INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO

DEPARTAMENTO ACADÊMICO DE INFORMAÇÃO E COMUNICAÇÃO
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIEL RODRIGUES PICANÇO

DATA CLEANER SERVICE - WEB SERVICE PARA APOIAR NA LIMPEZA DE DADOS

MANAUS –AM
2023

GABRIEL RODRIGUES PICANÇO

DATA CLEANER SERVICE - WEB SERVICE PARA APOIAR NA LIMPEZA DE DADOS

Trabalho de Conclusão de curso apresentado à banca examinadora do curso Superior de Tecnologia em Análise e Desenvolvimento de Sistema do Instituto Federal de educação, Ciência e Tecnologia do Amazonas – Campus Manaus Centro, como requisito para o cumprimento da disciplina TCC 2.

Orientador: Esp. Rogério Luiz Araújo Carminé

Manaus

2023

Biblioteca do IFAM – Campus Manaus Centro

P586d Picanço, Gabriel Rodrigues.
Data Cleaner Service - Web Service para apoiar na limpeza de dados / Gabriel Rodrigues Picanço. – Manaus, 2023.
52 p. : il. color.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistema) – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Centro, 2023.
Orientador: Prof. Esp. Rogério Luiz Araújo Carminé.

1. Desenvolvimento de sistema. 2. Serviço web. 3. Limpeza de dados. 4. Padronização de dados. 5. Python. I. Carminé, Rogério Luiz Araújo. (Orient.) II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas III. Título.

CDD 005.3

Elaborada por Márcia Auzier CRB 11/597

Gabriel Rodrigues Picanço

DATA CLEANER SERVICE - WEB SERVICE PARA APOIAR NA LIMPEZA DE DADOS

Trabalho de Conclusão de curso apresentado à banca examinadora do curso Superior de Tecnologia em Análise e Desenvolvimento de Sistema do Instituto Federal de educação, Ciência e Tecnologia do Amazonas – Campus Manaus Centro, como requisito para obtenção do título de tecnólogo.
Orientador: Esp. Rogério Luiz Araújo Carminé

Aprovado em _____ de _____ de 2023.

BANCA EXAMINADORA

Prof. Esp. ROGÉRIO LUIZ ARAÚJO CARMINÉ
Instituto Federal de Educação, Ciência e Tecnologia do Amazonas (IFAM)

Prof. Dr. RENILDO VIANA AZEVEDO
Instituto Federal de Educação, Ciência e Tecnologia do Amazonas (IFAM)

Prof. Me. VALCLIDES KID FERNANDES DOS SANTOS
Instituto Federal de Educação, Ciência e Tecnologia do Amazonas (IFAM)

MANAUS-AM
2023

RESUMO

É notório o grande aumento na quantidade de dados e de informação nos últimos anos. Dados errados geram informações erradas, que podem impactar significativamente na gestão das organizações e na vida das pessoas. As organizações precisam obter informação confiável e em tempo oportuno para a tomada de decisão efetiva nos negócios e, para isso, é importante investir em soluções que assegurem a qualidade de dados. As técnicas de limpeza de dados (em inglês, *Data Cleaning*) possibilitam identificar e corrigir valores não conformes baseados em regras e ações definidas. Essas técnicas são muito usadas nas atividades de preparação de dados nos processos de BI (*Business Intelligence*) e *Data Science* para contribuir com a qualidade dos resultados. Existem diversas ferramentas no mercado para limpeza de dados (p.ex., Excel, OpenRefine e Data Wrangler), que possuem características e funções específicas, e normalmente possibilitam a criação de *scripts* de programação para realizar as tarefas de limpeza. Esses *scripts* podem requerer bastante tempo e esforço para serem elaborados, e ser de difícil reutilização em ferramentas diferentes. Foi desenvolvida uma solução baseada em um serviço web (Data Cleaner Service) capaz de se integrar a ferramentas (p.ex., Aplicações Web) e realizar a limpeza de dados por meio da reutilização de *scripts* (p.ex., Python). Para demonstração da solução, foram desenvolvidas aplicações utilizando *scripts* com a ferramenta PANDAS, um componente e uma aplicação Web que consome o serviço. Com essa solução, espera-se contribuir positivamente na realização de tarefas de limpeza de dados em diversas áreas (p.ex., finanças, vendas e saúde), reduzindo esforço e tempo nessas atividades, promovendo a troca de experiência entre usuários e desenvolvedores, e impactando na geração de informação efetiva para tomada de decisão.

Palavras-chave: *limpeza de dados, serviço web, padronização de dados, Python.*

ABSTRACT

The large increase in the amount of data and information recently is notorious. Wrong data generates wrong information, which can significantly impact the management of organizations and people's lives. Organizations need to obtain reliable and timely information for effective business decision-making and, for that, it is important to invest in solutions that ensure data quality. Data Cleaning techniques make it possible to identify and correct non-compliant values based on defined rules and actions. These techniques are widely used in data preparation activities in BI (Business Intelligence) and Data Science processes to contribute to the quality of results. There are several tools on the market for data cleaning (e.g., Excel, OpenRefine and Data Wrangler), which have specific characteristics and functions, and normally allow the creation of programming scripts to carry out cleaning tasks. These scripts can take a lot of time and effort to write, and can be difficult to reuse across different tools. A solution based on a web service (Data Cleaner Service) capable of integrating with tools (e.g., Web Applications) and cleaning data through the reuse of scripts (e.g., Python) was developed. To demonstrate the solution, applications were developed using scripts with the PANDAS tool and a Web application that consumes the service. With this solution, it is expected to make a positive contribution in carrying out data cleaning tasks in several areas (e.g., finance, sales and health), reducing effort and time in these activities, promoting the exchange of experience between users and developers, and impacting the generation of effective information for decision-making.

