UNIVERSIDADE DO CONTESTADO – UnC

CURSO DE ENGENHARIA DE SOFTWARE

CAUÃ FILIPE GRAFF

**Protótipo para Extração, Tratamento e Padronização de Dados de Ponto Eletrônico Utilizando Python, PostgreSQL, FastAPI e React.js**

CONCÓRDIA

2025

CAUÃ FILIPE GRAFF

**Protótipo para Extração, Tratamento e Padronização de Dados de Ponto Eletrônico Utilizando Python, PostgreSQL, FastAPI e React.js**

Trabalho de Conclusão de Curso apresentado como exigência para obtenção de nota na graduação de Engenharia de Software, ministrado na Universidade do Contestado – UnC, Campus Concórdia, sob Orientação do (a) Professor (a) Moacir Solano Kichel.

**LISTA DE FIGURAS**

**Nenhuma entrada de sumário foi encontrada.**

**LISTA DE ABREVIATURAS E/OU SIGLAS**

**SUMÁRIO**

[1. INTRODUÇÃO 5](#_Toc193218172)

[1.2. OBJETIVOS 6](#_Toc193218173)

[1.3. Objetivo Geral 6](#_Toc193218174)

[1.4. Objetivos Específicos 6](#_Toc193218175)

[2 REFERENCIAL TEÓRICO 7](#_Toc193218176)

[2.1 PSICOLOGIA ORGANIZACIONAL E DO TRABALHO 7](#_Toc193218177)

[2.1.1 Histórico 7](#_Toc193218178)

[2.2 ESTRESSE NO TRABALHO 8](#_Toc193218179)

[2.2.1 Doenças Provocadas pelo Estresse no Trabalho 9](#_Toc193218180)

[2.3 ESTRESSE NA POLÍCIA MILITAR 9](#_Toc193218181)

[3 MATERIAL E MÉTODOS 12](#_Toc193218182)

[3.1 TIPO DE PESQUISA 12](#_Toc193218183)

[3.2 SUJEITOS DA PESQUISA 12](#_Toc193218184)

[3.3 ASPECTOS ÉTICOS 12](#_Toc193218185)

[3.4 COLETA DE DADOS 13](#_Toc193218186)

[4 RESULTADOS E DISCUSSÕES 14](#_Toc193218187)

[5 CONCLUSÃO 16](#_Toc193218188)

[REFERÊNCIAS 17](#_Toc193218189)

1. INTRODUÇÃO

A transformação digital tem impulsionado diversas áreas do conhecimento, incluindo a contabilidade, que cada vez mais adota ferramentas tecnológicas para otimizar processos e minimizar erros operacionais. O grande volume de informações geradas por empresas, especialmente no que se refere à gestão de ponto eletrônico e controle de jornada de funcionários, demanda soluções eficientes para organização e integração desses dados em sistemas contábeis. Conforme Padoveze (2019), a automação de processos contábeis não apenas reduz falhas humanas, mas também contribui para a eficiência e segurança das informações tratadas. De acordo com Rezende (2018), a digitalização e automação dessas tarefas são fundamentais para aumentar a produtividade e melhorar a qualidade das informações processadas

Atualmente, muitos departamentos contábeis ainda realizam manualmente a importação de dados de frequência de funcionários, o que pode levar a inconsistências fiscais e trabalhistas. Além disso, os formatos variados de arquivos utilizados pelas empresas, como *Portable Document File* ou Arquivo de Documento Portáil (PDF), *Comma-Separeted-Values* ou Valores Separados por Vírgula (CSV) e planilhas eletrônicas, dificultam a padronização e integração desses dados com sistemas de *Enterprise Resource Planning* ou Planejamento de Recursos Empresarias (ERP).

Segundo Laudon e Laudon (2021), os sistemas ERP são plataformas integradas que automatizam processos de negócios em diversas áreas de uma organização, como finanças, recursos humanos e contabilidade, promovendo uma base de dados única e a padronização das informações. Esses sistemas permitem que empresas tenham maior controle sobre suas operações e otimizem a tomada de decisão com base em dados consistentes e em tempo real.

Para o desenvolvimento do protótipo proposto, serão utilizadas tecnologias como Python, devido à sua robustez em processamento de dados e extração de informações de documentos estruturados e não estruturados (McKinney, 2018), além de bibliotecas como Pandas para manipulação de dados, PyPDF2 para leitura de arquivos PDF e OpenPyXL para interação com planilhas do Excel. Essas ferramentas são amplamente reconhecidas pela comunidade de ciência de dados por sua flexibilidade e eficiência no tratamento de grandes volumes de dados (VanderPlas, 2016).  
 Para o *frontend* será desenvolvido com o *framework* React.js, permitindo uma interface interativa e responsiva. Já o *backend* será construído com o *framework* FastAPI, que oferece rapidez na criação de APIs RESTful em Python. O banco de dados utilizado será o PostgreSQL, escolhido por sua robustez, confiabilidade e recursos avançados, sendo ideal para aplicações que exigem maior desempenho e escalabilidade.

