produto final e se o valor final de um leitor Biométrico irá aumentar ou diminuir dependendo da produção dos sensores.

Custo direto é aquele determinado necessariamente pela produção, como a compra de matéria-prima e a mão de obra para produção. Trata-se de um valor fácil de determinar, sem a necessidade de muitos cálculos.

Custo indireto, também, é um gasto determinado pela produção, mas a relação muitas vezes não é assim tão clara. É difícil mensurar, por exemplo, quanto uma fábrica gastou com energia elétrica para produzir um par de sapatos.

Segundo Schier (2006, p.26):

> Custos diretos são os custos que podem ser identificados e quantificados no produto ou no serviço e valorizados com relativa facilidade. Os materiais diretos, por exemplo, são normalmente requisitados com a identificação prévia de sua utilização, ou seja, ao emitir a requisição para o almoxarifado, o responsável pela produção já que não podem ser identificados de forma fácil, não podem ser apropriados de forma direta para as unidades específicas como, por exemplo, mão-de-obra indireta e matérias indiretas.

Dessa forma, pegamos a nossa empresa modelo, Sensify, e, como temos apenas um produto, todo custo foi considerado como direto/indireto, como as contas de energia elétrica, internet, os salários, aluguéis e peças utilizadas no produto.

3.4.2 CUSTOS DO PRODUTO

Nesta etapa do projeto, classificamos os custos do nosso trabalho, sendo eles: o arduino, os cabos USB AB, o leitor biométrico, os leds, a protoboard, as barras de pinos, os jumpers, a minicâmera e a mão de obra como custos diretos do produto, e, a energia elétrica, a internet, os anúncios patrocinados e os fretes como os custos indiretos.

Segue a planilha com os valores totais de nosso produto:

SENSIFY - GERAL	
COMPONENTES PARA FABRICAÇÃO DO PRODUTO	351.35
MÃO DE OBRA DIRETA PARA PRODUÇÃO DO PRODUTO (6)	1,000.00
ENERGIA ELÉTRICA	300.00
INTERNET	100.00
IMPORTAÇÕES (FRETES)	150.00
TOTAL	2,401.35

A partir dessa planilha, fizemos as contas necessárias para o novo custo do produto, chegando a conclusão de que seria possível a montagem de pelo menos dois deles, com os mesmos requisitos (mão de obra e componentes), abaixando o assim valor de venda e, dessa forma, tornando-o viável quando comparado ao preço de mercado.

Além disso, também fizemos uma pesquisa de mercado sobre nossos concorrentes, e um deles se chama Hikvision, uma empresa focada em tecnologia de segurança, disponibilizando desde câmeras simples até os dispositivos mais tecnológicos com reconhecimento facial e biométrico, os quais têm seus preços variados de R$170,00 até R$2800,00. (Imagem apresentada no tópico “Anexos” como “Anexo E”).

Ademais, para se ter uma ideia, há empresas, como a Ideal Technology, que vendem produtos parecidos pelo valor de R$5000,00. (Imagem apresentada no tópico “Anexos” como “Anexo F”).

Portanto, considerando que, desde o começo, focamos nosso projeto em criar um produto de baixo custo e funcional, temos certeza de que ele seria viável, tendo em vista nossa situação atual, a qual tem seus valores e conhecimentos, por enquanto, limitados.

3.4.3 PRECIFICAÇÃO

Agora, como última parte da Gestão Financeira do nosso projeto, decidimos fazer a precificação do nosso produto, utilizando o Markup. Portanto, para entendermos o que foi feito a seguir, será necessário a explicação de alguns de seus conceitos:

O Markup é um índice utilizado na formação do preço de venda de um produto ou serviço, que, mesmo não sendo tomado isoladamente como referência ao precificar, ele já pode ser considerado um ponto de partida fundamental. Calculando-o, dele, conseguimos tirar três outros conceitos chaves para o resultado final: a Margem de Contribuição, a Margem e o Índice de Markup.

A Margem de Contribuição, também conhecida como ganho bruto, representa quanto o lucro da venda de cada produto contribuirá para a empresa cobrir todos os seus custos e despesas e ainda gerar lucro. Com base nisso, você pode calcular a quantidade mínima de produtos que precisará vender para não sair no prejuízo.