Keywords: data cleaning, web service, data standardization, Python.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

Nesta lista, apresentamos algumas abreviações essenciais no campo da tecnologia e comunicação.

- URL: Uniform Resource Locator (Localizador Uniforme de Recursos).
- REST API: Representational State Transfer Application Programming Interface (Interface de Programação de Aplicativos de Transferência de Estado Representacional).
- HTTP: Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto).
- JSON: JavaScript Object Notation (Notação de Objeto JavaScript).
- CSV: Comma-Separated Values (Valores Separados por Vírgula).
- XML: Extensible Markup Language (Linguagem de Marcação Extensível).
- UC: Use Case (Casos de uso).

SUMÁRIO

1. INTRODUÇÃO

A sociedade lida com um grande volume de dados, que vem crescendo exponencialmente. Um estudo aponta que, no período de 2010 a 2020, houve um aumento de 5000% na quantidade de dados do planeta, ou seja, de 1,2 trilhões de gigabytes para 59 trilhões de gigabytes (PRESS, 2020). Muitos desses dados são utilizados para geração de informação para tomada de decisão, impactando nos resultados das organizações.

No contexto de um sistema da informação, a qualidade da informação tem um impacto significativo nesse sistema, pois a partir destas informações são tomadas as decisões (ABIB, 2010). A qualidade dos dados é muito importante para garantir uma informação confiável e embasar a tomada de decisões em uma organização.

Dados errados podem impactar significativamente na informação para tomada de decisão em diversas áreas (p. ex. medicina, economia, *marketing*). Por exemplo, um médico, após analisar um resultado errado de um exame, pode prescrever um medicamento inadequado para o paciente, ou um gestor pode decidir, de forma equivocada, ampliar o número de lojas de uma organização baseado no relatório errado recebido. Obter informações confiáveis e em tempo oportuno é um dos grandes desafios a serem vencidos e, para geração de informação confiável, é necessário um processo efetivo de coleta e processamento dos dados.

Os sistemas de informação informatizados favorecem maior qualidade dos dados coletados devido à validação no momento da coleta e, consequentemente, melhor qualidade da informação gerada, se comparado a sistemas baseados em processos manuais. Muitas organizações possuem sistemas informatizados de apoio à gestão, porém, isso não é realidade em muitas outras, que ainda lidam com papéis, planilhas eletrônicas e outros tipos de registro de dados cuja qualidade pode ser melhorada.

Para melhorar a qualidade dos dados coletados, pode-se aplicar técnicas de limpeza de dados. Essas técnicas consistem em identificar valores não conformes, com base em definições, ou regras, e depois realizar ações de correção para obter os dados preparados, ou "limpos", que serão utilizados para geração da informação. Em uma base de dados com muitos registros, o processo de limpeza realizado manualmente pode demandar muito tempo e esforço, além de ser mais propenso a

erros humanos. Por esse motivo, existem ferramentas informatizadas que apoiam nesse processo.

As ferramentas de limpeza possuem recursos específicos (rotinas de validação e correção de dados) para melhorar a qualidade dos dados, mas que, em muitos casos, são de difícil reutilização entre elas. Identificar maneiras de reutilizar recursos de uma ferramenta em outras pode gerar benefícios significativos para o processo de limpeza de dados nas organizações.

1.1. Problematização

Existem diversas ferramentas para limpeza de dados — como Excel, Linguagem R, Linguagem Python, OpenRefine, e Data Wrangler. Algumas delas (p.ex. Linguagens R e Python) exigem conhecimento em linguagem de programação para elaboração de *scripts* para realizar a limpeza de dados. Outras são mais simples e fáceis de usar, permitindo que o usuário realize as ações através de uma *interface* gráfica (p.ex. Excel).

As ferramentas que permitem a automação das tarefas de limpeza de dados possibilitam reduzir tempo e esforço da atividade em comparação ao método manual, que ainda pode ser mais propenso a erros. Normalmente, para realizar a automação dessas tarefas, são elaborados *scripts* utilizando linguagens de programação. As linguagens mais citadas quando o tema é ciência de dados, onde, normalmente, realizam-se atividades de preparação de dados, são as linguagens Python, R, Java e SQL, além das soluções MS-Excel e Google Planilhas (MARINO, 2018).

Problemas de qualidade de dados podem ocorrer em diversas áreas (p.ex. Financeira, Marketing e Saúde), que podem lidar com dados de mesmo domínio, por exemplo, dados de pessoas, de locais e de saúde. Pensar em formas de reutilizar scripts de ferramentas diferentes pode contribuir para redução de tempo e esforço, além de promover uma troca de experiência entre pessoas envolvidas no processo de limpeza de dados.