Nesse contexto, este trabalho propõe o desenvolvimento de um protótipo capaz de extrair, organizar e estruturar dados provenientes de arquivos de ponto, convertendo-os para um layout padronizado e adequado à importação em sistemas ERP. A implementação desse protótipo busca reduzir a necessidade de intervenções manuais, garantindo maior confiabilidade e eficiência no tratamento das informações. Além disso, Martins (2020) destaca que a adoção de tecnologias inovadoras permite que profissionais contábeis se concentrem em atividades estratégicas, deixando tarefas repetitivas e operacionais para sistemas automatizados.

Outro fator relevante é a economia de tempo e recursos. De acordo com Santos e Almeida (2022), empresas que adotam soluções tecnológicas para integração de dados conseguem reduzir o tempo gasto em tarefas operacionais em até 40%, permitindo que os profissionais direcionem esforços para análises mais estratégicas e consultivas. Além disso, a automação contribui para a redução de erros humanos, garantindo maior confiabilidade na geração de relatórios contábeis e fiscais.

A contabilidade moderna exige ferramentas que permitam a integração de diferentes fontes de dados de maneira rápida e precisa, eliminando a necessidade de processos manuais demorados e suscetíveis a falhas. Conforme aponta Silva (2021), empresas que não investem em automação tendem a enfrentar dificuldades na gestão de informações, o que pode impactar diretamente na tomada de decisão e na eficiência dos processos internos.

A gestão do ponto eletrônico é uma obrigação legal regida por normas trabalhistas, como a Portaria 671/2021 do Ministério do Trabalho, que estabelece critérios para o registro da jornada de trabalho. De acordo com Souza e Mendes (2020), o controle de ponto é essencial para assegurar o cumprimento de direitos trabalhistas e evitar passivos judiciais. No entanto, a diversidade de sistemas e formatos de exportação desses dados pode dificultar sua integração com sistemas contábeis, tornando a automação uma necessidade estratégica.

Dessa forma, a implementação de um sistema automatizado para organização e importação de dados de ponto pode trazer impactos positivos não apenas para empresas contábeis, mas também para organizações que precisam gerenciar essas informações de maneira eficiente. Segundo Costa e Ribeiro (2023), a adoção de ferramentas digitais para gestão de dados permite maior conformidade com legislações trabalhistas, assegurando que informações como horas trabalhadas, adicional noturno e horas extras sejam calculadas corretamente e sem falhas.

A justificativa para este estudo reside na necessidade crescente das empresas de automatizar processos internos, garantindo maior agilidade e precisão na gestão contábil. Segundo Oliveira e Silva (2021), a digitalização dos processos contábeis melhora a conformidade fiscal e reduz riscos associados ao manuseio manual de informações sensíveis. A implementação desse protótipo pode trazer impactos positivos para empresas e escritórios de contabilidade, garantindo maior eficiência e precisão na gestão de dados.

A utilização de ferramentas como Python, Pandas, PyPDF2, OpenPyXL, FastAPI, PostgreSQL e React.js é suficiente para consolidar a proposta de um protótipo capaz de automatizar a extração, tratamento e padronização dos dados.

## OBJETIVOS

### Objetivo Geral

Desenvolver um protótipo funcional capaz de automatizar a extração, organização, tratamento e padronização dos dados de ponto eletrônico para posterior importação no sistema ERP, utilizando tecnologias como Python, PostgreSQL, FastAPI e React.js.

### Objetivos Específicos

- Realizar uma revisão bibliográfica sobre os seguintes temas: sistemas de folha de ponto eletrônico, Engenharia de Software, prototipação, modelagem UML, Python, FastAPI, React.js banco de dados e ferramentas de desenvolvimento;

- Efetivar o levantamento dos requisitos do sistema

- Representar os modelos de dados e do sistema utilizando UML.

- Desenvolver um protótipo funcional capaz de automatizar a extração e tratamento dos dados, Python, PostgreSQL.

Realizar testes de funcionamento e utilização do protótipo.

# 2 REFERENCIAL TEÓRICO

**2.1. Sistemas de Controle de Ponto Eletrônico**

Os sistemas de controle de ponto eletrônico são ferramentas tecnológicas utilizadas para registrar e monitorar a jornada de trabalho dos colaboradores em uma organização. Seu principal objetivo é garantir o cumprimento da carga horária estabelecida, bem como registrar entradas, saídas, intervalos e horas extras de forma segura, precisa e automatizada. Esses registros são fundamentais tanto para o cumprimento da legislação trabalhista quanto para a geração de dados que subsidiem o setor contábil e de recursos humanos.

Segundo Souza e Mendes (2020), o ponto eletrônico é um recurso indispensável na administração moderna, pois contribui para a redução de fraudes, aumenta a confiabilidade das informações e facilita a auditoria dos dados registrados. Além disso, ele representa uma alternativa mais eficiente e segura em comparação aos métodos manuais, como livros de ponto e cartões mecânicos, que estão suscetíveis a erros e manipulações.