O Índice de Markup é uma das ferramentas usadas para determinar preços que sejam competitivos e que passem ao cliente final a exata noção do valor entregue. Então, ao se

calcular o índice de markup, se a sua empresa chegar a preços muito mais altos que os da concorrência, é bem provável que sua margem de lucro precise ser revista.

A Margem, ou Margem de Lucro, é o valor que sobra da venda de um produto ou serviço, descontadas as despesas resultantes de sua produção ou distribuição. No contexto do markup, essa grandeza deverá ser calculada na forma de percentual.

Segue as informações do nosso Markup:

Produto/Serviço	Custo de Produção	Custo Variável
Herefy	R$ 1.351,00	R$ 550,00

Custo Total	Preço	Margem de Contribuição
R$ 2.401,35	R$ 3.000,00	R$ 598,65

Margem	Mark Up	Índice de Markup
19,96%	24,93%	1,25

Dessa forma, como apresentado acima, dividimos nosso Markup em: custo de produção, custo variável, custo total e preço de venda, obtendo, assim, os seguintes valores: a margem de contribuição, que ficou em R$598,65; a margem, que ficou com uma porcentagem de 19,96%, e o Markup, que ficou com uma porcentagem de 24,93%, exibindo um índice de 1,25.

ISSN 1983-6767

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: GERENCIANDO FINANÇAS

A Formação para a Vida é um dos eixos do Projeto Pedagógico de Formação por Competências da UNIFEOB.

Esta parte do projeto está diretamente relacionada com a extensão universitária, ou seja, o objetivo é que seja aplicável e que tenha real utilidade para a sociedade, de um modo geral.

3.5.1 GERENCIANDO FINANÇAS

- **Tópico 1:** Introdução aos conceitos econômicos e financeiros básicos

  A introdução aos conceitos econômicos e financeiros básicos
  envolve a compreensão de conceitos como receita, despesa, orçamento e poupança. Por exemplo, entender que uma receita mensal de R$ 3.000 e despesas de R$ 2.500 resultam em uma economia de R$ 500 por mês.

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

  Aqui, aprendemos sobre a independência financeira e o valor da riqueza pessoal. Um exemplo prático seria calcular o patrimônio líquido, considerando ativos (como casa e investimentos) e passivos (como empréstimos).

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

  Neste tópico, discutimos como as dívidas e os juros compostos afetam as finanças pessoais. Um exemplo seria analisar o custo de um empréstimo de R$5.000 a uma taxa de juros de 10% ao ano e como esse valor cresce ao longo do tempo.

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

Aqui, aprendemos a definir metas financeiras e a envolver outras pessoas para atingi-las. Por exemplo, se o objetivo for economizar para uma viagem em família, cada membro pode contribuir com uma quantia mensal específica para alcançar a meta.

3.5.2 ESTUDANTES NA PRÁTICA

Nossa vida cotidiana é permeada por inúmeras decisões financeiras, desde escolher o que comer no almoço até decidir sobre a compra de uma casa. No entanto, a educação financeira muitas vezes é negligenciada em nossos currículos escolares, deixando as pessoas em um mar de incertezas. Sem orientação adequada, muitas vezes baseamos nossas decisões financeiras nas influências de amigos, familiares e colegas de trabalho, que nem sempre refletem práticas eficazes. É por isso que é crucial compartilhar boas práticas financeiras que sejam acessíveis a todos, independentemente de sua origem ou nível educacional.

A gestão financeira sólida não é apenas uma habilidade pessoal; é um legado que pode ser transmitido às futuras gerações. Portanto, neste material, vamos explorar seis áreas-chave de orientação financeira que podem contribuir para uma vida mais estável e para um futuro financeiramente saudável. Banner apresentado no tópico “Anexos” como: “Anexo G” (As imagens adicionais podem ser encontradas no Instagram da empresa “sensifyiot”).