As ferramentas, ou aplicativos, para limpeza de dados, normalmente, aceitam linguagens específicas para elaboração dos scripts. Além de ambientes integrados de desenvolvimento (IDE, do inglês *Integrated Development Environment*) para as linguagens Python e R, é possível citar o aplicativo MS-Excel, que possibilita escrever

scripts na linguagem VBScript, e também o aplicativo OpenRefine que permite utilizar as linguagens: GREL (General Refine Expression Language), Python e Clojure.

A reutilização dos *scripts* de limpeza de dados entre ferramentas diferentes pode ser um grande desafio. Normalmente, quando se deseja reaproveitar um algoritmo de limpeza de dados desenvolvido em uma ferramenta diferente, é necessário reescrever o *script* na linguagem necessária para execução. Por exemplo, se for preciso reutilizar um script elaborado em Python no aplicativo Excel, é necessário reescrevê-lo na linguagem VBScript.

Neste trabalho, foi desenvolvida uma solução para apoiar a reutilização de *scripts* de limpeza de dados em plataformas diferentes. Foi considerado uma solução

1.2. Justificativa

Com o crescimento do volume de dados no mundo, ocorre também a preocupação com a qualidade de dados, pois dados errados refletem na informação gerada, que, por sua vez, impactam nas decisões tomadas. Informações erradas podem gerar impactos negativos nas organizações, na saúde das pessoas, nas políticas públicas, e muitas outras situações (WANG; ZIAD; LEE, 2006).

Para melhorar a qualidade dos dados, é importante investir em soluções de limpeza de dados. Existem diversas soluções de limpeza de dados no mercado, mas que ainda não possibilitam uma forma fácil e simples de reutilização de scripts desenvolvidos nelas.

Uma solução que permita a reutilização de scripts de limpeza de dados desenvolvidos em plataformas diferentes irá permitir o compartilhamento mais fácil e rápido por usuários de diversas áreas (p.ex., indústria, ensino, ciência) que passam por problemas de qualidade de dados semelhantes, contribuindo para diminuir o esforço e o tempo na realização dessas tarefas.

1.3. Objetivos

Este trabalho de conclusão de curso tem como objetivos, os descritos a seguir.

1.3.1. Objetivo Geral

Desenvolver um software que permita a reutilização de *scripts* de limpeza de dados entre plataformas diferentes através de um serviço web, contribuindo, assim, para o compartilhamento de soluções por profissionais que vivenciam problemas semelhantes de qualidade de dados.

1.3.2. Objetivos Específicos

- Apresentar uma comparação de soluções de limpeza de dados existentes no mercado.
- Permitir a criação de aplicações de limpeza de dados com base no serviço web.
- Formação de uma base de scripts de limpeza de dados para reutilização.

1.4. Metodologia

A seguir a metodologia deste trabalho é explicada, que direcionou as etapas executadas durante este projeto.

Estudos de aplicativos relacionados: consiste no estudo de aplicativos relacionados para analisar as abordagens mais comuns entre eles e que melhor se encaixe na solução do problema.

Concepção da aplicação: consiste no levantamento dos requisitos na definição dos elementos da solução.

Implementação: consiste no desenvolvimento da solução proposta, com suas especificações, modelagens, definições de arquitetura do sistema e tecnologias que serão utilizadas no decorrer do projeto.

Demonstração de Uso da Solução: consiste na construção de 2 aplicações de demonstração com base na solução desenvolvida.

Escrita do Projeto: Consiste na elaboração e formatação de um documento escrito de forma incremental durante toda a elaboração do projeto, finalizado e entregue com a defesa da monografia.

1.5. Organização do Documento

Este documento seguirá a estrutura mostrada a seguir:

A apresentação da revisão bibliográfica sobre o tema de limpeza de dados, as soluções de limpeza de dados relacionadas e as tecnologias utilizadas no desenvolvimento da solução.

Em seguida apresenta a descrição da solução, contendo os requisitos, a modelagem do software e os principais detalhes da implementação.

E por fim apresenta as considerações finais sobre o desenvolvimento da solução e os resultados obtidos.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém uma revisão sobre conceitos de ciência de dados e limpeza de dados importantes para o entendimento da solução proposta, e também uma descrição de ferramentas relacionadas (Microsoft Excel, R, Python, OpenRefine e Clarity) para identificar suas características e funcionalidades mais comuns.

2.1. Ciência de dados

Pode-se atribuir à ciência de dados a extração de informação útil a partir de imensas bases de dados complexas, dinâmicas, heterogêneas e distribuídas (Bugnion; Manivannan; Nicolas, 2017). Tem na interdisciplinaridade um papel fundamental na era digital atual tendo como base três domínios de conhecimento: Ciência da Computação; Estatística e Matemática; O domínio do conhecimento como mostrado por Conaway com o Data Science Venn Diagram.

O *Data Science*, ou Ciência de Dados, envolve princípios, processos e técnicas para compreender fenômenos por meio da análise (automatizada) de dados através da coleta, análise e interpretação de grandes volumes de dados, com o objetivo de extrair insights (ideias) valiosos e tomar decisões embasadas em evidências. A relevância da ciência de dados reside na sua capacidade de lidar com a complexidade e a variedade de informações disponíveis, possibilitando a compreensão de padrões, previsões de tendências e a resolução de problemas em diversos setores.

