UNIFEOB
ESCOLA DE NEGÓCIOS

2024

PROJETO INTEGRADO

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

DESENVOLVIMENTO DE SOLUÇÕES CONSOLE INTEGRADAS PARA EDUCAÇÃO, SUSTENTABILIDADE, INCLUSÃO SOCIAL E EMPREENDEDORISMO

UNIFEOB

SÃO JOÃO DA BOA VISTA, SP

NOVEMBRO 2024

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

DESENVOLVIMENTO DE SOLUÇÕES CONSOLE INTEGRADAS PARA EDUCAÇÃO, SUSTENTABILIDADE, INCLUSÃO SOCIAL E EMPREENDEDORISMO

UNIFEOB

MÓDULO MODELAGEM E DESENVOLVIMENTO DE SISTEMAS

Business Intelligence – Profª. Mariângela Martimbianco Santos

Programação Orientada a Objeto – Prof. Nivaldo de Andrade

Lógica de Programação – Prof. Marcelo Ciacco Almeida

Modelagem de Dados – Prof. Max Streicher Vallim

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

Estudantes:

Beatriz Rizzo, RA 24001283

Hanry de Souza, RA 24001865

João Gabriel Roberto, RA 24001286

Lívia Menato, RA 24000015

Luis Miguel, RA 24000174

Maria Luiza Tavares Procopio, RA 24001256

Pedro Henrique Kawahara Sales, RA 24001492

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2024

SUMÁRIO

ISSN 1983-6767

1. INTRODUÇÃO

A instituição UNIFEOB (Centro Universitário Fundação de Ensino Octávio Bastos), situada na cidade de São João da Boa Vista, propôs aos alunos do módulo de "Modelagem e Desenvolvimento De Sistemas" dos cursos de Análise e Desenvolvimento de Sistemas e Ciências da Computação um projeto integrado, o qual tem como tema: "Desenvolvimento de Soluções Console Integradas para Educação, Sustentabilidade, Inclusão Social e Empreendedorismo" , a proposta do projeto tem como objetivo desenvolver uma aplicação inovadora que visa otimizar o controle de presença dos alunos em sala de aula por meio da leitura de QR Code.

A proposta é criar um aplicativo que irá obter informações dos alunos e que vai permiti-los registrar sua presença de forma ágil e prática. Durante as aulas, os professores irão gerar um código exclusivo que, ao serem inseridos pelos alunos em seus dispositivos, permitirão a criação de um QR Code que será lido por dispositivos estrategicamente instalados na sala de aula, facilitando não apenas a contabilização de presença, mas também assegurando a integridade do registro. O sistema contará ainda com a funcionalidade de geolocalização, que irá restringir o acesso ao aplicativo, garantindo que somente alunos presentes no campus possam registrar sua presença.

O projeto se alinha aos seguintes eixos temáticos:

Educação e Capacitação: Visa aprimorar competências e conhecimentos, cedendo uma abordagem que torna o registro de presenças mais eficiente, isso libera os professores para se concentrarem no ensino, em vez de se ocuparem com questões relacionadas a frequências.

Meio Ambiente e Sustentabilidade: Ao integrar tecnologias digitais, o projeto diminui a dependência de papel, estimulando práticas mais ecológicas e sustentáveis nas instituições de ensino.

Inclusão Social : o sistema foi feito para ser acessível a todos os alunos garantindo que as informações sobre a presença sejam claras e facilmente consultadas.

Tecnologia e Inovação: A aplicação da proposta representa um avanço significativo no cenário educacional, ao usar tecnologias de ponta para abordar e resolver problemas comuns enfrentados em sala de aula.

Empreendedorismo e Economia Solidária: o Projeto pretende melhorar uma cultura de uso inovador e responsável da tecnologia, contribuindo para o desenvolvimento de uma empreendedora nos alunos, que será valiosa em suas futuras carreiras e iniciativas

Com este projeto pretendemos proporcionar aos alunos uma vivência prática e interdisciplinar, orientando-os a empregar tecnologia de forma eficaz e criativa na resolução de desafios reais, promovendo assim um desenvolvimento sustentável e inclusivo

2. DESCRIÇÃO DA EMPRESA

A empresa a qual este projeto foi destinado foi a Universidade UNIFEOB (Centro Universitário Fundação de Ensino Octávio Bastos), cujo CNPJ é 59.764.555/0001-52. Localizada em São João da Boa Vista, SP, tem uma infraestrutura composta pelos Campus Mantiqueira e Fazenda-Escola, situados na rua Av. Dr. Otávio da Silva Bastos, 2439 - Jardim Nova São João, Campus Centro, na Rua General Osório, 433 - Centro, Vila Jurídica, na R. Riachuelo, 571 - Centro, e Palmeiras, na Av. Dona Gertrudes, 221 - Centro. Conta com mais de 4000 alunos e está há quase 60 anos formando alunos em áreas como Arquitetura e Urbanismo, Ciências da Saúde, Direito, Educação, Engenharia, Negócios e Tecnologia da Informação, abrangendo assim diversas áreas do conhecimento de forma a apresentar uma contribuição na evolução da sociedade.

Fundada em 1965, sendo a primeira universidade da cidade de São João da Boa Vista, no estado de São Paulo tinha como objetivo claro empenhado pelo fundador Octávio Bastos de "levar para a região formação de qualidade nas mais diversas áreas de conhecimento, contribuindo, assim, para o seu desenvolvimento econômico e social" Institucional - UNIFEOB, UNIFEOB, disponível em: <https://unifeob.edu.br/institucional/>. acesso em: 16 maio 2024.

Após quase 60 anos de existência, é uma instituição hoje que tem como principais valores a “aprendizagem e aprimoramento contínuos, inovação, engajamento do time, transparência e sucesso do cliente” MANUAL DO ESTUDANTE, [s.l.: s.n.], 2023. É uma instituição que tem a qualidade reconhecida pelo Ministério da Educação (MEC), pelas avaliações das Instituições de Ensino Superior (IES) e pelo IGC (Índice Geral de Cursos).

ISSN 1983-6767

3. PROJETO INTEGRADO

Neste projeto, estamos dando continuidade ao trabalho anterior que inicialmente explorou a IoT, evoluindo agora para o Desenvolvimento de Soluções Console Integradas. O objetivo é criar uma solução que possa ser implementada no futuro proporcionando um impacto direto nas áreas fundamentais para a sociedade.

Estaremos, portanto, apresentando detalhadamente todo o conteúdo do nosso projeto para cada unidade de estudo trabalhada neste módulo de “Modelagem e Desenvolvimento de Sistemas”, sendo elas a Programação Orientada a Objeto (POO), Lógica de Programação, Modelagem de Dados e Business Intelligence (BI). Apresentaremos também ao final a nossa proposta para o conteúdo de Formação para a Vida.

3.1 PROGRAMAÇÃO ORIENTADA A OBJETO

A programação orientada a objeto (POO) surge como um paradigma da programação que busca servir como uma alternativa à programação estruturada, utilizando-se da "lógica de programação para a solução de um problema do mundo real em um computador", é uma programação que se baseia na aplicação da abstração, tem a ideia de manusear as coisas do mundo real, por isso a utilização do nome de "objeto" de forma a representar qualquer coisa tangível.

Um objeto, para Sebesta (2003, p.412) é o encapsulamento de uma representação de dados de um tipo específico (abstração de dados) com os subprogramas (abstração de procedimentos) que fornecem as operações para esse tipo. Na POO necessita-se especificar uma classe (sendo este um tipo de dado abstrato) previamente a associá-lo a um objeto. A essa ação chamamos de instância. De forma simples, um objeto é criado semelhante à criação de uma variável estática, e dizemos que o objeto é uma instância da e classe definida.

Através do entendimento e da aplicação correta da POO, a estrutura de um código será percebida muito mais organizada, principalmente se os conceitos de classe, objetos, herança e polimorfismo estiverem bem definidos. Além disso, a POO serve como principal facilitadora para a integração, no contexto deste projeto, das áreas de Modelagem de Dados e a Business

Intelligence, por suas funcionalidades de encapsulamento e manipulação de dados de forma segura e eficiente.

Para uma melhor compreensão da Orientação a Objetos, é necessário compreender seus quatro pilares básicos, sendo eles: Classe, Objeto, Atributo e Método. A seguir, eles serão apresentados mais detalhadamente.

3.1.1 CLASSES E OBJETOS

Segundo o Dicionário Online de Português, classe é "cada um dos grupos ou divisões de uma série ou conjunto; categoria, ordem, seção" (DICIO, 2024). Levando essa ideia para a área da lógica de programação, "classe" é uma categoria descritiva geral, que abrange o conjunto de objetos que compartilham uma ou mais características quanto a seus itens de dados e procedimentos associados. “Classe é um modelo usado para formatar a estrutura de um objeto, ou seja, uma estrutura usada para criar (instanciar) um objeto. Uma classe é, em essência, um conjunto de campos e de funcionalidades (procedimentos e funções) associadas aos campos componentes da classe com o objetivo de controlar sua funcionalidade” (MANZANO; OLIVEIRA, 2019, p. 292). Em suma, classe é um conjunto de procedimentos e de funções, que são associadas aos campos que compõe a classe e têm como objetivo controlar sua funcionalidade.

Através de uma classe conseguimos especificar um conjunto de objetos. O conceito de classe estabelece o conjunto de objetos, seus atributos e os métodos em comum de um determinado objeto. “O conjunto de membros (atributos) e procedimentos e/ou funções membro (métodos, ou seja, suas funcionalidades) agregados à classe e que serão instanciados a certo objeto chama-se **encapsulamento**, e esses atributos e/ou métodos podem ser públicos, privados ou protegidos” (MANZANO; OLIVEIRA, 2019, p.293).

Uma classe pode ser derivada de outra classe existente, chama-se nesse caso de classe filha ou subclasse, enquanto a classe já existente denominamos de pai, ou superclasse. A esse comportamento dá-se o nome de herança. A classe filha pode ter novos atributos, modificar atributos herdados da classe pai (havendo, portanto, uma especificação).

É importante também comentarmos sobre a existência das classes abstratas e classes concretas. A uma classe concreta conseguimos instanciar objetos a partir dela, já as classes abstratas não permitem esse comportamento. Não conseguimos instanciar um objeto a partir dela, exceto quando há herança (conceito que detalharemos mais adiante).

Trazendo o conceito de objeto, segundo o Dicio (2024), objeto é "coisa material que pode ser percebida pelos sentidos (visão, audição, tato, olfato e paladar)". Trazendo essa ideia

para a nossa realidade na área da programação, em específico na POO, "objeto é "qualquer módulo que contém rotinas e estruturas de dados capaz de interagir com outros módulos similares, trocando mensagens". Desta explicação deve-se entender por rotinas o mesmo que módulos de procedimentos e/ou funções, e por troca de mensagens, algo semelhante ao uso de passagem de parâmetros por referência. Um objeto só poderá existir se este for definido a partir de uma classe, ou seja, se for instanciado a partir de uma classe" (MANZANO; OLIVEIRA, 2019, p.294).