A legislação brasileira regula o uso do ponto eletrônico por meio da Portaria nº 671/2021, do Ministério do Trabalho e Previdência, que estabelece regras para o registro da jornada de trabalho por meio de sistemas alternativos de controle. De acordo com essa norma, os empregadores podem utilizar diferentes tipos de registro eletrônico, como o Registrador Eletrônico de Ponto Convencional (REP-C), o Alternativo (REP-A) e o por Programa (**REP-P)**, desde que cumpram os requisitos legais e garantam a integridade e a inviolabilidade das informações.

A diversidade de sistemas disponíveis no mercado, bem como os variados formatos de exportação de dados, representa um desafio significativo para a integração dessas informações com sistemas contábeis. Os registros de ponto são frequentemente exportados em formatos como PDF, CSV e planilhas, os quais nem sempre seguem uma estrutura padronizada. Essa falta de uniformidade dificulta a importação direta para sistemas de folha de pagamento e ERP, exigindo, muitas vezes, intervenção manual para ajustes e correções.

Costa e Ribeiro (2023) destacam que a ausência de integração entre os sistemas de controle de ponto e os sistemas contábeis pode gerar atrasos no processamento da folha de pagamento, erros nos cálculos de horas extras e adicionais, além de impactar negativamente na conformidade com obrigações trabalhistas. Esses fatores tornam a automação e a padronização do fluxo de dados entre os sistemas uma necessidade estratégica para organizações que buscam maior eficiência operacional e segurança jurídica.

Além disso, é importante destacar a existência de sistemas legados de ponto eletrônico, ou seja, softwares antigos ainda em uso por diversas organizações, muitas vezes desenvolvidos sob tecnologias obsoletas e com estruturas rígidas de dados. Conforme Oliveira e Cruz (2020), esses sistemas, embora ainda operacionais, apresentam limitações significativas quanto à integração com plataformas modernas, dificultando a extração e o tratamento automatizado das informações. A falta de compatibilidade com padrões atuais de exportação de dados, bem como a ausência de APIs ou interfaces programáveis, impõe desafios adicionais para a automação e a padronização dos registros de jornada.

Dessa forma, observa-se que os sistemas de ponto eletrônico desempenham um papel essencial na gestão de pessoas e no cumprimento das normas legais. Entretanto, sua real efetividade depende da capacidade das empresas em integrar esses dados com outras plataformas corporativas, evitando retrabalho, aumentando a confiabilidade das informações e reduzindo o risco de inconformidades fiscais e trabalhistas.

**2.2. Engenharia de Software**

A Engenharia de Software é uma área da computação voltada ao desenvolvimento sistemático, disciplinado e quantificável de softwares. Segundo Pressman (2016), trata-se da aplicação de uma abordagem estruturada ao desenvolvimento, operação e manutenção de software, com foco na qualidade, produtividade e na redução de custos e riscos. Essa disciplina abrange processos, métodos, ferramentas e práticas que auxiliam na construção de soluções tecnológicas confiáveis e eficientes.

De acordo com Sommerville (2011), a Engenharia de Software envolve atividades como especificação de requisitos, projeto, codificação, testes e manutenção de sistemas. A adoção dessas práticas permite maior controle e previsibilidade no desenvolvimento, favorecendo a criação de sistemas que atendam às necessidades reais dos usuários.

Os processos de software podem seguir diferentes modelos de ciclo de vida, como a cascata, incremental, espiral ou ágil, cada um adequado a contextos e projetos específicos. A escolha do modelo depende da complexidade do sistema, dos requisitos dos usuários e do tempo disponível para o desenvolvimento.

**2.3. Prototipação**

A prototipação é uma abordagem amplamente utilizada na Engenharia de Software para desenvolver versões preliminares de um sistema, com o objetivo de validar funcionalidades, levantar requisitos e coletar feedback de usuários antes da construção final do produto. Essa técnica permite visualizar, testar e ajustar funcionalidades essenciais em ciclos iterativos, promovendo uma maior aderência às necessidades reais dos usuários.

De acordo com Sommerville (2011), a prototipação é essencial para o desenvolvimento incremental de sistemas, pois facilita a comunicação entre usuários e desenvolvedores, especialmente em contextos onde os requisitos são incertos ou mutáveis. Protótipos podem ser de baixa fidelidade como esboços ou de alta fidelidade com partes funcionais já implementadas.

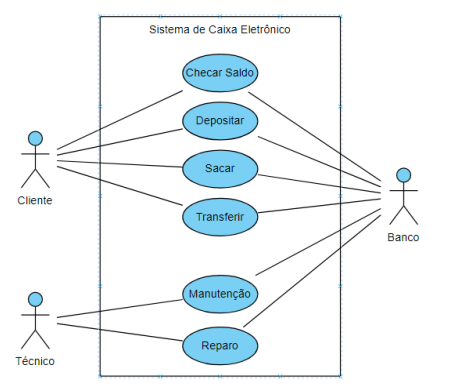
No presente trabalho, a prototipação será aplicada na criação de uma interface funcional que represente o processo de extração, organização e padronização de dados de ponto eletrônico, com o objetivo de validar os requisitos antes da implementação definitiva do sistema.