De acordo com Provost e Fawcett, a Tomada de Decisão Orientada por Dados (DOD) representa a prática de embasar as decisões na análise criteriosa dos dados,

em vez de depender exclusivamente da intuição. Nesse contexto, a Ciência de Dados surge como uma poderosa aliada na busca por insights e na melhoria contínua do processo decisório. A metodologia da Ciência de Dados oferece uma ampla gama de vantagens, incluindo a obtenção de informações contextualizadas, a revelação de fenômenos sutis ocultos nos dados e a validação ou refutação de hipóteses previamente estabelecidas. Sob essa perspectiva, a Tomada de Decisão Guiada por Dados, conceituada por Provost e Fawcett em 2013, ganha destaque como um processo que capacita a exploração plena do potencial da Ciência de Dados para embasar escolhas fundamentadas.

Uma das áreas em que a ciência de dados tem desempenhado um papel crucial é na saúde, especialmente em um contexto de avanços tecnológicos e descobertas de novos patógenos. Os profissionais de saúde estão coletando uma quantidade cada vez maior de dados, que abrangem desde registros eletrônicos de saúde dos pacientes até informações provenientes da decodificação genômica. Esses dados representam uma rica fonte de informações. Nesse contexto, a Ciência de Dados pode ser utilizada como ferramenta para sua análise em larga escala, proporcionando insights valiosos para o diagnóstico e tratamento de doenças, a descoberta de novos medicamentos e a identificação de padrões epidemiológicos. A análise das informações geradas em grande escala permite identificar padrões, correlações e tendências ocultas em uma análise manual contribuindo para a melhoria da precisão e eficácia dos diagnósticos. A consequência disso é que se pode realizar um tratamento mais personalizado e assertivo.

Além da saúde, a Ciência de Dados tem uma ampla aplicação em outros setores, tais como finanças, marketing, varejo e logística. Na área financeira, por exemplo, a Ciência de Dados desempenha um papel fundamental na identificação de fraudes, análise de riscos e tomada de decisões de investimento. Ela utiliza técnicas estatísticas e algoritmos para analisar grandes volumes de dados financeiros e identificar padrões suspeitos que possam indicar atividades fraudulentas. Além disso, a Ciência de Dados auxilia na avaliação e gerenciamento de riscos financeiros, permitindo que as instituições tomem decisões mais informadas sobre investimentos, empréstimos e seguros.

No campo do marketing, a Ciência de Dados desempenha um papel crucial na segmentação de clientes, personalização de campanhas e análise do comportamento

do consumidor. São utilizadas técnicas de análise de dados para identificar grupos de clientes com características semelhantes, permitindo que as empresas direcionem campanhas específicas. Além disso, ferramentas podem ser usadas para analisar o comportamento do consumidor, suas preferências, histórico de compras e interações com marcas, para fornecer *insights* que otimizam estratégias de marketing e melhoram a eficácia das campanhas.

No setor varejista, a Ciência de Dados é aplicada em áreas, como otimização de estoques, previsão de demanda e recomendação de produtos. Algoritmos de análise de dados são usados para analisar o histórico de vendas, padrões sazonais, dados demográficos e outros fatores relevantes, a fim de prever a demanda futura de produtos. Essas previsões ajudam na otimização de estoque, evitando escassez ou excesso de produtos.

Em relação à logística, a Ciência de Dados possui ferramentas de otimização de rotas, gestão de frota e previsão de demanda. Além de técnicas de otimização e análise de dados para identificar as rotas mais eficientes para a entrega de mercadorias, levando em consideração fatores como distância, tráfego e restrições de tempo. Dados históricos de demanda, sazonalidade e outros fatores relevantes são analisados por ferramentas de Ciência de Dados para prever a demanda futura e otimizar a gestão da frota e evitar atrasos.

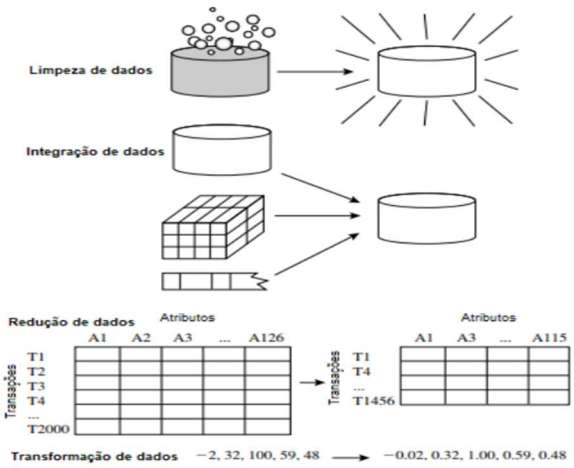
A relevância da Ciência de Dados consiste na sua capacidade de transformar dados brutos em conhecimentos úteis, auxiliando na tomada de decisões estratégicas e na criação de valor em diversas áreas. Através da análise de dados, é possível identificar oportunidades, mitigar riscos, melhorar a eficiência operacional e aprimorar a experiência do cliente. A ciência de dados proporciona uma visão mais precisa e baseada em evidências, permitindo que as organizações se tornem mais ágeis, competitivas e inovadoras em um mundo cada vez mais orientado por dados.

2.2. Limpeza de dados

A limpeza de dados visa detectar e remover anomalias dos dados para melhorar a sua qualidade (OLIVEIRA; RODRIGUES; RANGEL HENRIQUES, 2022). Logo, a limpeza de dados é uma operação realizada nos dados para adaptar formatos nas tuplas e valores, identificar restrições de integridade, derivar valores em falta a partir dos existentes, remover conflito entre tuplas, eliminar duplicidade, além de

detectar desvios, ou seja, valores com um grande potencial de serem inválidos (OLIVEIRA; RODRIGUES; RANGEL HENRIQUES, 2022).