3.1.2 ATRIBUTOS, MÉTODOS, ENCAPSULAMENTO, HERANÇA E POLIMORFISMO.

Seguindo adiante nos conceitos de Programação Orientada a Objetos, veremos agora o que é um atributo. Segundo o dicionário atributo é "o que é próprio, característico de algo ou de alguém; particularidade: a palavra é um atributo do homem" (DICIO, 2024). Na POO, um atributo é uma propriedade que define as características de um objeto ou uma entidade. Pode ser comparado a uma variável, tendo como função o armazenamento de informações específicas do objeto.

Um atributo pode ser público ou privado, ou seja, pode ser acessível por qualquer parte do código ou acessível somente dentro da própria classe ou por classes derivadas. Isso pode ser feito através do encapsulamento, técnica esta que oculta os detalhes internos de uma classe, permitindo o acesso controlado por métodos.

A métodos entendemos como funções associadas a uma classe e que define como um atributo será manipulado. É o método que indica como uma ação específica do objeto será executada, o que reflete em seu comportamento. Um método pode modificar os atributos de um objeto, o que garante o controle sobre suas operações.

Além disso, métodos precisam ser coerentes com a estrutura da classe e suas funcionalidades, representando o comportamento do objeto.

O conceito a seguir, já comentado anteriormente, é o encapsulamento. Tendo como principal função o controle do como os atributos e métodos de uma classe serão acessados e modificados, o encapsulamento impede o acesso direto aos atributos de um objeto quando estes são privados ou protegidos, permitindo, portanto, que estes dados sejam acessados ou alterados por meio de métodos específicos (como getters e setters). Isso é essencial para uma maior segurança e controle do comportamento interno de uma classe.

Iremos tratar agora de um conceito bastante importante, sendo ele a herança. Essa técnica permite que uma classe filha (subclasse) herda atributos e métodos de uma classe pai

(superclasse). É importante pois facilita a reutilização de código, simplificando a estruturação do sistema e promovendo a manutenção, já que uma subclasse pode reutilizar ou mesmo estender comportamentos da classe pai. Contudo, é importante ressaltar que atributos e métodos privados de uma classe pai não são herdados.

Por fim, falaremos sobre polimorfismo, ou seja, a capacidade de um objeto assumir diferentes formas de comportamento. Em POO, isso significa que métodos de classes diferentes podem compartilhar o mesmo nome, mas com implementações distintas, dependendo do contexto. Isso garante a interação com um objeto mesmo sem saber exatamente qual é seu tipo específico, o que garante flexibilidade no código.

3.1.3 MÉTODOS ESTÁTICOS, PÚBLICOS E PRIVADOS

Métodos são funções associadas a uma classe ou um objeto e representam o comportamento desse objeto. Quando pensamos em métodos estáticos, públicos ou privados, estamos pensando nas formas que temos de definir e acessar essas funções dentro de uma classe.

Aos métodos públicos entendemos como aqueles que podem ser acessados fora da classe, ou seja, em qualquer outro código que interage com uma instância de classe. Quando um método é declarado como público, ele se torna acessível por outras classes, o que permite que o comportamento da classe seja utilizado de forma ampla.

Já os métodos privados, como seu nome diz, são projetados para serem acessíveis apenas dentro da própria classe, diferentemente dos métodos públicos. São usados para encapsular a lógica interna, de forma a manter partes da implementação retidas do mundo exterior, o que promove a ocultação de informações.

Por fim, sobre os métodos estáticos, estes pertencem à classe em si e podem ser chamados sem que seja necessário criar um objeto daquela classe. Não recebem o "self" como parâmetro, ou seja, não têm acesso ao estado da instância e nem podem acessar atributos e métodos de instância. São utilizados, de forma geral, para funcionalidades relacionadas à classe e que não precisam de uma instância para funcionar.

Métodos privados ou protegidos são usados como métodos auxiliares para ações de controle da própria classe, não sendo acessíveis aos objetos instanciados, ou seja, um método que não possua acesso fora de uma classe não necessita ser visível e acessível fora da classe.

Nesse projeto o objetivo foi elaborar um sistema que permite o controle de presença dos alunos em uma universidade. O sistema conta com funcionalidades de login, exibição da

grade horária tanto dos alunos quanto dos professores, apresenta um relatório que permite controle de faltas e presenças, e por fim permite gerar o QR Code que servirá como forma de registro de presença.

O sistema é integrado a um banco de dados que, nesse momento do projeto, apenas busca as informações dos usuários, grades de horários e históricos de presenças.

Durante a codificação do código, este foi embasado nos princípios da Programação Orientada a Objetos (POO), podendo ser destacado que as funções mais utilizadas foram o encapsulamento, a herança e o polimorfismo.

Para uma melhor aplicação dos conceitos, foram criados vários arquivos Python onde cada um representa componentes específicos do sistema. Para clarificar, temos uma classe Usuario, esta serviu como uma classe base (ou superclasse) para Aluno e Professor.
Em usuários temos atributos como o tipo, que serve para conseguirmos definir o tipo de usuário, o id, que é o identificador que cada usuário utiliza (o RA no caso de aluno, e matrícula no caso de professor), e por fim a senha, que é algo que será necessário para ambos usuários mais adiante.

Temos também nessa classe o método `login()` que permite aos usuários a autenticação no sistema, o método `verificar_credenciais()` que busca as informações no banco de dados a fim de verificar as credenciais para o login.

Seguindo adiante, temos dois arquivos sendo um para Aluno e um para Professor. Essa forma de divisão no código serve para que fique mais estruturado tudo que precisamos para um determinado objeto. Então, apesar de serem dois arquivos bem parecidos, ambos herdam tudo do usuário, modificando apenas que um passa o tipo “aluno” e portanto pega as informações do aluno, e o outro passa o tipo “professor” utilizando nos métodos específicos para o mesmo. No entanto, a classe Aluno possui uma diferença que é o atributo “localização”, que será necessário para a validação geográfica.

Dando continuidade, para a validação geográfica foi utilizada uma biblioteca chamada *Geopy*. Esta biblioteca permite, dentre várias funções, o cálculo de distância utilizando da latitude e longitude de dois locais, além de ser possível definir um raio do local permitido (em km ou em milhas).

Tanto nessa parte do código, quanto no arquivo onde foi elaborado todo o menu, para cada tipo de usuário, e definido as funções para cada opção possível do menu, foram utilizados a herança, que permite comportamentos específicos das classes pai.

Enfim, temos o arquivo do banco de dados, este foi responsável pela conexão ao banco e o gerenciamento da comunicação entre o sistema e o banco.

De forma resumida, a classe Aluno e Professor herdam de Usuário, permitindo o reaproveitamento do código e uma arquitetura mais modular. Ambas classes têm funcionalidades específicas, mas compartilham também de comportamentos em comum, como o login.

Foi utilizado também o encapsulamento, como por exemplo LocalizaçãoAluno e LocalizaçãoCampus, que encapsulam as coordenadas geográficas específicas, e também no banco de dados, onde foi encapsulado os dados para a conexão em ConexaoBanco.

A codificação elaborada no VsCode, em linguagem Python, pode ser vista detalhadamente no anexo 1, ao final deste projeto.

3.2 LÓGICA DE PROGRAMAÇÃO

> “A lógica de programação é o conjunto de regras e técnicas que os programadores utilizam para projetar e desenvolver programas de computador. É a habilidade de pensar de forma lógica e estruturada, decompondo um problema complexo em etapas mais simples [...]" (Rocketseat, s.d.).

Ao utilizar ferramentas de programação em console, garantimos que o sistema seja acessível, simples de usar e de fácil manutenção. A lógica aplicada neste projeto abrange algoritmos para gerenciamento de dados, controle de fluxo por meio de estruturas condicionais e funções que facilitam a estruturação e reutilização do código. Tais soluções permitem que a aplicação se expanda conforme a necessidade do usuário.

3.2.1 CONCEITOS FUNDAMENTAIS DO DESENVOLVIMENTO DE SOFTWARE

Neste projeto estão sendo incorporados conceitos que permitirão uma transição de IoT para um sistema baseado em console, proporcionando uma maior acessibilidade e inclusão.

- Algoritmos: Os algoritmos possuem um papel importante na automação de processos, como a validação de dados;
- Variáveis: No novo contexto, variáveis continuam a armazenar e processar informações cruciais, como dados de usuários e registros da faculdade. A simplicidade no uso de variáveis será mantida, permitindo que o sistema gerencie grandes volumes de dados de maneira eficaz;

- Tipos de Dados: Com tipos de dados adequados, como inteiros para cálculos de impacto ou strings para armazenar informações de usuários, o sistema se mantém eficaz. Dessa forma, o sistema irá realizar operações complexas com simplicidade;
- Modularização: A estruturação do código por meio de funções será reforçada nesta nova fase do projeto;
- Estruturas Condicionais: É necessário o uso de estruturas condicionais para a tomada de decisões dentro do sistema, como a verificação de requisitos para inclusão social ou de metas educacionais. A lógica condicional, que já foi utilizada no projeto anterior, será adaptada para suportar novas funcionalidades e requisitos;
- Operadores Lógicos e de Comparação: Irão garantir que as validações necessárias ao sistema sejam realizadas de forma correta, como comparações entre dados educacionais e métricas. Esses operadores permitem que o sistema execute várias verificações simultaneamente.

3.2.2 DESENVOLVIMENTO DE APLICAÇÕES

Definimos uma série de regras de negócio para a aplicação, visando atender às necessidades de educação, sustentabilidade, inclusão social e empreendedorismo. Essas regras foram traduzidas em algoritmos e funções modulares, garantindo uma solução eficiente e de fácil manutenção.

Regras de Negócio:

- Registro de Presença: A presença dos alunos é registrada via QR code, que só pode ser validado se o aluno estiver fisicamente presente no campus, com validação da geolocalização.
- Relatórios Educacionais: O sistema gera relatórios detalhados sobre o desempenho acadêmico dos alunos, contabilizando faltas e presenças.
- Impacto Sustentável: A aplicação calcula e gera relatórios sobre o impacto sustentável das ações realizadas, como a redução do uso de papel.
- Inclusão Social: Alunos de programas de inclusão social têm acesso a funcionalidades personalizadas, como acompanhamento de desempenho e suporte adicional.

Lógica dos Algoritmos: A lógica da aplicação foi implementada em algoritmos que garantem a execução correta das regras de negócio. A função validar_presenca() verifica a

geolocalização e o QR code, assegurando que o aluno esteja dentro do campus antes de registrar sua presença. Funções como gerar_qr_code() automatizam o processo de geração dos códigos de presença.

Funcionalidades Principais:

- Geração e Validação de QR Code: O sistema cria códigos QR únicos para cada aula, validando a presença dos alunos.
- Cálculo de Impacto Sustentável: O sistema calcula e gera relatórios sobre a economia de energia e recursos com base nas ações dos usuários.
- Inclusão Social: Funcionalidades personalizadas para alunos que participam de programas sociais, permitindo monitorar o desempenho acadêmico.

O sistema foi projetado com funções específicas para garantir modularidade e foi desenvolvido seguindo os princípios de código limpo, onde cada função possui uma única responsabilidade, facilitando entendimento e manutenção. A divisão em módulos independentes permite a adição de novas funcionalidades sem impactar a estrutura existente.