**2.4. Modelagem UML**

A UML (Unified Modeling Language) é uma linguagem de modelagem padronizada que permite representar graficamente os aspectos estruturais e comportamentais de um sistema de software. Seu principal objetivo é fornecer uma forma visual de documentar, projetar e comunicar a arquitetura de sistemas de forma clara e compreensível para desenvolvedores, analista. Segundo Larman (2007), a UML é uma ferramenta essencial para o processo de desenvolvimento de software, pois facilita a validação dos requisitos e o planejamento da implementação.

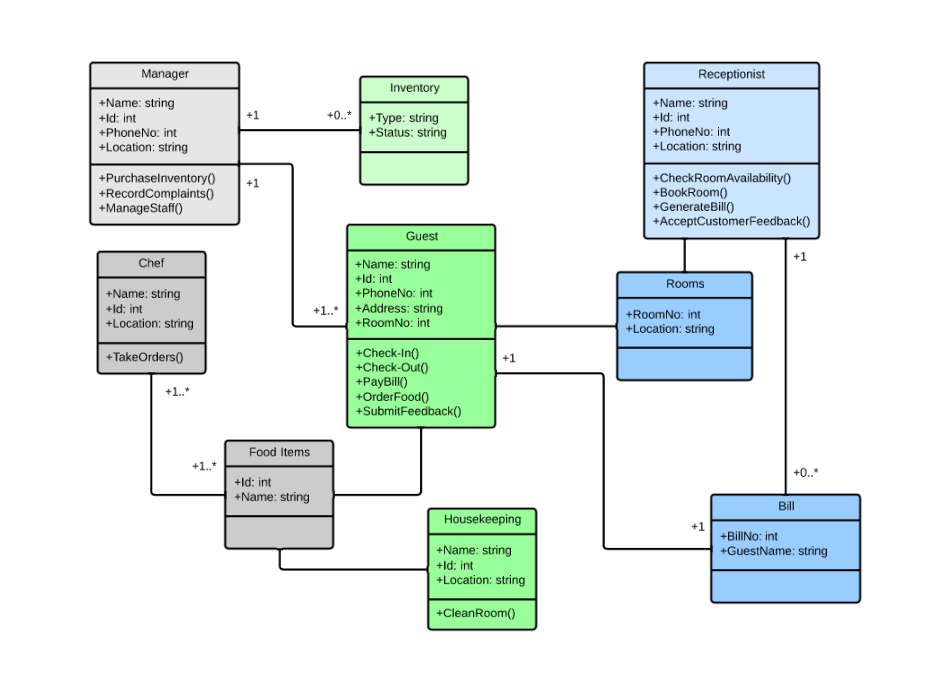
No contexto deste trabalho, a UML será utilizada para representar os principais componentes e funcionalidades do sistema proposto, garantindo maior clareza e organização no projeto. Tendo seus principais diagramas Caso de uso, Classes e Atividades.

Figura 1 – Diagrama de Caso de uso

Fonte: linkedin (2024)

Como apresentado na figura 1, um diagrama de caso de uso é um tipo de diagrama na UML que descreve as funcionalidades ou recursos que um sistema oferece do ponto de vista dos usuários. Ele mostra as interações entre os atores e o sistema em si, destacando os diferentes casos de uso ou cenários de uso que descrevem as operações que o sistema pode realizar em resposta a entradas dos atores. Esses diagramas são frequentemente usados na fase de análise para capturar os requisitos do sistema de uma forma compreensível para todas as partes interessadas.

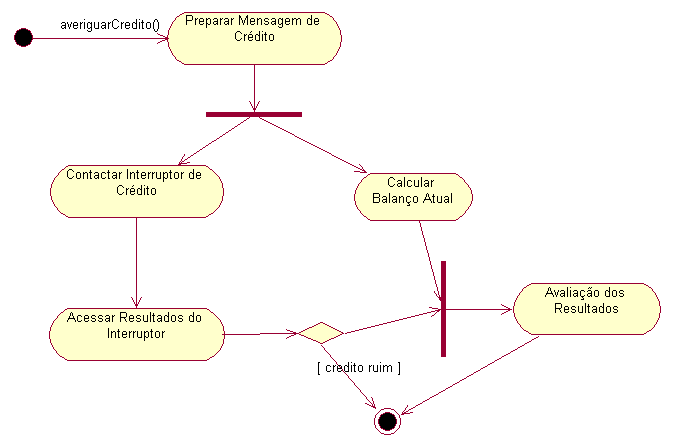
Figura 2 – Diagrama de Classes



Fonte: lucidchart (2025)

O Diagrama de Classes representa a estrutura estática do sistema conforme representado na figura 2, exibindo as classes, seus atributos, métodos e os relacionamentos entre elas. Esse tipo de diagrama é essencial para projetar a arquitetura lógica do sistema e orientar a programação orientada a objetos. De acordo com Pressman (2016), o diagrama de classes fornece uma visão detalhada das entidades do sistema e seus vínculos, como herança, associação e composição.