FIGURA 01 - Etapas de pré-processamento



Fonte: HAN, 2011.

A Figura 01 pode ser dividida em três partes e cada uma dessa parte é a representação gráfica do que foi citado, assim, na parte que se refere a “Limpeza de dados” seria a parte de adaptar formatos nas tuplas e valores, identificar restrições de integridade, a segunda parte “Integração de dados” seria identificação de restrições de integridade, derivar valores em falta a partir dos existentes, remover conflito entre tuplas, a terceira e última parte “Redução de dados” representa a eliminar duplicidade e remover desvio.

2.3. Ferramentas de limpezas de dados

2.3.1. Excel

Microsoft Excel é uma ferramenta de propriedade da Microsoft que pode ser adquirida com o pacote Office. É um software com *interface* gráfica para manipulação das planilhas eletrônicas (“Microsoft Excel”, 2022). Estas auxiliam na gestão de dados e também na geração de informação em várias formas (p.ex., tabelas e gráficos). No que se refere a limpeza de dados, tem um recurso chamado “validação de dados”, que identifica os dados conforme um padrão definido, mas não realiza a correção.

2.3.2. Google planilhas

O Google planilhas (ou Google Sheets) é uma ferramenta disponibilizada gratuitamente com pacote de ferramentas para documentos do Google (Google Docs). Ela é compatível com a maioria das ferramentas de planilhas existentes, como o MS-Excel, auxiliando na gestão de dados e possibilitando a análise de grande quantidade de informações.

A ferramenta possui opções para limpeza de dados, sendo possível verificar os dados na coluna selecionada, inclusive, oferecendo sugestões de correções. É possível corrigir os dados com base nessas sugestões ou de forma individual, além da criação de scripts utilizando a linguagem Javascript para acessar os recursos da ferramenta (“Google Sheets API Overview”, 2022).

2.3.3. OpenRefine

OpenRefine é uma ferramenta de código aberto que permite trabalhar com dados padronizados, limpá-los, transformá-los de um formato para outro, além de disponibilizar um serviço web e dados externos (“OpenRefine”, 2012).

Nesta ferramenta há diversas funcionalidades de manipulação e limpeza de dados, mas, ao se falar em regras de validação e remoções de anomalias, são definidas de forma parecida com linguagem de programação, assim, gerando um grau de complexidade quando não se tem domínio.

2.3.4. Clarity

O Clarity, da TIBCO, é um software que integra e unifica sua massa de dados de forma confiável e entrega resultados em tempo real e em escalas. Oferece recursos como integração, descoberta de dados, remoção de dados duplicados, padronização do endereço e transformação de dados (“Introduction to TIBCO Clarity”, 2014). É um software com muitos recursos, e as funcionalidades pré-estabelecidas tornam a curva de aprendizado muito rápida. Sobre a forma de aquisição do produto, pode haver preocupações sobre o preço da ferramenta, que é elevado em comparação às outras, que possuem um menor preço ou são gratuitas.

2.3.5. Linguagem R para limpeza de dados

R é uma linguagem de programação de licença pública, fracamente tipada, voltada para manipulação e visualização de dados. A linguagem foi criada no ano de 1993 por Ross Ihaka e Robert Gentleman, e possuía a premissa de suprir a necessidade de uma linguagem de programação com uma ampla variedade de técnicas estatísticas e gráficas. Assim, o ambiente estatístico R uma boa ambiente para limpeza de dados reproduzíveis, pois todas as ações de limpeza podem ser roteirizadas e reproduzidas (DE JONGE, 2013).

2.3.6. Python para Limpeza de Dados

Python é uma linguagem de programação de alto nível, com ênfase na legibilidade do código, bastante utilizada para análise de dados. Esta linguagem tem diversas bibliotecas disponíveis para ajudar no desenvolvimento de aplicações, as mais conhecidas para análise de dados são Pandas, NumPy e MatplotLib (“Python Geral - documentação Python”, 2022).

2.3.7. Comparação entre ferramentas

A seguir, é apresentada uma comparação entre as ferramentas de limpeza de dados consideradas neste trabalho. Embora existam outras ferramentas disponíveis no mercado, as ferramentas a seguir foram selecionadas por estarem mais relacionadas com a proposta do trabalho, além da maioria delas serem muito utilizadas no mercado.

QUADRO 01: Comparação entre as Ferramentas de Limpeza de Dados

Critérios	MS-Excel	Google Planilhas	OpenRefine	Clarity
Definição de Scripts pelo Usuário	Sim	Sim	Sim	Sim
Linguagens de programação Aceitas	VBA	Javascript	GREL, Jython e Cloujure	GREL
Permite reutilizar funções de limpeza	Sim	Sim	Sim	Sim
Plataformas de Execução (Linux, Windows)	Windows	Ambas as plataformas	Ambas as plataformas	Ambas as plataformas
Permite acesso por REST API	Sim	Sim	Sim	Sim
Preço	R$ 359,00/ano	Grátis	Grátis	R$ 6.755,00/ano

Fonte: Próprio autor (2023)

Foi identificado que as ferramentas comparadas, além de possuírem funções próprias para limpeza de dados, permitem a definição de scripts de programação pelos usuários para estender suas funcionalidades, aceitando diversas linguagens de programação. Com relação à reutilização de código, elas permitem reutilizar os scripts através de ações de "copiar e colar" ou por um serviço web através de uma API padrão REST.