3.2.3 IMPLEMENTAÇÃO E VALIDAÇÃO

Após a implementação do módulo de autenticação, foram realizados testes para validar sua funcionalidade. O código foi executado, e as seguintes saídas foram obtidas:

- Login bem-sucedido: Esse resultado indica que a autenticação foi realizada com sucesso utilizando as credenciais corretas (usuário: ”usuario_teste” e senha: “senha_teste”).
- Usuário ou senha incorretos: Esse resultado é retornado ao tentar autenticar com um usuário ou senha inválida, demonstrando que o sistema está verificando corretamente as credenciais e impedindo acessos indevidos.

3.3 MODELAGEM DE DADOS

A modelagem de dados possui uma importância notável na visualização dos dados armazenados devido à sua organização. Com ela, é possível estruturar as informações adquiridas pelas instituições, facilitando a interpretação de potenciais problemas futuros. Essa situação só se torna realizável porque, em sua estruturação inicial, é necessário que os dados estejam relacionados entre si, tornando a visibilidade

mais efetiva e o mapeamento mais claro. Além disso, a modelagem de dados colabora com a regulamentação da Lei Geral de Proteção de Dados Pessoais (LGPD), Lei nº 13.709, criada em 2018, com o objetivo de proteger os dados pessoais dos usuários.

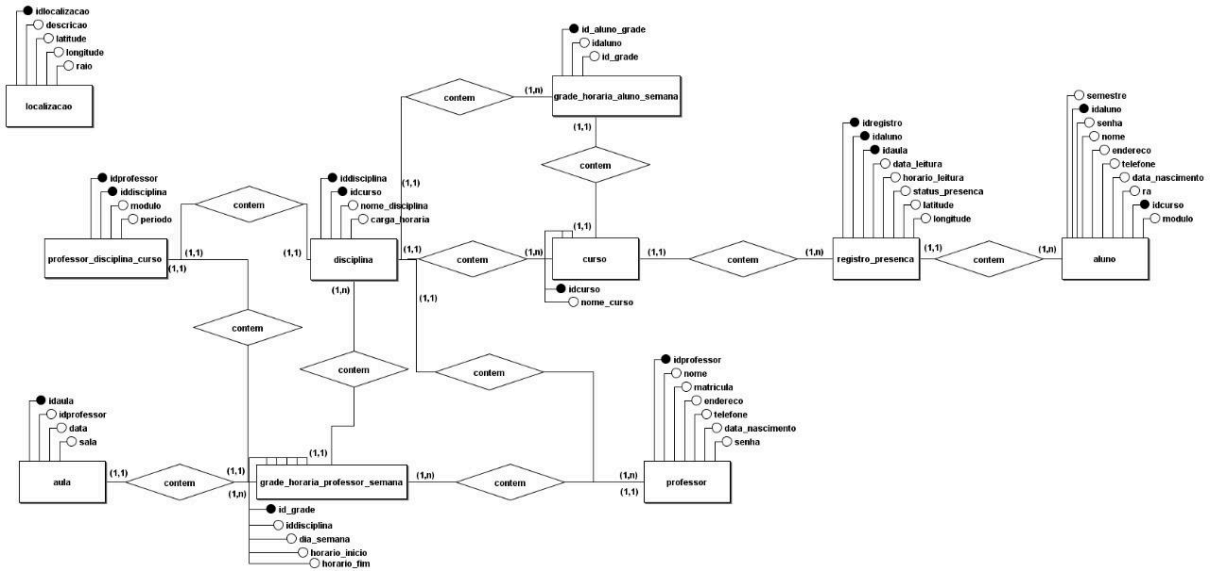
Segundo o site Alura (2024),

> "Uma vez que estivermos com a modelagem de dados adequada, conseguimos garantir a integridade, consistência e qualidade dos dados, facilitando a tomada de decisões estratégicas do seu negócio ou da empresa onde você trabalha."

Com o andamento do projeto de contabilização de chamadas por QR code, é fundamental utilizar a modelagem de dados em sua criação, considerando que serão armazenados dados importantes para a identificação dos alunos e professores. Além disso, serão mantidos os registros das chamadas dos estudantes da instituição. Para que a modelagem de dados se torne compatível com as necessidades de qualquer instituição, é necessário passar por três etapas: modelo conceitual, modelo lógico e modelo físico.

3.3.1 MODELO CONCEITUAL

Figura 1 - Modelo conceitual



Fonte: Autores

A proposta de solução visa criar um banco de dados que represente um sistema acadêmico, com a gestão de alunos, cursos, disciplinas, aulas e registros de presença. O modelo de dados deve considerar as seguintes entidades e relacionamentos:

- **Entidades Principais:**
    - Aluno: Representa os estudantes que fazem parte de um curso. Cada aluno tem um RA (registro acadêmico), nome, endereço, telefone, data de nascimento e senha.
    - Curso: Refere-se aos cursos oferecidos, como "Análise e Desenvolvimento de Sistemas" e "Ciência da Computação". Cada curso tem um identificador único (idcurso) e um nome.
    - Disciplina: Refere-se às matérias que compõem o curso. Cada disciplina tem um identificador (iddisciplina) e um nome.
    - Professor: Refere-se aos docentes que ministram as aulas. Cada professor tem um idprofessor e nome.
    - Aula: Refere-se à realização de uma aula, associada a uma grade horária e a um professor específico.
    - Registro de Presença: Controla as presenças dos alunos nas aulas, incluindo a data e horário de leitura, status da presença (presente ou ausente), e as coordenadas de latitude e longitude no momento do registro.
    - Localização: Representa os locais onde as aulas acontecem (salas e auditórios), com suas coordenadas geográficas e raio de cobertura.
- **Relacionamentos:**
    - Aluno-Disciplina (Grade Horária): A tabela aluno_grade_horaria liga os alunos às disciplinas que cursam, a partir da grade horária.
    - Professor-Disciplina-Curso (Professor Disciplina Curso): A tabela professor_disciplina_curso relaciona os professores às disciplinas que eles ministram, incluindo o curso, módulo e período.
    - Aluno-Aula-Registro de Presença: A tabela registro_presenca registra a presença do aluno nas aulas.

Modelo de Entidade Relacionamento (DER)

O Diagrama de Entidade Relacionamento (DER) foi elaborado com base nas entidades e relacionamentos descritos. O modelo está estruturado da seguinte forma:

- Aluno (1) → (M) Registro de Presença: Um aluno pode ter múltiplos registros de presença, mas cada registro pertence a um único aluno.
- Aula (1) → (M) Registro de Presença: Cada aula pode ter múltiplos registros de presença, mas cada registro pertence a uma única aula.

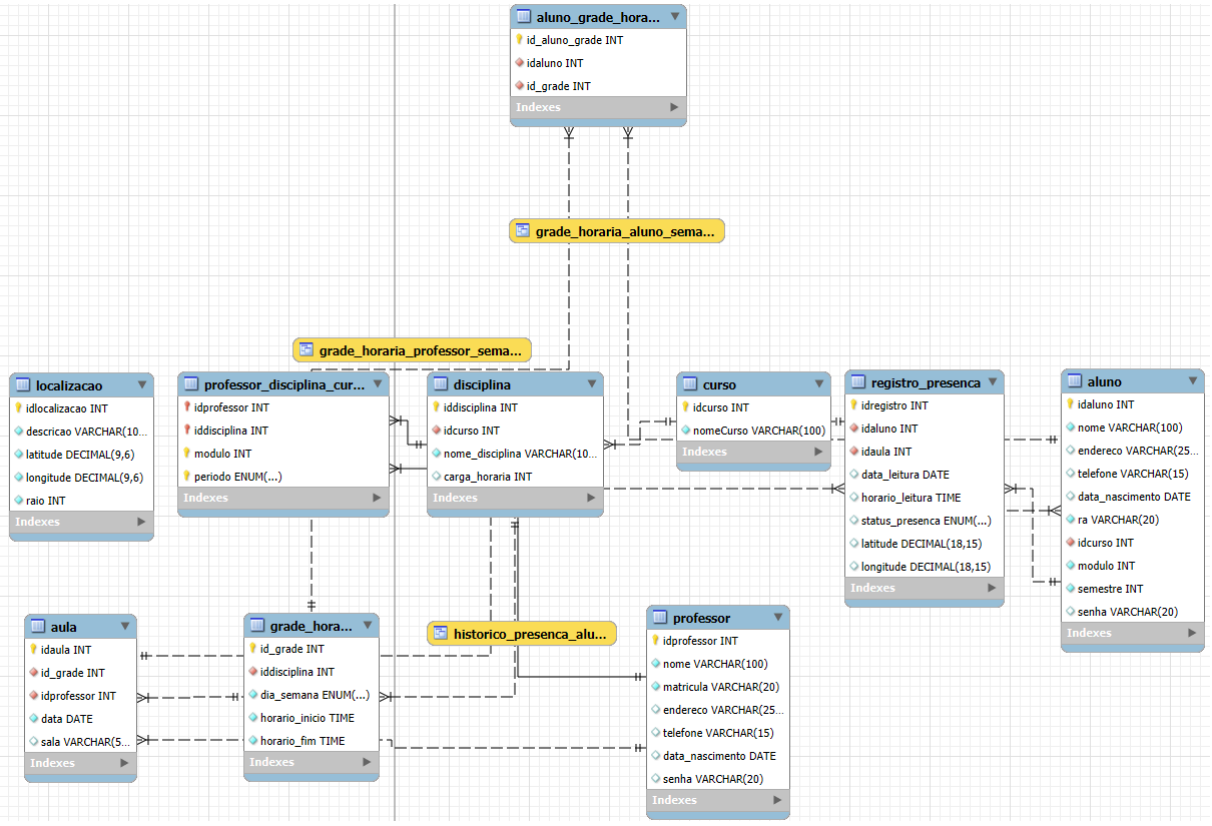
- Professor (1) → (M) Professor Disciplina Curso: Cada professor pode ministrar várias disciplinas em diferentes cursos, mas cada entrada na tabela professor_disciplina_curso se refere a um único professor.
- Disciplina (1) → (M) Professor Disciplina Curso: Uma disciplina pode ser ministrada por vários professores, mas cada associação de professor, disciplina e curso é única.
- Aluno (M) → (M) Disciplina através da Grade Horária: Cada aluno pode estar matriculado em várias disciplinas e cada disciplina pode ter múltiplos alunos.

3.3.2 MODELO LÓGICO E FÍSICO

O banco de dados foi projetado e implementado utilizando o SGBD MySQL (anexo 3), com tabelas que representam as entidades mencionadas no modelo. As instruções INSERT já foram fornecidas para popular as tabelas com dados fictícios de alunos, cursos, disciplinas, professores, aulas e registros de presença.

O modelo lógico descreve as tabelas e seus relacionamentos de forma detalhada, incluindo a definição de chaves primárias, chaves estrangeiras e tipos de dados apropriados para cada coluna.

Figura 2 - Estrutura Lógica das Tabelas



Fonte: Autores

- **Chaves Primárias**: As chaves primárias são usadas para identificar cada registro unicamente em suas respectivas tabelas.
- **Chaves Estrangeiras**: São definidas para manter os relacionamentos entre as tabelas, assegurando que os dados relacionados permaneçam consistentes.
- **Tipos de Dados**: Foram escolhidos tipos de dados apropriados para cada coluna, como INT para identificadores, VARCHAR para strings, BOOLEAN para presença e DATE para datas.