Figura 2 – Diagrama de Classes



Fonte: ufmg (2025)

O Diagrama de Atividades representa o fluxo de trabalho ou a lógica de execução de um processo dentro do sistema, modelando decisões, ações e paralelismos. Ele é particularmente útil para descrever o comportamento dinâmico de funcionalidades, como a execução de um processo do início ao fim, passo a passo.

**2.5. Banco de Dados**

Os bancos de dados são estruturas essenciais para o armazenamento, organização e recuperação eficiente de informações. Eles são projetados para manter dados consistentes, acessíveis e seguros em sistemas computacionais. Segundo Elmasri e Navathe (2011), um banco de dados é uma coleção de dados inter-relacionados, organizada de forma lógica e estruturada, com o objetivo de atender às necessidades de uma aplicação específica.

**2.5.1. SGBD**

Para que esses dados possam ser manipulados de maneira eficaz, utiliza-se um Sistema de Gerenciamento de Banco de Dados (SGBD), que é um conjunto de programas responsáveis por gerenciar o acesso, inserção, exclusão, atualização e recuperação dos dados armazenados. De acordo com Date (2004), os SGBDs também são responsáveis por garantir propriedades fundamentais de segurança, integridade e concorrência, especialmente em ambientes multiusuários.

Existem diversos tipos de SGBDs, cada um projetado para atender a necessidades específicas os SGBD Relacional (RDBMS) que armazenam dados em tabelas com linhas e colunas, permitindo relações entre diferentes conjuntos de dados. Exemplos: Oracle, MySQL, Microsoft SQL Server, PostgreSQL.  
**2.5.2. PostgreSQL**

O PostgreSQL é um sistema gerenciador de banco de dados relacional de código aberto, amplamente reconhecido por sua estabilidade, escalabilidade e conformidade com os padrões da linguagem SQL. Diferentemente de outros SGBDs, o PostgreSQL também oferece suporte a extensões como tipos de dados personalizados, funções escritas em diferentes linguagens (como PL/pgSQL e Python), e funcionalidades avançadas como transações complexas, controle de concorrência multiversão (MVCC) e replicação de dados (POSTGRESQL, 2024).

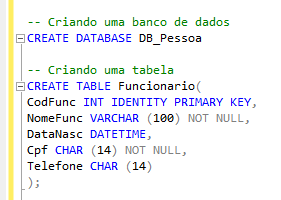
Sua robustez o torna uma escolha comum em aplicações de missão crítica, além de ser amplamente utilizado em ambientes acadêmicos e corporativos. Outro ponto de destaque é a comunidade ativa de desenvolvedores, que contribui continuamente para melhorias de desempenho, segurança e compatibilidade com novas tecnologias.

**2.5.2. SQL**

A linguagem SQL (Structured Query Language) é o padrão adotado para gerenciamento de dados em sistemas de bancos de dados relacionais. Criada nos anos 1970 pela IBM, a SQL foi projetada para possibilitar a definição, manipulação e controle de dados estruturados, permitindo que os usuários interajam com os bancos de dados de forma declarativa — ou seja, descrevendo o que se deseja obter e não como obter (DATE, 2004).

A SQL é composta por vários subconjuntos de comandos, entre os quais se destacam:

*Data Definition Language* (DDL) um subconjunto da linguagem SQL utilizado para definir e modificar estruturas de dados, como bancos de dados, tabelas, índices, views e relacionamentos. Comandos como CREATE, ALTER e DROP fazem parte desse conjunto e são fundamentais para a modelagem do banco de dados em qualquer sistema relacional.

Figura 3 – Exemplo Data Definition Language

jvilar.wordpress.com Fonte: ufmg (2025)

A Figura 3 apresenta um exemplo prático da utilização da DDL para criação de um banco de dados chamado DB\_Pessoa, seguido da criação de uma tabela denominada Funcionario.

A *Data Manipulation Language* (DML) é o subconjunto da linguagem SQL responsável pela manipulação dos dados dentro das tabelas. Por meio dela, é possível realizar operações como inserção (INSERT), atualização (UPDATE), exclusão (DELETE) e consulta (SELECT). Essas instruções são fundamentais para a interação entre o sistema e o banco de dados durante a execução de aplicações.

Figura 3 – Exemplo Data Manipulation Language



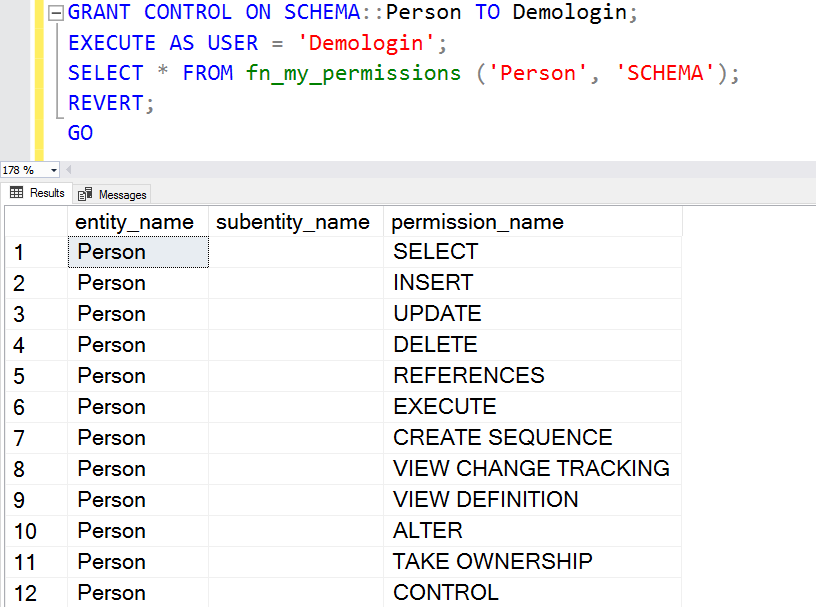
Fonte: jvilar.wordpress.com (2025)