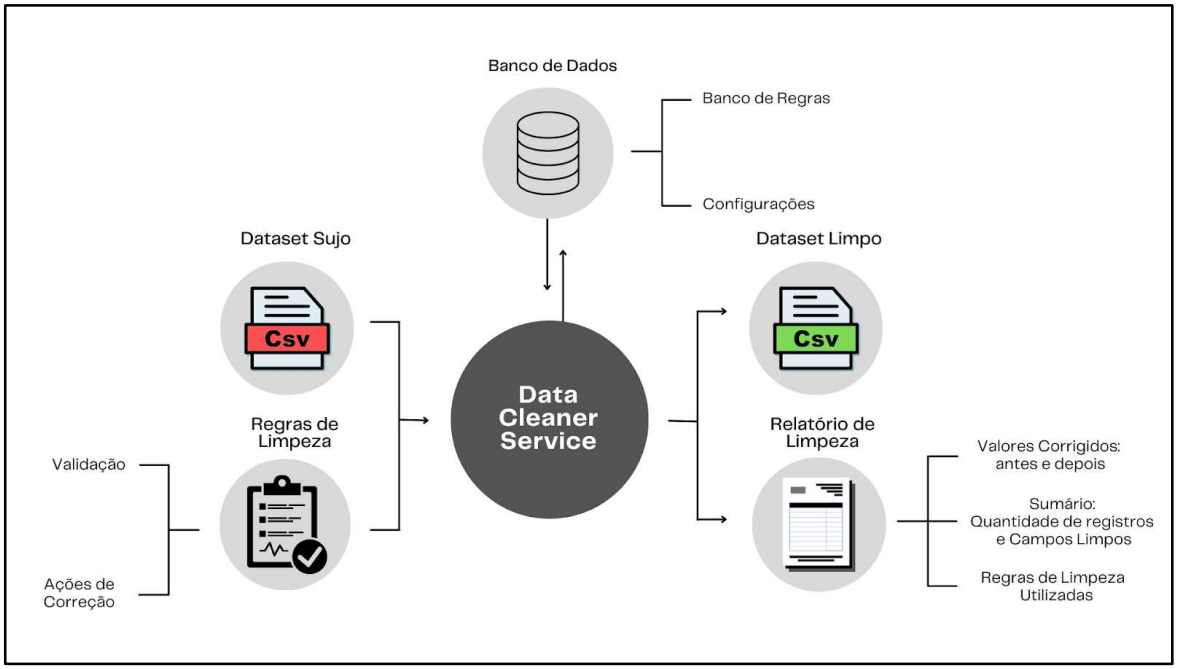
3. DESENVOLVIMENTO DA SOLUÇÃO

Esta seção descreve o desenvolvimento da solução para o problema descrito na introdução, suas principais características e funcionalidades, além de cenários de demonstração da solução.

3.1. Descrição da Solução

A solução consiste em um serviço web capaz de realizar a limpeza em um conjunto de dados (dataset) com base em regras e ações definidas. Conforme mostrado na Figura 02, o serviço recebe um dataset com problemas de qualidade de dados, ou "sujo", e também definições para limpeza (regras de validação e ações de correção) conforme a plataforma definida (Python, por exemplo); depois, ele realiza a validação e correção dos dados conforme as regras e ações; por fim, o serviço retorna um dataset corrigido, ou "limpo", e um relatório sobre o processo de limpeza, citando, por exemplo, os valores corrigidos e as ações executadas.

FIGURA 02 - Diagrama Conceitual da Solução



Fonte: Próprio autor (2023)

Um conjunto de dados, ou dataset, pode ser em vários formatos, por exemplo, CSV (*Comma-separated Values*), JSON (*Javascript Object Notation*), XML (*eXtensible Markup Language*) ou XLSX (formato do MS-Excel). Neste trabalho, por questões de simplicidade será considerado apenas o formato JSON.

Como exemplo de um *dataset* "sujo", ou com problemas de qualidade de dados, é mostrado, na Figura 03, um cadastro fictício de dados pessoais. É possível observar os valores não conformes em vermelho; por exemplo, na coluna "Altura", é possível identificar os valores "1.50" e "asd", que não passaram na validação para esse campo.

FIGURA 03 - Exemplo de Dataset com Problemas de Qualidade de Dados

ID	Nome	Sexo	Altura	Data Nascimento
1	Reginaldo Cosme Brito Guimarães	masculino	1.50	23-08-1990
2	J. Ronaldo Saraiva Portugal Paiva	M	asd	
3	Edena Coutinho Figueredo Barcelos	F	178	31/03/1985
		Feminino	123	07.07.1997
5	Calebe Monteiro Aguiar Garbelini	m	188	01/31/69
6	123123	123	164	abcd

Fonte: Próprio autor (2023)

Após o *dataset* "sujo" passar pelo serviço web (*Cleaner Service*), será disponibilizado um *dataset* "limpo". Na Figura 04, é possível ver um exemplo desse *dataset*, que está com os valores corrigidos em cor verde. É possível verificar que, na coluna "Altura", os valores que estavam antes "1.50" e "asd", na Figura 02, aparecem corrigidos na Figura 03, respectivamente, "150" e valor vazio.