3.3.3 SQL

Comandos Básicos SQL:

- Insert: Utilizado para adicionar dados nas tabelas. Exemplo de inserção de alunos e registros de presença.
- Update: Pode ser utilizado para atualizar informações de alunos ou professores, conforme necessário.
- Delete: Utilizado para excluir registros de presença ou qualquer outra entidade.
- Select: Consultas SQL para testar o banco de dados e gerar relatórios. Exemplos de consultas já foram fornecidos, como: calcular_faltas_presencas_disciplina_nome_parcial, calcular_faltas_presencas_modulo_ra e calcular_faltas_presencas_curso_ra.

Procedures SQL para Testes

Os procedures SQL são essenciais para testar a implementação e gerar relatórios relevantes. Alguns exemplos de procedures já fornecidas nos procedimentos armazenados são:

- calcular_faltas_presencas_disciplina_nome_parcial: Realiza o cálculo de faltas e presenças dos alunos em uma disciplina específica.
- calcular_faltas_presencas_modulo_ra: Realiza o cálculo de faltas e presenças dos alunos em um módulo, baseado no registro acadêmico (RA).
- calcular_faltas_presencas_curso_ra: Realiza o cálculo de faltas e presenças dos alunos em um curso específico, com base no RA.

Esses procedimentos podem ser invocados utilizando o comando CALL no MySQL, permitindo a geração de relatórios detalhados sobre as presenças e faltas.

O comando CREATE PROCEDURE calcular_faltas_presencas_curso_ra define a store procedure chamada de calcular_faltas_presencas_curso_ra. A utilização desse procedure é útil para geral um relatório de presenças e faltas de um aluno específico em seu curso, utilizando o RA como filtro.

Ao utilizar a PROCEDURE o processamento é feito no servidor, o que otimiza o processo e evita que consultas SQL sejam feitas diretamente na aplicação.

Figura 3 - Criação da "Procedure" para registro de presenças

```
735 • CREATE PROCEDURE calcular_faltas_presencas_curso_ra(
736       IN ra_aluno VARCHAR(20)
737   )
738   BEGIN
739       SELECT
740           a.nome AS NomeAluno,
741           c.nomeCurso AS Curso,
742           COUNT(CASE WHEN rp.status_presenca = 'Presente' THEN 1 END) AS TotalPresencas,
743           COUNT(CASE WHEN rp.status_presenca = 'Ausente' THEN 1 END) AS TotalFaltas
744       FROM
745           registro_presenca rp
746       JOIN
747           aula au ON rp.idaula = au.idaula
748       JOIN
749           grade_horaria g ON au.id_grade = g.id_grade
750       JOIN
751           disciplina d ON g.iddisciplina = d.iddisciplina
752       JOIN
753           aluno a ON rp.idaluno = a.idaluno
754       JOIN
755           curso c ON a.idcurso = c.idcurso
756       WHERE
757           a.ra = ra_aluno
758       GROUP BY
759           a.nome, c.nomeCurso;
760   END //
```

Fonte: Autores

Figura 4 - Chamada da "Procedure" dentro do MySQL Workbench

```
1 • CALL calcular_faltas_presencas_curso_ra('24000001');
2
```

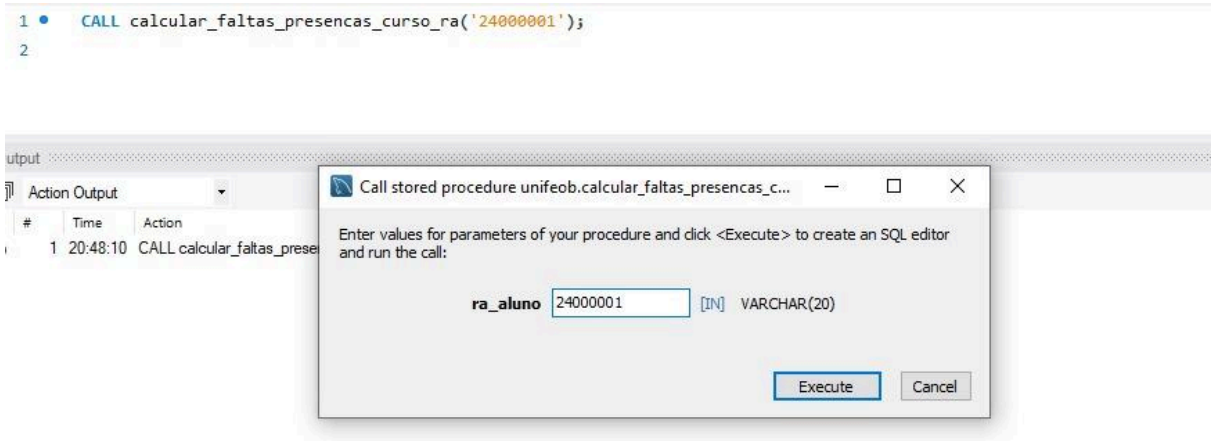
utput

Action Output

#	Time	Action	Message
1	20:48:10	CALL calcular_faltas_presencas_curso_ra('24000001')	1 row(s) returned

Fonte: Autores

Figura 5 - Resultado esperado da chamada da "Procedure"



Fonte: Autores

3.4 BUSINESS INTELLIGENCE

Business Intelligence (ou BI) refere-se ao acompanhamento e armazenamento de dados essenciais por meio de um dashboard, que é um painel visual com conteúdos cruciais, tornando mais fácil a tomada de decisões importantes. Com essa ferramenta, é possível ter uma visualização mais clara das informações coletadas de uma empresa. O BI é importante, pois cria conexões entre os dados coletados e os disponibiliza em tempo real, de forma visual, por meio de gráficos, mapas e tabelas interativas.

Segundo o site Oracle (2017), "muitas soluções inteligentes vêm com visualização de dados, que oferecem a capacidade de transformar dados automaticamente em gráficos ou outros tipos de apresentação visual."

Com isso em mente, torna-se necessário criar um sistema de Business Intelligence (BI) no projeto de contabilização de chamadas por QR code. Com esse sistema, será possível encontrar soluções racionais para os problemas gerados por um registro de presença frágil, permitindo visualizar e acompanhar a participação dos estudantes. Além disso, isso tornará perceptível o comprometimento dos acadêmicos em relação ao processo de aprendizado, permitindo identificar padrões de participação e ausência, o que colabora no processo de tomada de decisões educacionais.

3.4.1 ORGANIZAÇÃO E IDENTIFICAÇÃO DAS INFORMAÇÕES

De acordo com o site Five performance digital (2024),

> "A organização de dados é um processo fundamental para empresas e organizações que desejam gerenciar e utilizar eficientemente as infinidade de armações que possuem. Com o avanço da tecnologia e o aumento exponencial da qualidade de dados gerados diariamente, a organização adequada dessas informações se tornou essencial para a tomada de decisões estratégicas e o sucesso dos negócios".

Esse cenário enfatiza a importância de identificar dados relevantes para a iniciação mais organizada de projetos. Uma vez identificados, o próximo passo para um sistema mais preciso é organizar as informações coletadas, de modo a possibilitar uma visualização clara dos dados armazenados. Com isso, ferramentas como Business Intelligence (BI) auxiliam na visualização de dados, pois possuem interfaces nítidas e intuitivas, tornando possível a utilização de gráficos que interpretam as informações coletadas e armazenadas.

Com o projeto de contabilização de chamadas, torna-se necessário coletar informações dos estudantes e professores da instituição para uma identificação mais precisa. Isso envolve organizar os dados gerados em sala de aula em seus respectivos registros individuais, tornando a ferramenta de busca limpa e precisa, facilitando a visualização.

3.4.2 MANIPULAÇÃO E ANÁLISE DE DADOS

Após a estruturação e identificação das informações relevantes, o próximo passo é manipular e analisar os dados para formar indicadores que possam ser visualizados e usados na tomada de decisões.

O domínio de dados envolve o processo de coleta, e limpeza e padronização e integração de dados procedentes de diversas fontes, como o banco de dados do sistema e fontes externas, de modo a criar uma base de dados confiável.

A primeira ação a ser tomada é a criação de uma estrutura organizada para o banco de dados, que devem incluir:

- Dados do Aluno: nome, RA, endereço, telefone, data de nascimento, curso, semestre, módulos e senha.
- Dados da Aula: data, horário, professor, prédio, sala e status da presença.
- Dados do professor: nome, número da matrícula, endereço, telefone, data de nascimento e senha.
- Dados da Disciplina: nome da disciplina e carga horária.

- Dados da Grade horária: dia da semana, horário de início da aula e horário que termina a aula.

Estas informações, uma vez organizadas, poderão ser visualizadas de maneira clara através de gráficos e tabelas no dashboard, facilitando a compreensão da participação dos alunos.

Após a organização, a manipulação e análise dos dados o próximo passo. este processo envolve:

- Coleta de dados: Reunir informações do sistema e de fontes externas.
- Limpeza de dados: Remover duplicatas, corrigir erros e garantir a integridade dos dados.
- Padronização: Estabelecer formatos consistentes para os dados, como datas e horários.
- Integração: Unir dados de diferentes fontes para criar uma base de dados coesa e confiável.
- A Partir dessa base de dados estruturada, indicadores-chaves poderá ser gerados. como
- Taxa de Presença: Percentual de alunos presentes em cada aula.
- Frequência Média: Média de presenças por aluno ao longo do semestre.
- Identificação de Padrões: Análise de frequências para identificar alunos com alta taxa de faltas.

Segundo o site Alura (2023),“Por meio da análise de dados, as perguntas podem ser respondidas de maneira embasada, com base na realidade que os dados mostram. Sem achismos e suposições.”

Indicadores são extremamente importantes para uma visualização clara das informações, mas também contribuem para a tomada de decisões educacionais mais informadas, como uma intervenção para melhorar a participação dos alunos.

3.4.3 CRIAÇÃO DE MODELOS DE ANÁLISE DE DADOS

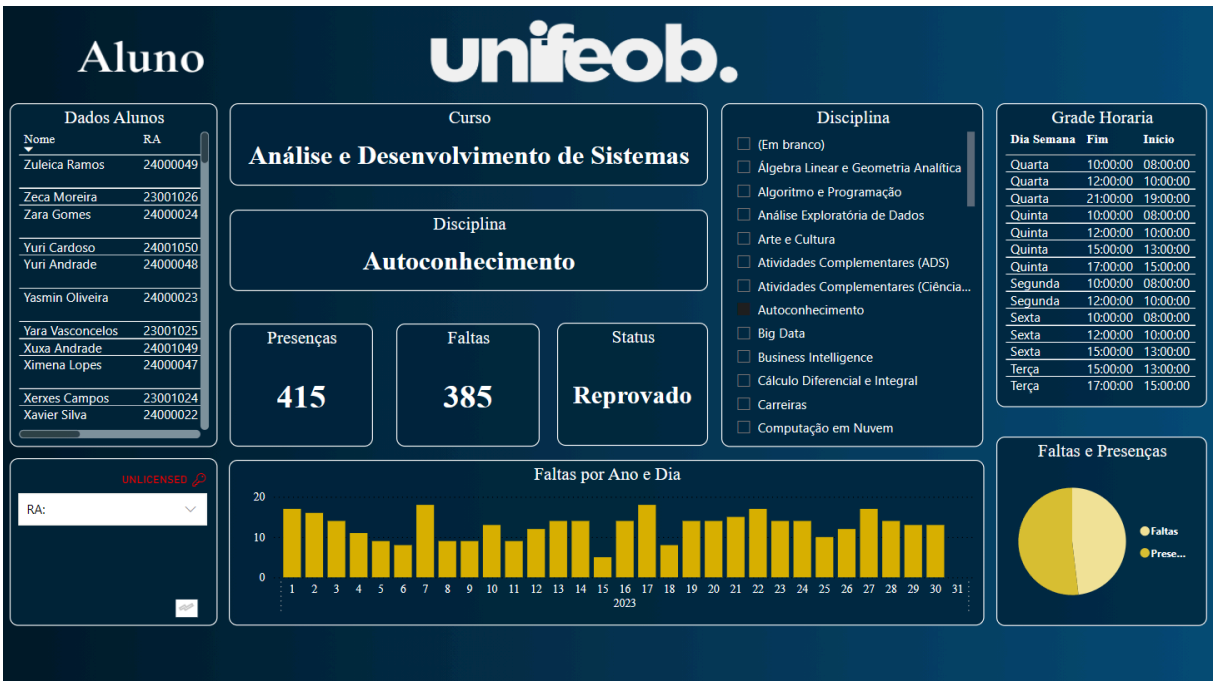
De acordo com o site Alura(2024)

> “O uso de dashboards serve como uma solução indispensável, pois ao sintetizar dados complexos em representações visuais intuitivas, eles capacitam os tomadores de decisão a compreenderem rapidamente suas operações, identificarem oportunidades e tomarem decisões fundamentadas.”