ISSN 1983-6767

4. CONCLUSÃO

Este trabalho enfatizou a atuação da nossa empresa especializada no desenvolvimento de soluções avançadas de reconhecimento. Por meio do nosso produto "Herefy", almejamos aprimorar significativamente a eficiência e a segurança no processo de chamada de presença, adotando um sistema automatizado baseado em biometria.

Para o aprimoramento contínuo deste produto, utilizamos conceitos de Programação Orientada a Objetos, integrando um robusto banco de dados SQL, além disso refinamos nosso código para garantir a total segurança e confidencialidade das informações dos nossos usuários.

Enfrentamos desafios significativos, como a gestão financeira do produto, especialmente no início, devido à natureza de baixo custo e ao grupo inicial reduzido de colaboradores. No entanto, superamos tais obstáculos com resiliência e determinação.

Em resumo, reafirmamos nosso compromisso de oferecer soluções inovadoras, adaptadas às demandas do mercado e capazes de proporcionar uma experiência superior aos nossos usuários. Estamos firmemente empenhados em evoluir continuamente e em oferecer produtos que não apenas atendam, mas superem as expectativas do público, refletindo nossa busca incansável pela excelência e pela satisfação dos nossos clientes.

ISSN 1983-6767

REFERÊNCIAS

BOOCH, G.; RUMBAUGH, J.JACOBSON, I. UML: guia do usuário. 2. ed. Rio de Janeiro: Elsevier; Campus, 2006. 474 p.

KULPAS, S. Identificação biométrica. **GV-executivo**, Rio de Janeiro, v. 5, n. 5, p. 41-42, 2006. Disponível em: https://bibliotecadigital.fgv.br/ojs/index.php/gvexecutivo/article/download/34192/32983/0. Acesso em: 23 maio. 2023.

OLIVEIRA, Antônio Benedito Silva. **Métodos e Técnicas de Pesquisa em Contabilidade**. São Paulo: Saraiva, 2003.

POLIMORFISMO EM PYTHON. Python Progressivo, Disponível em: https://www.pythonprogressivo.net/2018/11/Polimorfismo-O-que-Como-Usar-Como-fazer.html. Acesso em: 27 out. 2023.

SCHIER, Carlos Ubiratan da Costa. **Gestão de custos**. Curitiba: Ibpex, 2006.p.25.

SCHIER, Carlos Ubiratan da Costa. **Gestão de custos**. Curitiba: Ibpex, 2006.p.26.

SEBESTA, R. W. Conceitos de linguagem de programação. 11. ed. Porto Alegre: Bookman, 2018. 758 p.

ZAKAS, Nicholas C. **JavaScript de Alto Desempenho**. São Paulo: Novatec Editora, 2010. 7 p.

ANEXOS

Anexo A - Inserts das tabelas “aluno”;”sala”;”materia”;”especializacao” e “conta”

```
insert into aluno (nome, ra, email) values ("Jorge Luis", "000001", "jorge.luis@sou.unifeob.edu.br");
insert into aluno (nome, ra, email) values ("Wesley", "000002", "wesley.luis@sou.unifeob.edu.br");
insert into aluno (nome, ra, email) values ("Gabi", "000003", "gabi.luis@sou.unifeob.edu.br");
insert into aluno (nome, ra, email) values ("Kevin", "000004", "kevin.luis@sou.unifeob.edu.br");
insert into aluno (nome, ra, email) values ("Leonardo", "000005", "leo.luis@sou.unifeob.edu.br");

SELECT * FROM aluno;

INSERT INTO sala (nome, lotação) values ("Lab. informática 1", "50");
INSERT INTO sala (nome, lotação) values ("Lab. informática 2", "50");
INSERT INTO sala (nome, lotação) values ("Lab. informática 3", "50");
INSERT INTO sala (nome, lotação) values ("Lab. informática 4", "50");

select * from sala;

INSERT INTO materia (nome) values ("Modelagem de dados");
INSERT INTO materia (nome) values ("Logica de Programação");
INSERT INTO materia (nome) values ("Gestão Financeira");
INSERT INTO materia (nome) values ("Programação Orientada a Objeto");

select idmateria, nome from materia;

INSERT INTO especializacao (nome) values ("Modelagem de dados");
INSERT INTO especializacao (nome) values ("Logica de Programação");
INSERT INTO especializacao (nome) values ("Gestão Financeira");
INSERT INTO especializacao (nome) values ("Programação Orientada a Objeto");

select idespecializacao, descricao from especializacao;

INSERT INTO conta (email, nomecompleto, senha, telefonecompleto) values ("max@sou.unifeob.pro.br", "Max Streicher Vallim", "senha123", "123456789");
INSERT INTO conta (email, nomecompleto, senha, telefonecompleto) values ("marcelo@sou.unifeob.pro.br", "Marcelo Ciacco de Almeida", "senha123", "123456789");
INSERT INTO conta (email, nomecompleto, senha, telefonecompleto) values ("renata@sou.unifeob.pro.br", "Renata E. de Alencar Marcondes", "senha123", "123456789");
INSERT INTO conta (email, nomecompleto, senha, telefonecompleto) values ("nivaldo@sou.unifeob.pro.br", "Nivaldo de Andrade", "senha123", "123456789");
```