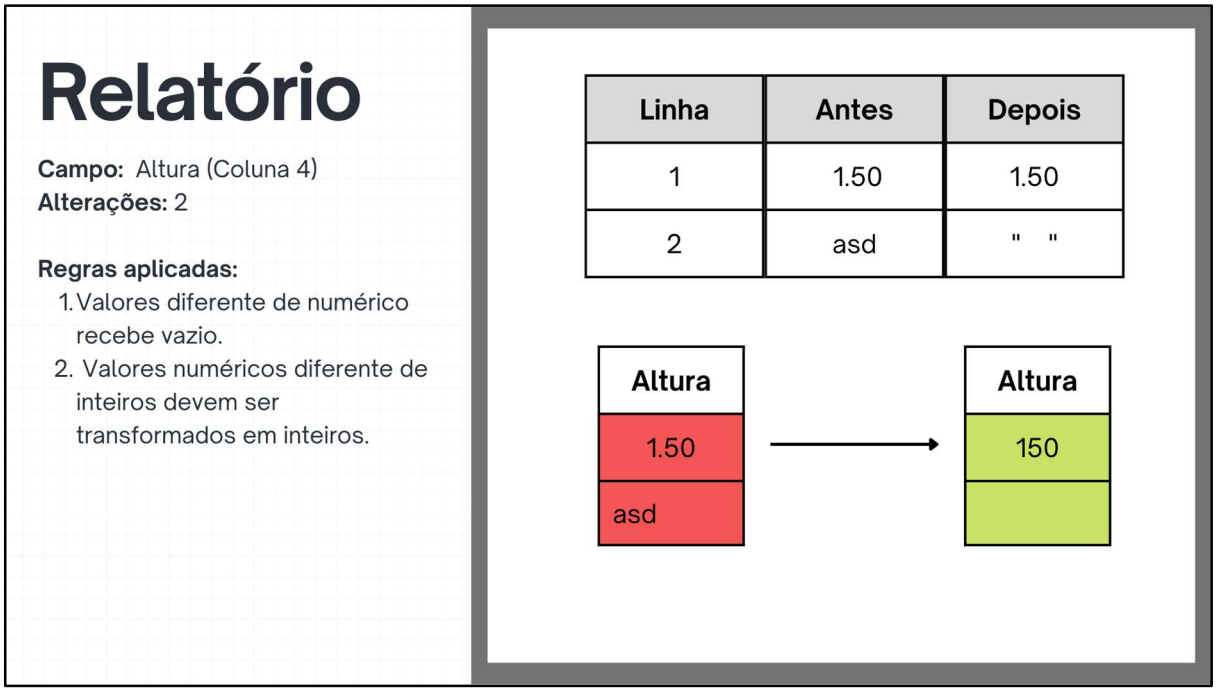
FIGURA 04 - Exemplo de Dataset "Limpo" no final do processo de limpeza com o uso da solução.

ID	Nome	Sexo	Altura	Data Nascimento
1	Reginaldo Cosme Brito Guimarães	M	150	23/08/1990
2	J. Ronaldo Saraiva Portugal Paiva	M		
3	Edesio Coutinho Figueredo Barcelos	F	178	31/03/1985
4	SEM NOME	F	123	07/07/1997
5	Calebe Monteiro Aguiar Garbelini	M	188	31/01/1969
6	SEM NOME	N	164	

Fonte: Próprio autor (2023)

No final do processo de limpeza de dados, também é mostrado um relatório sobre as ações executadas e os valores afetados. Na Figura 05, é mostrado um exemplo deste relatório, onde é possível verificar os valores corrigidos, um sumário com a quantidade de registros e campos que sofreram ação, além das regras de limpeza utilizadas.

FIGURA 05 - Modelo de Relatório de Limpeza de Dados



Fonte: Próprio autor (2023)

3.2. Tecnologias utilizadas na solução

3.2.1. REST API.

A tecnologia REST API é um conjunto de regras e convenções para comunicação entre sistemas distribuídos. Ela permite que diferentes aplicações se comuniquem por meio de requisições HTTP, utilizando os métodos padrão GET, POST, PUT e DELETE. Ao utilizar REST API, pode facilitar a integração e a troca de dados entre diferentes componentes de um sistema, permitindo que sejam desenvolvidas interfaces de comunicação flexíveis e escaláveis.

3.2.2. Linguagem Python

Python é uma linguagem de programação versátil e de alto nível, amplamente utilizada no desenvolvimento de aplicações web, científicas e de automação. Ela possui uma sintaxe simples e clara, o que a torna fácil de aprender e ler. Python oferece uma ampla variedade de bibliotecas e frameworks, ajudando no desenvolvimento de soluções robustas e eficientes. Sua popularidade crescente também se deve à sua comunidade ativa e suporte contínuo, tornando-a uma escolha sólida para o desenvolvimento de diversas aplicações.