A Figura 3 apresenta um exemplo prático da instrução SELECT aplicada à tabela TbPessoa. Tem como objetivo selecionar todos os registros da tabela e ordená-los em ordem alfabética crescente com base no campo Nome. Como resultado, são exibidas colunas como IdPessoa, Nome e DataNascimento, permitindo visualizar os dados de maneira organizada.

Esse tipo de consulta é amplamente utilizado em aplicações reais para apresentar listas de registros, realizar buscas, gerar relatórios ou alimentar interfaces visuais. A flexibilidade e simplicidade da DML tornam-na uma ferramenta poderosa na administração e exploração de dados relacionais (DATE, 2004).

O *Data Control Language* (DCL) é o subconjunto da linguagem SQL responsável por definir regras de segurança, controle de acesso e permissões dentro de um banco de dados. Os comandos principais da DCL são GRANT, para conceder permissões, e REVOKE, para removê-las. Essa camada é essencial para manter a integridade, a confidencialidade e o controle de operações dentro de sistemas multiusuários.

Figura 3 – Exemplo



Fonte: www.sqlshack.com (2025)

A Figura 3 mostra um exemplo do uso da DCL para conceder controle sobre o esquema Person ao usuário Demologin. Logo após, o sistema executa um bloco que simula a ação como o usuário Demologin e exibe suas permissões.

Com uma sintaxe relativamente simples e padronizada, a SQL tornou-se universalmente compatível com diversos SGBDs (Sistemas Gerenciadores de Bancos de Dados), como MySQL, PostgreSQL, Oracle e SQL Server. Sua utilização é fundamental em praticamente todos os sistemas que dependem de persistência e consulta eficiente de dados.

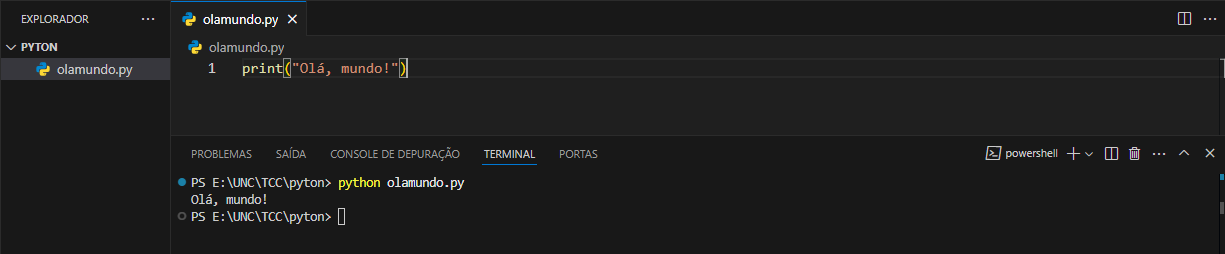
**2.6. Linguagem e Ferramentas de Desenvolvimento**

**2.6.1. Python**

As linguagens de programação e ferramentas tecnológicas desempenham papel fundamental no desenvolvimento de sistemas informatizados, pois permitem desde a construção da lógica de negócio até a apresentação visual ao usuário.

Python é uma linguagem de alto nível conhecida por sua sintaxe clara e objetiva, o que facilita seu aprendizado e aplicação em diversas áreas da computação. É amplamente utilizada para automação de processos, análise de dados, inteligência artificial e desenvolvimento web. Uma das principais vantagens do Python está em sua vasta biblioteca de pacotes, que simplificam tarefas complexas como leitura de arquivos, manipulação de dados e conexão com bancos de dados.

Figura 3 – Exemplo Código Payton



Fonte: Autor

Na figura 3 mostra a execução de um programa básico em Python utilizando o Visual Studio Code (VS Code). O arquivo olamundo.py contém uma única linha de código: Este comando imprime a mensagem "Olá, mundo!" no terminal. No terminal do VS Code (parte inferior da imagem), é possível ver que o programa foi executado com sucesso através do comando python olamundo.py .

Logo após a execução, o terminal retorna a saída "Olá, mundo!", indicando que o código foi interpretado corretamente pela linguagem Python. Este é um exemplo clássico usado como primeiro contato com a linguagem de programação, demonstrando a estrutura básica de um programa em Python.