Anexo B - Inserts da tabela “professor”

```
INSERT INTO professor (nome,especializacao_idespecializacao) values ("Max Streicher Vallim","1");
INSERT INTO professor (nome,especializacao_idespecializacao) values ("Marcelo Ciacco de Almeida","2");
INSERT INTO professor (nome,especializacao_idespecializacao) values ("Renata E. de Alencar Marcondes","3");
INSERT INTO professor (nome,especializacao_idespecializacao) values ("Nivaldo de Andrade","4");
```

Anexo C - Inserts da tabela “aula”

```
INSERT INTO aula (dataaula, horainicio, horafim, materia_idmateria, sala_idsala, professor_idprofessor) values ("2023-10-01", "19:40", "22:20", 1, 1, 1);
INSERT INTO aula (dataaula, horainicio, horafim, materia_idmateria, sala_idsala, professor_idprofessor) values ("2023-10-02", "19:40", "22:20", 1, 2, 1);
INSERT INTO aula (dataaula, horainicio, horafim, materia_idmateria, sala_idsala, professor_idprofessor) values ("2023-10-03", "19:40", "22:20", 1, 3, 1);
INSERT INTO aula (dataaula, horainicio, horafim, materia_idmateria, sala_idsala, professor_idprofessor) values ("2023-10-04", "19:40", "22:20", 1, 4, 1);
INSERT INTO aula (dataaula, horainicio, horafim, materia_idmateria, sala_idsala, professor_idprofessor) values ("2023-10-05", "19:40", "22:20", 1, 2, 1);
```

Anexo D - Inserts da tabela “aula_has_aluno”

```
-- insert aulas em que o aluno de id 1(Jorge Luis) terá aula na semana
insert into aula_has_aluno (aula_idaula, aluno_idaluno, presença) values (1,1, NULL);
insert into aula_has_aluno (aula_idaula, aluno_idaluno, presença) values (7,1, NULL);
insert into aula_has_aluno (aula_idaula, aluno_idaluno, presença) values (13,1, NULL);
insert into aula_has_aluno (aula_idaula, aluno_idaluno, presença) values (19,1, NULL);
insert into aula_has_aluno (aula_idaula, aluno_idaluno, presença) values (5,1, NULL);
```

Anexo E - Produto da empresa “Hikvision”

ISSN 1983-6767

Controladora de Acesso Com Reconhecimento Facial, Biometria e RFID Hikvision DS-K1T671MF-L

Outros produtos: **Hikvision** Modelo: **DS-K1T671MF-L** Garantia: **12 Meses**

De: ~~R$ 2.799,90~~

R$ 2.564,91

à vista no PIX ou boleto **(5% de desconto)**

ou **2.699,90** à prazo

8x de **R$ 337,49** sem juros

Disponibilidade: **Imediata, Em Estoque!**

\- 1 + COMPRAR

Consulte o prazo de entrega do seu pedido

XXXXX-XXX CALCULAR

Anexo F - Produto da empresa “Ideal Technology”

ISSN 1983-6767

Anexo G - Produto da empresa “Ideal Technology”