A criação de um dashboard dinâmico e interativo é uma ferramenta poderosa para a análise de dados de presença. Com ele, será possível não apenas visualizar informações relevantes, mas também tomar decisões informadas que impactem positivamente o ambiente

educacional. Com a criação do projeto de contabilização de chamadas, tornou-se necessário desenvolver um dashboard para alunos e professores, a fim de que possam acompanhar o desempenho em sala de aula. Além disso, a universidade terá acesso a essas informações, o que será útil caso o aluno enfrente dificuldades para visualizar ou encontrar as informações que procura. Cada acesso terá suas limitações específicas tornando sua utilização adequada para cada tipo de usuário. (anexo 4 e anexo 5)

Figura 6 - Dashboard Aluno



Fonte: Autores

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: GERENCIANDO FINANÇAS

É impossível não pensarmos em dinheiro quando pensamos em "finanças", e é justamente sobre o como podemos fazer a melhor gestão do nosso dinheiro que iremos tratar sob o tema "Gerenciando Finanças". A administração financeira é uma ferramenta essencial para alcançar a independência financeira e realizar sonhos pessoais e profissionais.

Compreender e aplicar conceitos financeiros e econômicos básicos, controlar despesas e gerir investimentos são passos fundamentais para construir uma vida financeira saudável e sustentável. A seguir, exploramos tópicos que abordam desde os princípios de finanças até o planejamento de longo prazo para metas e projetos pessoais.

Exemplos de Aplicação de Administração Financeira na Vida Pessoal:

- Despesas com Pessoal: Assim como as empresas têm despesas com funcionários, você pode considerar despesas relacionadas à assistência médica, odontológica e seguro de vida. Essas despesas garantem sua segurança e bem-estar, assim como as empresas investem na saúde de seus colaboradores.

- Despesas com Ocupação: As empresas pagam aluguel e IPTU por seus escritórios, e você pode aplicar o mesmo conceito em sua vida pessoal. Considere o pagamento do aluguel, IPTU e contas de casa como parte de suas despesas mensais essenciais.
- Despesas com Serviços Pessoais: Empresas frequentemente contratam serviços de manutenção, advogados e contadores. Da mesma forma, você pode destinar parte do seu orçamento para serviços pessoais que garantem a eficiência do seu dia a dia, como manutenção do carro, consultas jurídicas e serviços contábeis.
- Despesas Diversas: Assim como as empresas têm despesas variadas, você também pode incluir categorias como refeições, viagens e combustíveis. Monitorar essas despesas ajuda a ter uma visão clara de onde seu dinheiro está sendo gasto e permite ajustar o orçamento conforme necessário.

No contexto pessoal, o gestor financeiro ajuda indivíduos a organizar suas finanças, elaborar orçamentos, planejar investimentos e preparar-se para a aposentadoria. Ele orienta na escolha de produtos financeiros, como contas de poupança, investimentos em ações e até mesmo criptomoedas, garantindo que as pessoas alcancem seus objetivos financeiros de forma sustentável.

Em empresas, o gestor financeiro é responsável por decisões que impactam diretamente a saúde financeira da organização. Isso inclui a captação de recursos, análise de custos, gestão de riscos e planejamento estratégico. O objetivo é maximizar o valor da empresa e assegurar sua sustentabilidade a longo prazo.

Portanto, a atuação de um gestor financeiro não se limita às empresas; ela é igualmente importante na gestão das finanças pessoais. Compreender o dinheiro e suas funções, aliado ao conhecimento de como gerenciar esses recursos, é essencial para promover uma vida financeira saudável e uma economia robusta.

3.5.1 GERENCIANDO FINANÇAS

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

A gestão financeira é crucial tanto para empresas quanto para indivíduos, e as estratégias utilizadas em um contexto podem ser aplicadas com sucesso no outro. O gestor financeiro, em ambos os casos, têm um papel fundamental em assegurar que os recursos sejam utilizados de forma eficiente.

Um dos principais objetivos de um gestor financeiro em uma empresa é garantir a liquidez. Isso significa que a empresa deve ser capaz de cumprir suas obrigações financeiras no curto prazo. Da mesma forma, na vida pessoal, é vital manter uma reserva de liquidez, ou seja, ter um fundo de emergência que permita cobrir despesas inesperadas, como consertos de carro ou despesas médicas.

Outro aspecto importante é a otimização de gastos. Nas empresas, isso envolve a análise contínua das despesas para identificar áreas onde é possível cortar custos sem sacrificar a qualidade dos produtos ou serviços. Na vida pessoal, essa prática se traduz na revisão do orçamento familiar para eliminar gastos desnecessários e maximizar a utilização do que se tem.

Além disso, o gestor financeiro deve focar em resultados e custos. Em uma empresa, maximizar os ganhos significa aumentar o faturamento e controlar os custos, garantindo assim a saúde financeira. Para indivíduos, a fonte de geração de caixa é o trabalho, mas é importante considerar outras formas de aumentar a renda.

Exemplos de Gestão Financeira Pessoal:

- Equilíbrio entre Receitas e Despesas: Avaliar regularmente o que entra e o que sai do orçamento para garantir que não haja um descompasso significativo. Se a diferença for muito próxima, ações preventivas são necessárias.
- Evitar Dívidas Bancárias: Estabelecer limites de uso de cartões de crédito e evitar empréstimos desnecessários para não comprometer a saúde financeira.
- Renegociação com Fornecedores: Conversar com prestadores de serviços para obter melhores condições de pagamento ou tarifas mais baixas.
- Controle de Gastos: Manter um registro detalhado das despesas para identificar onde é possível economizar. Isso pode incluir a troca de hábitos de consumo, como reduzir o número de refeições fora de casa.

- Verificação do Nível de Gastos: Analisar mensalmente os gastos para assegurar que estão dentro do orçamento planejado.
- Venda de Bens ou Ativos: Considerar vender itens que não estão sendo usados, como eletrônicos ou móveis, para aumentar a liquidez e melhorar a situação financeira.

Por fim, uma estratégia eficaz de gestão financeira é reduzir custos. Se uma pessoa aumentar sua receita sem controlar as despesas, pode acabar não vendo um aumento significativo na sobra no final do mês. Portanto, assim como as empresas buscam equilibrar suas contas e maximizar lucros, os indivíduos também devem ter em mente que o controle de custos é fundamental para alcançar uma vida financeira saudável.

Em resumo, as habilidades e estratégias de um gestor financeiro são aplicáveis tanto em empresas quanto na vida pessoal. Compreender como gerenciar recursos financeiros, otimizar gastos, garantir liquidez e explorar diversas fontes de renda são passos essenciais para construir uma base financeira sólida e sustentável.

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

A independência financeira é alcançada por meio da disciplina na geração de receitas e na redução de custos. Para alcançar esse objetivo, é necessário adotar uma boa gestão financeira, maximizando a riqueza, controlando os custos e tomando decisões inteligentes sobre como investir os recursos excedentes.

Além do trabalho como fonte primária de renda, existem outras opções de geração de riqueza, como investimentos em imóveis, ações e outros ativos financeiros. Saber identificar o perfil de investidor — conservador, moderado ou agressivo — ajuda a escolher as melhores opções para investir e maximizar o retorno.

Controlar os gastos de forma inteligente é igualmente importante para garantir que as receitas sejam utilizadas de maneira eficiente, gerando excedentes que podem ser aplicados em novos investimentos. É crucial, por exemplo, adotar medidas para reduzir custos e evitar que os gastos superem os ganhos. Isso é algo que pode ser feito negociando contratos, trocando fornecedores ou controlando despesas pessoais e empresariais.

Para se obter o sucesso financeiro precisamos, portanto, entender os conceitos de investimento e o como aplicá-los para aumentar a riqueza ao longo do tempo. O controle adequado do fluxo de caixa permite um planejamento mais concreto do futuro e facilita a tomada de decisões mais seguras.

Em suma, planejamento, disciplina e entendimento claro de como suas fontes de renda e investimentos podem trabalhar em seu favor são a base para alcançar a independência financeira almejada.

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

Compreender o impacto das dívidas e dos juros é fundamental para evitar problemas financeiros. Os juros simples e compostos são duas modalidades que influenciam diretamente o montante de uma dívida ou de um investimento. Nos juros simples, o cálculo é feito apenas sobre o valor inicial, resultando em uma remuneração constante. Já os juros compostos, os juros de cada período são somados ao valor principal e os novos juros são calculados a esse montante, o que aumenta significativamente o saldo final.

Os juros compostos, em particular, são amplamente utilizados e têm um impacto significativo a longo prazo, tanto em empréstimos quanto em aplicações financeiras.

Além disso, uma análise cuidadosa das opções de crédito disponíveis, juntamente com a organização financeira, pode prevenir o endividamento excessivo e garantir uma melhor administração dos recursos.

Para quem já está endividado, é crucial organizar as finanças, de forma a estabelecer prioridades e criar um plano orçamentário sólido. Reduzir cursos e evitar novas dívidas são passos essenciais para recuperar o equilíbrio financeiro. Automatizar pagamentos e controlar gastos também ajuda a evitar atrasos e juros adicionais.

Portanto, o planejamento financeiro é uma ferramenta valiosa, que permite uma visão clara de entradas e saídas, e possibilita a tomada de decisões mais seguras e informadas. Investir em educação financeira e usar ferramentas como planilhas orçamentárias são alternativas eficazes para alcançar maior controle sobre as finanças pessoais e empresariais.

Exemplos de Gestão Financeira Pessoal:

- Débito Automático: Uma forma de ajudar na gestão financeira e otimizar pagamentos é configurar contas fixas em débito automático. Isso minimiza o risco de esquecer de pagar algo e incorrer em juros por atraso.
- Plano de Ação Orçamentária: Ter um plano orçamentário é essencial para determinar se seus objetivos financeiros são viáveis. Esse recurso permite que você avalie e monitore suas despesas e receitas, garantindo que você esteja no caminho certo. Essa

estratégia, amplamente usada nas empresas, pode ser aplicada na vida pessoal para organizar finanças e alcançar metas de maneira eficaz.