**2.6.1.2 PyPDF2**

A biblioteca PyPDF2 é uma ferramenta amplamente utilizada em Python para a leitura, manipulação e extração de informações contidas em arquivos PDF. Com ela, é possível dividir e mesclar documentos, extrair texto de páginas específicas, criptografar e descriptografar arquivos, entre outras funcionalidades. Apesar de suas limitações em relação à precisão na extração de conteúdo de PDFs com formatação complexa, sua simplicidade e integração com outros módulos a tornam uma escolha prática para aplicações automatizadas de processamento de documentos (PyPDF2, 2024).

**2.6.1.3 pdfplumber**

Diferente do PyPDF2, o pdfplumber oferece uma abordagem mais precisa e detalhada para a extração de conteúdo de arquivos PDF. Essa biblioteca permite acessar não apenas o texto, mas também a estrutura de tabelas, localização de elementos e metadados do documento. É especialmente útil para documentos que exigem fidelidade visual, como notas fiscais, boletins ou formulários estruturados, proporcionando controle sobre o posicionamento e o formato do conteúdo extraído (PDFPlumber, 2024).

**2.6.1.2** FastAPI

O FastAPI é um *framework* moderno para o desenvolvimento de APIs RESTful utilizando Python. Projetado com foco em desempenho, simplicidade e segurança, ele adota recursos como tipagem estática e programação assíncrona para facilitar a criação de aplicações robustas. Além disso, sua integração com o OpenAPI permite a geração automática de documentação interativa, favorecendo a comunicação entre equipes de desenvolvimento e a manutenção de sistemas. Por essas razões, o FastAPI tem ganhado destaque em ambientes de desenvolvimento ágeis e modernos (FASTAPI, 2024).

**2.6.2. JavaScript**

A linguagem JavaScript é um dos pilares fundamentais do desenvolvimento web moderno, sendo executada no lado do cliente para adicionar interatividade e dinamismo às páginas. Originalmente criada para permitir ações simples no navegador, como validação de formulários e reações a eventos, a linguagem evoluiu significativamente. Atualmente, JavaScript é uma linguagem de programação completa, com suporte a paradigmas funcional, orientado a objetos e assíncrono, possibilitando o desenvolvimento de aplicações ricas, responsivas e em tempo real (FLANAGAN, 2020). Além disso, ela é amplamente compatível com todos os navegadores modernos e conta com um vasto ecossistema de bibliotecas e ferramentas que ampliam seu potencial.

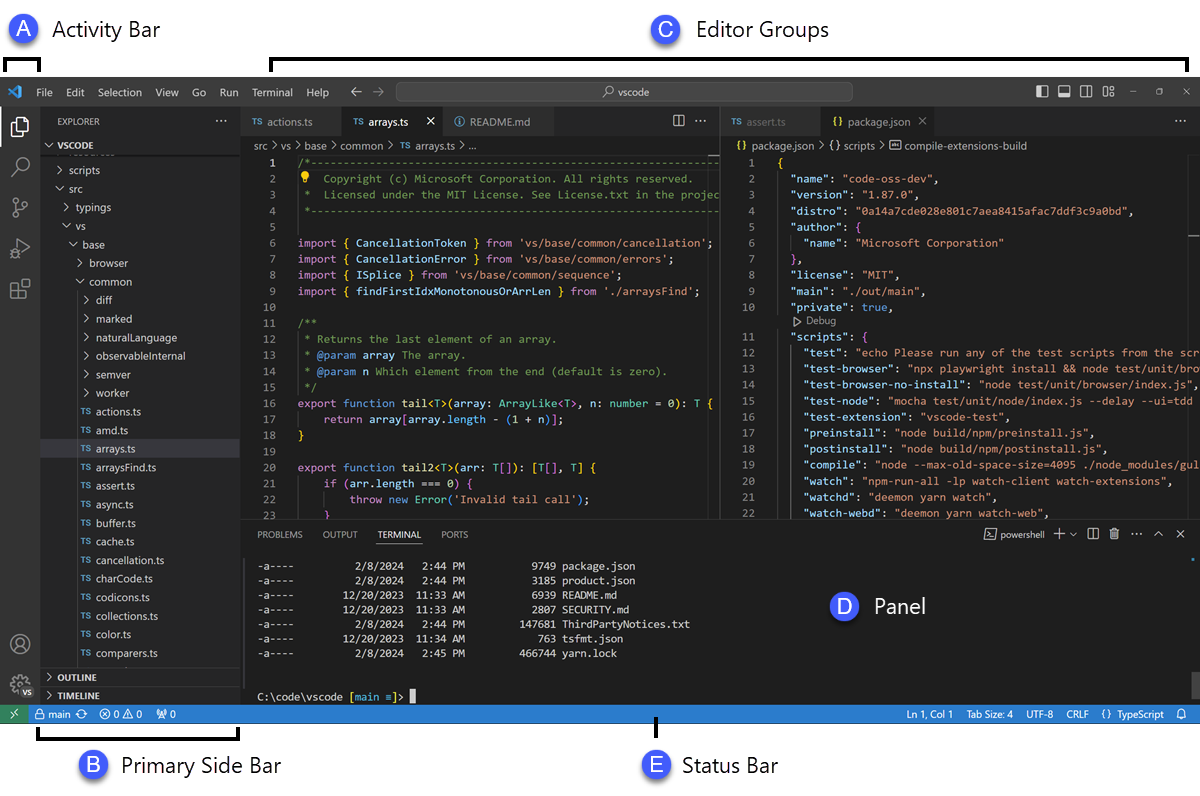
**2.6.2.1 React.js**

Entre os diversos frameworks e bibliotecas desenvolvidos com base em JavaScript, o **React.js** se destaca como uma das tecnologias mais populares para a construção de interfaces de usuário. Desenvolvido pelo Facebook, o React permite a criação de **componentes reutilizáveis**, facilitando a manutenção e escalabilidade de grandes aplicações. Ele adota o conceito de DOM virtual, otimizando a renderização de elementos na página, o que resulta em melhor desempenho e maior fluidez na navegação (REACTJS, 2024). Por essas características, o React é amplamente utilizado em projetos modernos, tanto em aplicações web quanto móveis, por meio do React Native.

**2.6.3. Visual Studio Code**

Para facilitar o processo de desenvolvimento de software, é comum a utilização de ambientes de desenvolvimento integrados, conhecidos como *Integrated Development Environments* (IDEs). Combinando com diversas ferramentas de desenvolvimento, como edição de código, compilação, depuração e testes, em um único ambiente..

Figura 4 – Visual Studio Code



Fonte: code.visualstudio.com

Um dos mais populares atualmente é o *Visual Studio Code* (VS Code), um editor de código-fonte desenvolvido pela Microsoft. Leve, rápido e altamente personalizável, o VS Code suporta diversas linguagens de programação por meio de extensões, como *Python*, *JavaScript*, C++, entre outras.

A combinação dessas tecnologias possibilita o desenvolvimento de sistemas completos e robustos, com *backend* eficiente, *frontend* responsivo e integração facilitada com bases de dados e outras aplicações. Cada linguagem, biblioteca, framework e ferramenta possui um papel específico e complementar, contribuindo para a construção de soluções escaláveis, seguras e de alto desempenho. Ao utilizar um ecossistema integrado e bem estruturado, o desenvolvedor ganha produtividade, reduz erros e garante uma melhor experiência para o usuário final. Dessa forma, o domínio dessas tecnologias é essencial para profissionais que atuam na criação de aplicações modernas e alinhadas às exigências do mercado atual.

# 3 MATERIAL E MÉTODOS

Este trabalho caracteriza-se como uma pesquisa aplicada com abordagem qualitativa, tendo como objetivo desenvolver um protótipo funcional.

A metodologia utilizada é tipo de pesquisa aplicada com foco na construção de software utilizando práticas da Engenharia de *Software* para extração e tratamento de dados de ponto eletrônico.

Para fundamentar o desenvolvimento do protótipo proposto, foi realizada uma revisão bibliográfica abrangente, contemplando temas essenciais como sistemas de folha de ponto eletrônico, fundamentos da Engenharia de Software, modelagem de dados com UML, banco de dados relacionais, sistemas gerenciadores de banco de dados (SGBD), linguagem SQL, além das tecnologias Python, FastAPI e React.js. As informações foram obtidas a partir de livros acadêmicos, artigos científicos, materiais disponíveis na biblioteca virtual da Universidade do Contestado (UNC) e fontes especializadas da internet. Essa etapa teve como objetivo reunir o conhecimento necessário para orientar as decisões técnicas e metodológicas adotadas no projeto.

Foi realizado o levantamento dos requisitos funcionais e não funcionais do sistema proposto. Essa etapa visou identificar as necessidades práticas do usuário final e definir as funcionalidades essenciais que o protótipo deveria contemplar, como a importação de arquivos em formatos variados (PDF e planilhas), extração automática de informações relevantes da jornada de trabalho, tratamento e padronização dos dados, visualização em interface gráfica e exportação para sistemas ERP.

Após a definição dos requisitos, o sistema foi modelado com a linguagem UML, a fim de representar sua estrutura e funcionamento. Foram elaborados diagramas de caso de uso, classes e atividades, permitindo visualizar as interações com o usuário, os componentes internos e o fluxo das operações. Essa modelagem serviu como base para guiar a implementação do protótipo de forma estruturada.

O desenvolvimento seguiu os modelos propostos, utilizando Python para leitura e tratamento de arquivos PDF e planilhas, com apoio das bibliotecas PyPDF2, pdfplumber, OpenPyXL e Pandas. O backend foi implementado com FastAPI, enquanto o PostgreSQL foi usado para armazenamento dos dados. A interface foi construída com React.js, possibilitando a interação do usuário com o sistema para importar, visualizar e exportar os dados processados.

Foram realizados testes de funcionamento com o objetivo de validar a eficácia do protótipo em diferentes cenários. Os testes incluíram a importação de arquivos com variações de layout, a verificação da precisão na extração de informações, a análise da padronização dos dados conforme o modelo definido, e a integridade no armazenamento no banco de dados. Também foram avaliadas as funcionalidades da interface, como a visualização dos dados processados e a exportação para formatos compatíveis com sistemas ERP.