Em suma, tanto os juros simples quanto os compostos desempenham um papel crucial na forma como gerenciamos nossas finanças. Compreender suas diferenças e aplicações, juntamente com um uso responsável do crédito e práticas financeiras organizadas, pode levar a decisões financeiras mais saudáveis e sustentáveis, tanto no âmbito pessoal quanto empresarial.

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

O planejamento financeiro é indispensável para transformar sonhos em realidade. Estabelecer metas claras e criar um plano de ação é o primeiro passo para alcançar objetivos de longo prazo, como uma viagem, a compra de um imóvel ou a aposentadoria. Manter hábitos financeiros saudáveis, como monitorar o fluxo de caixa, criar reservas para emergências e evitar mitos comuns sobre finanças, proporciona segurança e estabilidade. Ao alinhar o orçamento com os objetivos, é possível construir uma vida financeira sólida e satisfatória, que permita a realização de projetos pessoais e profissionais.

Para concretizar um sonho, é necessário transformá-lo em um projeto. Ou seja, criar um plano detalhado, de forma a estipular prazos e recursos financeiros necessários. Por exemplo, se o sonho é fazer uma viagem ao exterior, é importante definir quanto tempo você terá para viajar (exemplo: em um determinado feriado), calcular os custos com transporte, hospedagem e alimentação, planejar como irá acumular dinheiro ao longo do tempo. Caso seja necessário, adiar esse sonho a fim de garantir os recursos adequados para realizá-lo.

É essencial também contar com o apoio e envolvimento do grupo familiar ou social ao qual pertence. É necessário comunicar claramente os objetivos financeiros das metas propostas, ter a cooperação de todos para manter um equilíbrio nas finanças de todo o grupo.

Em suma, o sucesso na realização de sonhos depende de uma atitude positiva e disciplinada. É essencial manter o foco, controlar as finanças e sempre revisar o progresso rumo às metas. A tomada de decisões assertivas, com base em dados financeiros reais, é o que evitará frustrações futuras.

3.5.2 ESTUDANTES NA PRÁTICA

Diante das rápidas transformações tecnológicas e das necessidades emergentes de educação financeira, desenvolvemos um banner com dicas práticas para ajudar as pessoas a gerenciar melhor suas finanças no dia a dia. Sabemos que a educação financeira não faz parte do currículo escolar e, por isso, muitos tomam decisões financeiras baseadas em exemplos familiares ou de amigos, que nem sempre são os mais adequados.

Nosso objetivo é compartilhar boas práticas financeiras que podem ser aplicadas por qualquer pessoa, independentemente de sua classe social ou nível de escolaridade. O banner aborda conceitos fundamentais de finanças, como a importância de controlar os gastos, investir de forma inteligente e planejar o futuro. Com um conteúdo claro e acessível, queremos mostrar que com planejamento, disciplina e uma mentalidade positiva, é possível transformar a gestão financeira em uma ferramenta para alcançar a estabilidade e realizar sonhos.

ISSN 1983-6767

4. CONCLUSÃO

Este projeto teve como objetivo principal desenvolver uma solução para a ineficiência dos métodos tradicionais de contabilização de presenças no ambiente escolar. A proposta consiste na criação de um sistema em que o professor gera um QR Code, que será disponibilizado aos alunos presentes na sala de aula, permitindo a contabilização automática das presenças que será lido por dispositivos estrategicamente instalados disponíveis no local.

Sua construção exigiu a utilização da lógica de programação para a resolução estruturada de problemas, juntamente com o paradigma de programação orientada a objetos, visando um desenvolvimento de sistemas mais completo e organizado. Isso torna o sistema do projeto mais robusto e com uma manipulação mais eficaz. Além disso, a estrutura envolve a modelagem de dados em conjunto com Business Intelligence (BI), que consiste na organização e criação de informações em uma representação visual. Esses foram temas importantes abordados ao longo do projeto, com o objetivo de criar uma plataforma completa e proporcionar uma visualização intuitiva das presenças registradas pelos alunos e professores da instituição de ensino.

Na parte de lógica de programação a principal dificuldade foi estruturar uma base inicial que integrasse o código em programação orientada a objetos (POO) ao banco de dados (modelagem), de forma a garantir uma conexão eficaz entre ambos.

Na programação orientada a objetos, houve desafios para fazer com que as funções acessassem corretamente os dados integrados ao banco de dados. Embora não fosse uma exigência direta da POO, essa integração foi necessária para que todas as funcionalidades presentes no menu, e também no login, tivessem acesso aos dados do sistema, como o número de matrícula do professor, registro de aluno, senhas, grade horária, faltas e presenças.

Na modelagem de dados o maior obstáculo foi assegurar que todas as tabelas contivessem chaves estrangeiras, o que permitiu consultas mais precisas e consistentes. Além disso, populá-las para simular um cenário realista demandou imenso esforço.

Em Business Intelligence (BI), a dificuldade principal foi integrar o banco de dados para exibir corretamente as informações e criar medidas e tabelas na Dashboard do Power BI, o que é essencial para dar visibilidade aos dados e possibilitar a tomada de decisões.

REFERÊNCIAS

ALURA. **O que é e para que serve a modelagem de dados?** Disponível em: <https://www.alura.com.br/artigos/modelagem-de-dados?srsltid=AfmBOoodPmKi-ZTQyBNRPP_NCAKqKA_pe-VkcHrytdLlHRtXKNiMoFOh>. Acesso em: 28 set. 2024.

ALURA. **O que é um dashboard: tipos, objetivos, benefícios, como criar e ferramentas** Disponível em: <Tudo sobre um dashboard: o que é, como criar e quais ferramentas usar | Alura>. Acesso em: 24 out. 2024.

ALURA. MySQL: executando Procedures. Disponível em: <https://www.alura.com.br/artigos/mysql-executando-procedures/> . Acesso em: 08 out. 2024.

ALURA**. Business Intelligence: o que é BI, o que faz e como usar no dia a dia** Disponível em: <https://www.alura.com.br/artigos/business-intelligence-bi#:~:text=De%20forma%20geralC%20Business%20Intelligence,de%20pessoas%20e%20de%20tecnologia>. Acesso em: 30 set. 2024.

DEVMEDIA. Introdução ao Diagrama de Entidade Relacionamento (DER). Disponível em: <https://www.devmedia.com.br/introducao-ao-diagrama-entidade-relacionamento-der/13287> . Acesso em: 31 set. 2024.

DICIO. **Classe**. Disponível em: <https://www.dicio.com.br/classe/>. Acesso em: 30 set. 2024.

FIVE PERFORMANCE DIGITAL. **O que é: Organização de Dados**. Disponível em: <https://fiveperformancedigital.com.br/glossario/o-que-e-organizacao-de-dados/#:~:text=Conclus%C3%A3o/> . Acesso em: 29 set. 2024.

MANZANO, José Augusto Navarro G.; OLIVEIRA, Jayr Figueiredo de. **Algoritmos - Lógica para Desenvolvimento de Programação de Computadores.** Rio de Janeiro: Érica, 2019. E-book. ISBN 9788536531472. Disponível em:

<https://integrada.minhabiblioteca.com.br/#/books/9788536531472> Acesso em: 30 set. 2024.

ORACLE. **O que Business Intelligence significa para você?** Disponível em: <https://www.oracle.com/br/what-is-business-intelligence>. Acesso em: 29 set. 2024.

POSTGRESQL. Diagrama Entidade Relacionamento (DER) - Como Criar? Disponível em: https://www.postgresqltutorial.com/entidade-relacionamento-der> . Acesso em: 04 out. 2024.

ROCKETSEAT**. Lógica de Programação para iniciantes em programação.** *Rocketseat*, s.d. Disponível em: <ttps://www.rocketseat.com.br/blog/artigos/post/logica-de-programacao-para-iniciantes-em-programacao>. Acesso em: 26 out. 2024.

RIZZO, Beatriz; ROBERTO, João; MENATO, Lívia; VITOR, Luís; PROCÓPIO, Maria; SALES, Pedro. **IoT: Facilitando Conexões Inteligentes.** 2024.1. 37f. Projeto Integrado - Escola de Negócios, Centro Universitário da Fundação de Ensino Octávio Bastos, São João da Boa Vista

W3SCHOOLS. SQL Tutorial. Disponível em: <https://www.w3schools.com/sql>. Acesso em: 26 out. 2024.

ISSN 1983-6767

ANEXOS

ANEXO 1 - CODIFICAÇÃO POO

unipresence > aluno.py > ...

You, 34 minutes ago | 1 author (You)

```python
from unipresence.usuario import Usuario


You, 34 minutes ago | 1 author (You)
class Aluno(Usuario):
    def __init__(self):
        super().__init__(tipo="aluno")
        self.ra = None
        self.localizacao = None

```

unipresence > professor.py > ...

You, 20 hours ago | 1 author (You)

```python
from unipresence.aluno import Usuario


You, 20 hours ago | 1 author (You)
class Professor(Usuario):
    def __init__(self):
        super().__init__(tipo="professor")

```

unipresence > usuario.py > Usuario > _verificar_credenciais

You, 19 hours ago | 1 author (You)

```python
from unipresence.bd import ConexaoBanco
from mysql.connector import Error


You, 19 hours ago | 1 author (You)
class Usuario:
    def __init__(self, tipo):
        self.tipo = tipo
        self.id = None
        self.senha = None
        # chama o construtor de Pessoa passando "aluno" como parâmetro do tipo de pessoa

    def login(self):
        if self.tipo == "aluno":
            self.id = input("Digite seu RA: ").strip()
        else:
            self.id = input("Digite sua matrícula(RM): ").strip()

        self.senha = input("Digite sua senha: ")

        return self._verificar_credenciais()

    def _verificar_credenciais(self):      You, 20 hours ago • novos arquivos: menu e usuario
        conn = ConexaoBanco.get_connection()
        if not conn:
            print("Não foi possível conectar ao banco de dados.")
            return False
        try:
            cursor = conn.cursor(dictionary=True)
            if self.tipo == "aluno":
                query = "SELECT * FROM aluno WHERE ra = %s AND senha = %s"
            else:
                query = "SELECT * FROM professor WHERE matricula = %s AND senha = %s"

            cursor.execute(query, (self.id, self.senha))
            usuario = cursor.fetchone()

            if usuario:
                print("Login bem sucedido")
                return True
            else:
                print("Credenciais incorretas.")
                return False
        except Error as e:
            print(f"Erro ao verificar credenciais: {e}")
            return False
```

unipresence > validacao_geografica.py > ...

```python
# You, 36 minutes ago | 1 author (You)
from geopy.distance import geodesic

#       You, 2 weeks ago • atualização validação geografica
# You, 2 weeks ago | 1 author (You)
class LocalizacaoCampus:
    def __init__(self):
        self.__campus_mantiqueira_coords = (-21.966571251651015, -46.77271231380196)
        self.__campus_fazenda_coords = (-21.959124918142102, -46.75541626064883)
        self.__palmeiras_coords = (-21.973273070543343, -46.79749880108253)

    def get_campus(self, nome):
        if nome == "mantiqueira":
            return self.__campus_mantiqueira_coords
        elif nome == "fazenda":
            return self.__campus_fazenda_coords
        elif nome == "palmeiras":
            return self.__palmeiras_coords
        else:
            raise ValueError("Campus não encontrado")


# You, 2 weeks ago | 1 author (You)
class LocalizacaoAluno:
    def __init__(self, latitude: float, longitude: float):
        self.__latitude = latitude
        self.__longitude = longitude

    def get_coordenadas(self):
        return (self.__latitude, self.__longitude)


# You, 36 minutes ago | 1 author (You)
class DistanciaAutorizada(LocalizacaoAluno):
    def __init__(self, localizacao_aluno: LocalizacaoAluno, nome_campus: str):
        self.localizacao_aluno = localizacao_aluno
        self.campus = LocalizacaoCampus().get_campus(nome_campus)

    def local_autorizado(self):
        raio_permitido = 0.3  # 50 metros
        coordenadas_aluno = self.localizacao_aluno.get_coordenadas()
        distancia = geodesic(coordenadas_aluno, self.campus).km
        return distancia <= raio_permitido

    def validar_localizacao(self, campus_nome):
        if not self.localizacao:
            self.localizacao = self.obter_localizacao()
        coordenadas_campus = LocalizacaoCampus().get_campus(campus_nome)
        validacao = DistanciaAutorizada(self.localizacao, coordenadas_campus)
        if validacao.local_autorizado():
            return True
        else:
            return False
```

```python

def validar_localizacao(self, campus_nome):
    if not self.localizacao:
        self.localizacao = self.obter_localizacao()
    coordenadas_campus = LocalizacaoCampus().get_campus(campus_nome)
    validacao = DistanciaAutorizada(self.localizacao, coordenadas_campus)
    if validacao.local_autorizado():
        print("Localização autorizada.")
        return True
    else:
        print("Localização fora do permitido")
        return False
```

unipresence > menu.py > ...

You, 1 second ago | 1 author (You)

```python
from unipresence.bd import ConexaoBanco
from tabulate import tabulate

# from unipresence.aluno import Aluno
from unipresence.validacao_geografica import (
    LocalizacaoAluno,
    DistanciaAutorizada,
    # LocalizacaoCampus,
)


class Menu:
    def __init__(self, usuario):
        self.usuario = usuario
        self.conn = ConexaoBanco.get_connection()
        if self.conn:
            self.cursor = self.conn.cursor()

    def exibe_grade_horaria(self):
        if self.usuario.tipo == "aluno":
            view_query = "SELECT * FROM grade_horaria_aluno_semana WHERE RA = %s"
            self.cursor.execute(view_query, (self.usuario.id,))
            resultados = self.cursor.fetchall()

            table = []
            # Exibir os resultados
            for linha in resultados:
                (
                    Aluno,
                    RA,
                    Curso,
                    Disciplina,
                    Dia_da_Semana,
                    Inicio,
                    Fim,
                    Modulo,
                ) = linha
                table.append([Disciplina, Dia_da_Semana, Inicio, Fim])
            print(
                tabulate(
                    table, headers=["Disciplina", "Dia da Semana", "Inicio", "Fim"]
                )
            )

        elif self.usuario.tipo == "professor":
            view_query = (
                "SELECT * FROM grade_horaria_professor_semana WHERE Matricula = %s"
            )
            self.cursor.execute(view_query, (self.usuario.id,))
            resultados = self.cursor.fetchall()

            table = []
            # Exibir os resultados
            for linha in resultados:
                (
                    Professor,
```

unipresence > menu.py > ...

```
class Menu:
    def exibe_grade_horaria(self):
                        Matricula,
                        Curso,
                        Disciplina,
                        Dia_da_Semana,
                        Inicio,
                        Fim,
                        Modulo,
                        Periodo,
                    ) = linha
                    table.append([Dia_da_Semana, Disciplina, Inicio, Fim])
                print(
                    tabulate(
                        table, headers=["Dia da Semana", "Disciplina", "Inicio", "Fim"]
                    )
                )

    def sair(self):
        print("Saindo do sistema...")


You, 41 minutes ago | 1 author (You)
class MenuAluno(Menu):
    def __init__(self, usuario):
        super().__init__(usuario)

    def menu_layout(self):
        print("\n--- Menu Aluno ---")
        print("1. Grade Horária")
        print("2. Faltas Totais")
        print("3. Presenças Totais")
        print("4. Escanear QR Code")
        print("5. Sair")

    def executar_opcao(self, opcao):
        # escolha = input("Escolha uma opção: ")
        if opcao == "1":
            self.exibe_grade_horaria()
        elif opcao == "2":
            self.exibe_faltas_totais()
        elif opcao == "3":
            self.exibe_presencas_totais()
        elif opcao == "4":
            self.escanear_qr_code()
        elif opcao == "5":
            self.sair()
        else:
            print("Opção inválida.")

    def exibe_faltas_totais(self):
        view_query = "SELECT * FROM historico_presenca_aluno WHERE RA = %s"
        self.cursor.execute(view_query, (self.usuario.id,))
        resultados = self.cursor.fetchall()

        table = []
        # Exibir os resultados
        for linha in resultados:
```

unipresence > menu.py > ...

```
 77  class MenuAluno(Menu):
104      def exibe_faltas_totais(self):
112              (RA, NomeAluno, Disciplina, TotalPresencas, TotalFaltas) = linha
113              table.append([Disciplina, TotalFaltas])
114          print(tabulate(table, headers=["Disciplina", "Total de Faltas"]))
115
116      def exibe_presencas_totais(self):
117          view_query = "SELECT * FROM historico_presenca_aluno WHERE RA = %s"
118          self.cursor.execute(view_query, (self.usuario.id,))
119          resultados = self.cursor.fetchall()
120
121          table = []
122          # Exibir os resultados
123          for linha in resultados:
124              (RA, NomeAluno, Disciplina, TotalPresencas, TotalFaltas) = linha
125              table.append([Disciplina, TotalPresencas])
126          print(tabulate(table, headers=["Disciplina", "Total de Presenças"]))
127
128      def escanear_qr_code(self):
129          campus_nome = input(
130              "Digite o nome do campus que está tendo aulas (mantiqueira, fazenda, palmeiras):
131          )
132          # Definindo coordenadas fictícias do aluno
133          coordenadas_aluno = LocalizacaoAluno(-21.96614140503053, -46.77420297206084)
134          validacao = DistanciaAutorizada(coordenadas_aluno, campus_nome)
135          if validacao.local_autorizado():
136              print("Localização válida para escanear o QR code!")
137          else:
138              print("Você não está dentro da área permitida. Não pode gerar o QR Code")
139
140
      You, 20 hours ago | 1 author (You)
141  class MenuProfessor(Menu):
142      def __init__(self, usuario):
143          super().__init__(
144              usuario,
145          )
146
147      def menu_layout(self):
148          print("\n--- Menu ---")
149          print("1. Grade horária")
150          print("2. Aula do dia")
151          print("3. Relatório")
152          print("4. Liberar QR Code ")
153          print("5. Sair")
154
155          # choice = input("Escolha uma opção: ")
156
157      def executar_opcao(self, opcao):
158          if opcao == "1":
159              self.exibe_grade_horaria()
160
161          elif opcao == "2":
162              self.exibir_aula_dia()
163
164          elif opcao == "3":
165              self.relatorio()
166
```

unipresence > menu.py > MenuProfessor > exibir_aula_dia

```python
class MenuProfessor(Menu):
    def executar_opcao(self, opcao):
        elif opcao == "4":
            self.liberar_qr_code()

        elif opcao == "5":
            self.sair()

    def exibir_aula_dia(self):
        view_query = "SELECT * FROM grade_horaria_professor_semana WHERE Matricula = %s"
        self.cursor.execute(view_query, (self.usuario.id,))
        resultados = self.cursor.fetchall()

        table = []
        # Exibir os resultados
        for linha in resultados:
            (
                Professor,
                Matricula,      You, 1 second ago • Uncommitted changes
                Curso,
                Disciplina,
                Dia_da_Semana,
                Inicio,
                Fim,
                Modulo,
                Periodo,
            ) = linha
            table.append([Dia_da_Semana, Disciplina])
            print(tabulate(table, headers=["Dia da semana", "Disciplina"]))

    def relatorio(self):
        query = """SELECT Disciplina, SUM(`Total Presencas`) AS TotalPresencas, SUM(`Total Faltas`) AS TotalFaltas
                   FROM historico_presenca_aluno
                   GROUP BY Disciplina"""
        self.cursor.execute(query)
        resultados = self.cursor.fetchall()
        print("Resumo por Disciplina:")
        table = []
        for linha in resultados:
            disciplina, total_presencas, total_faltas = linha
            table.append([disciplina, total_presencas, total_faltas])
        print(
            tabulate(
                table,
                headers=["Disciplina", "Total de Presenças", "Total de Faltas"],
            )
        )

    def liberar_qr_code(self):
        pass
```

ISSN 1983-6767

ANEXO 2 - CODIFICAÇÃO LÓGICA DE PROGRAMAÇÃO

```
class UserDB:
    def __init__(self):
        self.users = {}

    def add_user(self, username, password):
        self.users[username] = password

    def get_user(self, username):
        if username in self.users:
            return User(username, self.users[username])
        return None

class User:
    def __init__(self, username, password):
        self.username = username
        self.password = password

    def check_password(self, password):
        return self.password == password

user_db = UserDB()
user_db.add_user("usuario_teste", "senha_teste")

# Autenticação
class AuthModule:
    def __init__(self, user_db):
        self.user_db = user_db

    def login(self, username, password):
        user = self.user_db.get_user(username)
        if user and user.check_password(password):
            return "Login bem-sucedido"
        return "Usuário ou senha incorretos"
```

ANEXO 3 - SCRIPT SGBD MySQL

```sql
CREATE DATABASE unifeob;
USE unifeob;

-- Tabela de Cursos
CREATE TABLE curso (
    idcurso INT AUTO_INCREMENT PRIMARY KEY,
    nomeCurso VARCHAR(100) NOT NULL
);

-- Tabela de Professores
CREATE TABLE professor (
    idprofessor INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    matricula VARCHAR(20) NOT NULL UNIQUE,
    endereco VARCHAR(255),
    telefone VARCHAR(15),
    data_nascimento DATE,
    senha VARCHAR(20)

);

-- Tabela de Disciplinas (cada disciplina pertence a um curso)
CREATE TABLE disciplina (
    iddisciplina INT AUTO_INCREMENT PRIMARY KEY,
    idcurso INT NOT NULL,
    nome_disciplina VARCHAR(100) NOT NULL,
    carga_horaria INT,
    FOREIGN KEY (idcurso) REFERENCES curso(idcurso) ON DELETE CASCADE
);

-- Tabela de Grade Horária (definindo horários das disciplinas)
CREATE TABLE grade_horaria (
    id_grade INT AUTO_INCREMENT PRIMARY KEY,
    iddisciplina INT NOT NULL,
    dia_semana ENUM('Segunda', 'Terça', 'Quarta', 'Quinta', 'Sexta') NOT NULL,
    horario_inicio TIME NOT NULL,
    horario_fim TIME NOT NULL,
    FOREIGN KEY (iddisciplina) REFERENCES disciplina(iddisciplina) ON DELETE CASCADE
);

-- Tabela para associar Professores com Disciplinas, Cursos, Módulos e Períodos
CREATE TABLE professor_disciplina_curso (
    idprofessor INT NOT NULL,
    iddisciplina INT NOT NULL,
    modulo INT NOT NULL,
    periodo ENUM('Matutino', 'Vespertino', 'Noturno') NOT NULL,
    PRIMARY KEY (idprofessor, iddisciplina, modulo, periodo),
    FOREIGN KEY (idprofessor) REFERENCES professor(idprofessor) ON DELETE CASCADE,
    FOREIGN KEY (iddisciplina) REFERENCES disciplina(iddisciplina) ON DELETE CASCADE
);

-- Tabela de Alunos
```

```sql
-- Tabela de Alunos
CREATE TABLE aluno (
    idaluno INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    endereco VARCHAR(255),
    telefone VARCHAR(15),
    data_nascimento DATE,
    ra VARCHAR(20) NOT NULL UNIQUE,
    idcurso INT NOT NULL,
    modulo INT NOT NULL,  -- Módulo atual do aluno
    semestre INT NOT NULL, -- Semestre atual do aluno
    FOREIGN KEY (idcurso) REFERENCES curso(idcurso) ON DELETE CASCADE,
    senha VARCHAR(20)

);

-- Tabela para registrar Aulas específicas (ligação entre professor, grade horária e data)
CREATE TABLE aula (
    idaula INT AUTO_INCREMENT PRIMARY KEY,
    id_grade INT NOT NULL,
    idprofessor INT NOT NULL,
    data DATE NOT NULL,
    sala VARCHAR(50),
    FOREIGN KEY (id_grade) REFERENCES grade_horaria(id_grade) ON DELETE CASCADE,
    FOREIGN KEY (idprofessor) REFERENCES professor(idprofessor) ON DELETE CASCADE
);

-- Tabela de Localização para controle de presença (validar localização geográfica)
CREATE TABLE localizacao (
    idlocalizacao INT AUTO_INCREMENT PRIMARY KEY,
    descricao VARCHAR(100) NOT NULL,
    latitude DECIMAL(9,6) NOT NULL,
    longitude DECIMAL(9,6) NOT NULL,
    raio INT NOT NULL -- Raio em metros para considerar presença válida
);

-- Tabela de Registro de Presença
CREATE TABLE registro_presenca (
    idregistro INT AUTO_INCREMENT PRIMARY KEY,
    idaluno INT NOT NULL,
    idaula INT NOT NULL,
    data_leitura DATE,
    horario_leitura TIME,
    status_presenca ENUM('Presente', 'Ausente') DEFAULT 'Ausente',
    latitude DECIMAL(18, 15),
    longitude DECIMAL(18, 15),
    FOREIGN KEY (idaluno) REFERENCES aluno(idaluno) ON DELETE CASCADE,
    FOREIGN KEY (idaula) REFERENCES aula(idaula) ON DELETE CASCADE
);
```

```sql
-- Tabela para associar Alunos com a Grade Horária (disciplinas e horários específicos)
CREATE TABLE aluno_grade_horaria (
    id_aluno_grade INT AUTO_INCREMENT PRIMARY KEY,
    idaluno INT NOT NULL,
    id_grade INT NOT NULL,
    FOREIGN KEY (idaluno) REFERENCES aluno(idaluno) ON DELETE CASCADE,
    FOREIGN KEY (id_grade) REFERENCES grade_horaria(id_grade) ON DELETE CASCADE
);

-- View para o Professor visualizar sua Grade Horária por semana
CREATE VIEW grade_horaria_professor_semana AS
SELECT
    p.nome AS Professor,
    p.matricula AS Matrícula,
    c.nomeCurso AS Curso,
    d.nome_disciplina AS Disciplina,
    g.dia_semana AS "Dia da Semana",
    g.horario_inicio AS "Início",
    g.horario_fim AS Fim,
    pd.modulo AS "Módulo",
    pd.periodo AS "Período"
FROM
    professor p
JOIN
    professor_disciplina_curso pd ON p.idprofessor = pd.idprofessor
JOIN
    disciplina d ON pd.iddisciplina = d.iddisciplina
JOIN
    curso c ON d.idcurso = c.idcurso
JOIN
    grade_horaria g ON d.iddisciplina = g.iddisciplina
ORDER BY
    FIELD(g.dia_semana, 'Segunda', 'Terça', 'Quarta', 'Quinta', 'Sexta');

```

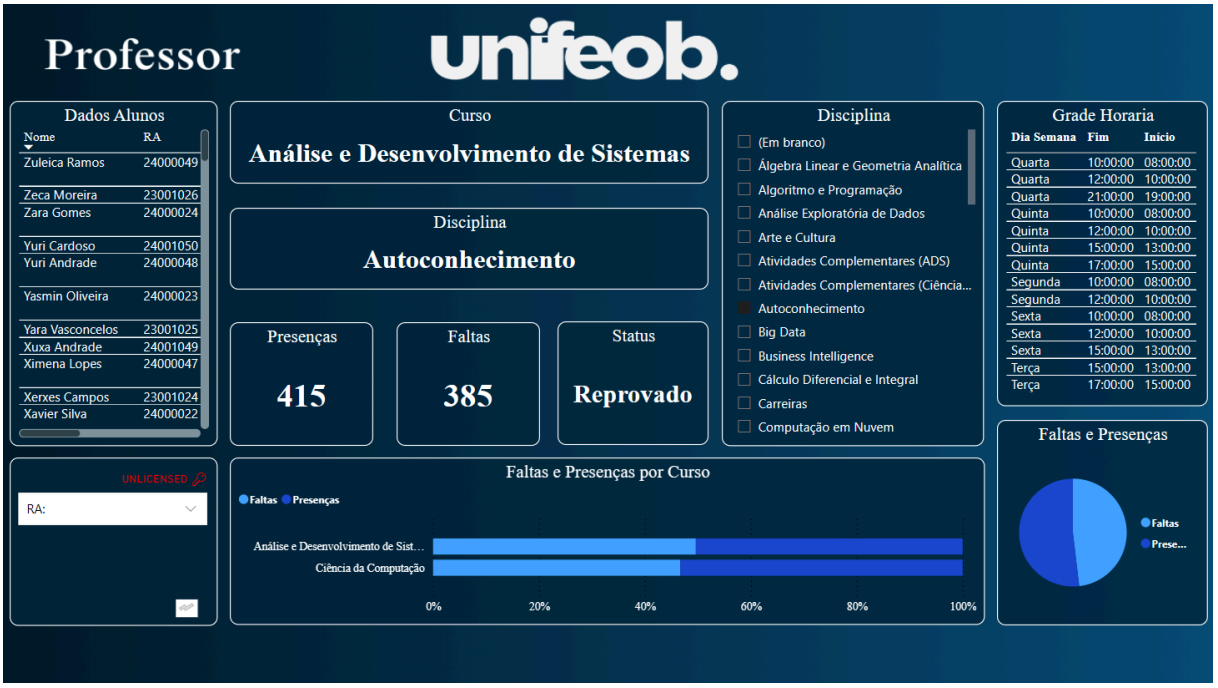
```sql
-- View para o Aluno  visualizar sua Grade Horária por semana
CREATE VIEW grade_horaria_aluno_semana AS
SELECT
    a.nome AS Aluno,
    a.ra AS RA,
    c.nomeCurso AS Curso,
    d.nome_disciplina AS Disciplina,
    g.dia_semana AS "Dia da Semana",
    g.horario_inicio AS "Início",
    g.horario_fim AS Fim,
    a.modulo AS "Módulo"
FROM
    aluno a
JOIN
    aluno_grade_horaria agh ON a.idaluno = agh.idaluno
JOIN
    grade_horaria g ON agh.id_grade = g.id_grade
JOIN
    disciplina d ON g.iddisciplina = d.iddisciplina
JOIN
    curso c ON a.idcurso = c.idcurso
ORDER BY
    FIELD(g.dia_semana, 'Segunda', 'Terça', 'Quarta', 'Quinta', 'Sexta');

-- criar View para visualizar as faltas
CREATE VIEW historico_presenca_aluno AS
SELECT
    a.ra AS RA,
    a.nome AS Aluno,
    d.nome_disciplina AS Disciplina,
    COUNT(CASE WHEN rp.status_presenca = 'Presente' THEN 1 END) AS "Total Presencas",
    COUNT(CASE WHEN rp.status_presenca = 'Ausente' THEN 1 END) AS "Total Faltas"
FROM
    registro_presenca rp
JOIN
    aula au ON rp.idaula = au.idaula
JOIN
    grade_horaria g ON au.id_grade = g.id_grade
JOIN
    disciplina d ON g.iddisciplina = d.iddisciplina
JOIN
    aluno a ON rp.idaluno = a.idaluno
GROUP BY
    a.ra, a.nome, d.nome_disciplina;

```

```sql
-- Calcular Faltas e Presenças por Disciplina usando parte do nome da disciplina
DELIMITER //

CREATE PROCEDURE calcular_faltas_presencas_disciplina_nome_parcial(
    IN ra_aluno VARCHAR(20),
    IN nome_parcial_disciplina VARCHAR(100)
)
BEGIN
    SELECT
        a.nome AS Aluno,
        d.nome_disciplina AS Disciplina,
        COUNT(CASE WHEN rp.status_presenca = 'Presente' THEN 1 END) AS "Total Presencas",
        COUNT(CASE WHEN rp.status_presenca = 'Ausente' THEN 1 END) AS "Total Faltas"
    FROM
        registro_presenca rp
    JOIN
        aula au ON rp.idaula = au.idaula
    JOIN
        grade_horaria g ON au.id_grade = g.id_grade
    JOIN
        disciplina d ON g.iddisciplina = d.iddisciplina
    JOIN
        aluno a ON rp.idaluno = a.idaluno
    WHERE
        a.ra = ra_aluno
        AND d.nome_disciplina LIKE CONCAT('%', nome_parcial_disciplina, '%')
    GROUP BY
        a.nome, d.nome_disciplina;
END //

DELIMITER ;
```

```sql
-- Calcular Faltas e Presenças por Módulo

DELIMITER //

CREATE PROCEDURE calcular_faltas_presencas_modulo_ra(
    IN ra_aluno VARCHAR(20),
    IN modulo INT
)
BEGIN
    SELECT
        a.nome AS NomeAluno,
        c.nomeCurso AS Curso,
        a.modulo AS Modulo,
        COUNT(CASE WHEN rp.status_presenca = 'Presente' THEN 1 END) AS TotalPresencas,
        COUNT(CASE WHEN rp.status_presenca = 'Ausente' THEN 1 END) AS TotalFaltas
    FROM
        registro_presenca rp
    JOIN
        aula au ON rp.idaula = au.idaula
    JOIN
        grade_horaria g ON au.id_grade = g.id_grade
    JOIN
        disciplina d ON g.iddisciplina = d.iddisciplina
    JOIN
        aluno a ON rp.idaluno = a.idaluno
    JOIN
        curso c ON a.idcurso = c.idcurso
    WHERE
        a.ra = ra_aluno
        AND a.modulo = modulo
    GROUP BY
        a.nome, c.nomeCurso, a.modulo;
END //

DELIMITER ;
```

ISSN 1983-6767

ANEXO 4 - DASHBOARD PROFESSOR



ANEXO 5 - DASHBOARD UNIFEOB