“Python tem sido o vencedor do prêmio anual do índice TIOBE por 3 vezes nos últimos 5 anos. Ele cresceu em popularidade de forma impressionante, devido aos avanços nos campos de ciência de dados e inteligência artificial. O aumento começou em algum momento do outono de 2017, com uma participação de 3% e terminou no final do ano passado com uma participação de 17%.” (junho de 2023, Fonte: TIOBE - Índice TIOBE, disponível em: https://www.tiobe.com/tiobe-index/)

3.2.3. Django Rest Framework

O Django REST Framework é uma biblioteca flexível para o desenvolvimento de APIs RESTful em Python. Ele fornece um conjunto abrangente de ferramentas e funcionalidades para criar APIs de alta qualidade e bem estruturadas. Com o Django REST Framework, pode facilmente definir modelos de dados, rotear URLs, serializar e desserilizar dados, implementar autenticação e autorização, além de fornecer recursos adicionais, como paginação e filtragem. O framework simplifica o desenvolvimento de APIs seguras e escaláveis em Python.

3.2.4. Pandas

Pandas é uma biblioteca de análise de dados em Python que oferece estruturas de dados flexíveis e eficientes para manipulação e análise de conjuntos de dados. Com o Pandas, você pode facilmente carregar, limpar, transformar e analisar dados estruturados, como tabelas e planilhas. Ele fornece muitas funcionalidades para filtrar, agrupar, ordenar e calcular estatísticas sobre os dados, permitindo que você extraia insights valiosos para sua solução.

3.2.5. PostgreSQL

PostgreSQL, também conhecido como Postgres, é um sistema de gerenciamento de banco de dados relacional de código aberto. Ele oferece recursos avançados de armazenamento, consultas e manipulação de dados, garantindo alta confiabilidade e desempenho. O Postgres é conhecido por sua estabilidade, escalabilidade e conformidade com padrões SQL, tornando-o uma escolha popular em diversas aplicações.

3.2.6. Flask

O Flask é um framework web em Python conhecido por sua simplicidade e facilidade de uso. Ele permite aos desenvolvedores criar aplicações web de forma rápida e eficiente, fornecendo uma estrutura flexível e modular para lidar com requisições HTTP. O Flask suporta extensões que adicionam funcionalidades extras, como integração com bancos de dados e validação de formulários. Além disso, ele segue o padrão WSGI, podendo ser implantado em diferentes servidores web. Com o ecossistema e bibliotecas do Python, o Flask oferece uma ampla gama de recursos para o desenvolvimento de aplicações web, tornando-se uma escolha popular entre os desenvolvedores.

3.3. Requisitos de Software

Neste tópico, serão apresentados os requisitos de software. Serão discutidos os principais aspectos relacionados às funcionalidades, comportamentos e características que o software deve possuir para atender as necessidades dos usuários.

3.3.1. Requisitos Funcionais.

A seguir são apresentados os requisitos funcionais do sistema, os quais determinam o escopo das funcionalidades elaboradas durante a fase de implementação.

RF001 **-** Limpeza de Dados

O serviço deve fornecer funcionalidades para limpar e transformar os dados importados, após a definição de regras e ação de correção como:

- Remoção de duplicatas
- Tratamento de valores nulos
- Formatação de datas
- Correção de erros
- Normalização de dados

RF002 - Gerenciar Ação de correção.

O sistema deve permitir o cadastro ação de correção para correção dos dados inconsistentes. Cada ação de correção específica as ações a serem executadas, nos dados que não estiverem corretos ou de acordo. As ações podem incluir correção automática, substituição por um valor padrão, entre outras.

RF003 - Gerenciar Regras de Validação.

O sistema deve aplicar as regras de validação definidas aos conjuntos de dados importados. As regras devem ser aplicadas antes da execução das ações de correção. Os dados devem ser avaliados de acordo com as regras.

RF004- Aplicação de Ação de Correção

O sistema deve aplicar os modelos de regras e ações de correção definidos aos conjuntos de dados importados. As regras devem ser validadas antes da aplicação das ações de correção. As ações de correção devem ser aplicadas de acordo com as configurações definidas nos conjuntos de dados. Os dados devem ser avaliados e corrigidos de acordo com as regras e ações especificadas. O sistema deve registrar as correções realizadas nos conjuntos de dados.

RF005- Gerenciamento de Modelos

O sistema deve fornecer funcionalidades para criação, edição e exclusão de modelos de regras e ação de correção. Os modelos de regras definem as regras de validação a serem aplicadas aos conjuntos de dados. Os modelos de Ação de correção especificam as etapas e procedimentos para a correção dos dados, conforme definidos. O gerenciamento das regras e ações é essencial para personalizar e adaptar a limpeza de dados de acordo com os requisitos específicos de cada conjunto de dados.

3.3.2. Requisitos Não Funcionais

A seguir são apresentados os requisitos não funcionais do sistema, os quais determinam o escopo das funcionalidades elaboradas durante a fase de implementação.

RNF001 - Escalabilidade:

- O serviço deve ser escalável, permitindo lidar com um aumento no volume de dados processados, sem comprometer o desempenho ou a disponibilidade.

RNF002 - Desempenho:

- O serviço deve ter um desempenho satisfatório, garantindo tempos de resposta não superiores a 5 segundos para as requisições dos clientes.

RNF003 - Integração:

- O serviço deve ser projetado para permitir integração fácil e flexível com outros sistemas e serviços, seguindo modelo arquitetural REST de serviços WEB.

RNF004 - Usabilidade:

- A API deve ser intuitiva e fácil de usar para os desenvolvedores e clientes que a utilizam, considerando os elementos de maturidade dos serviços web.

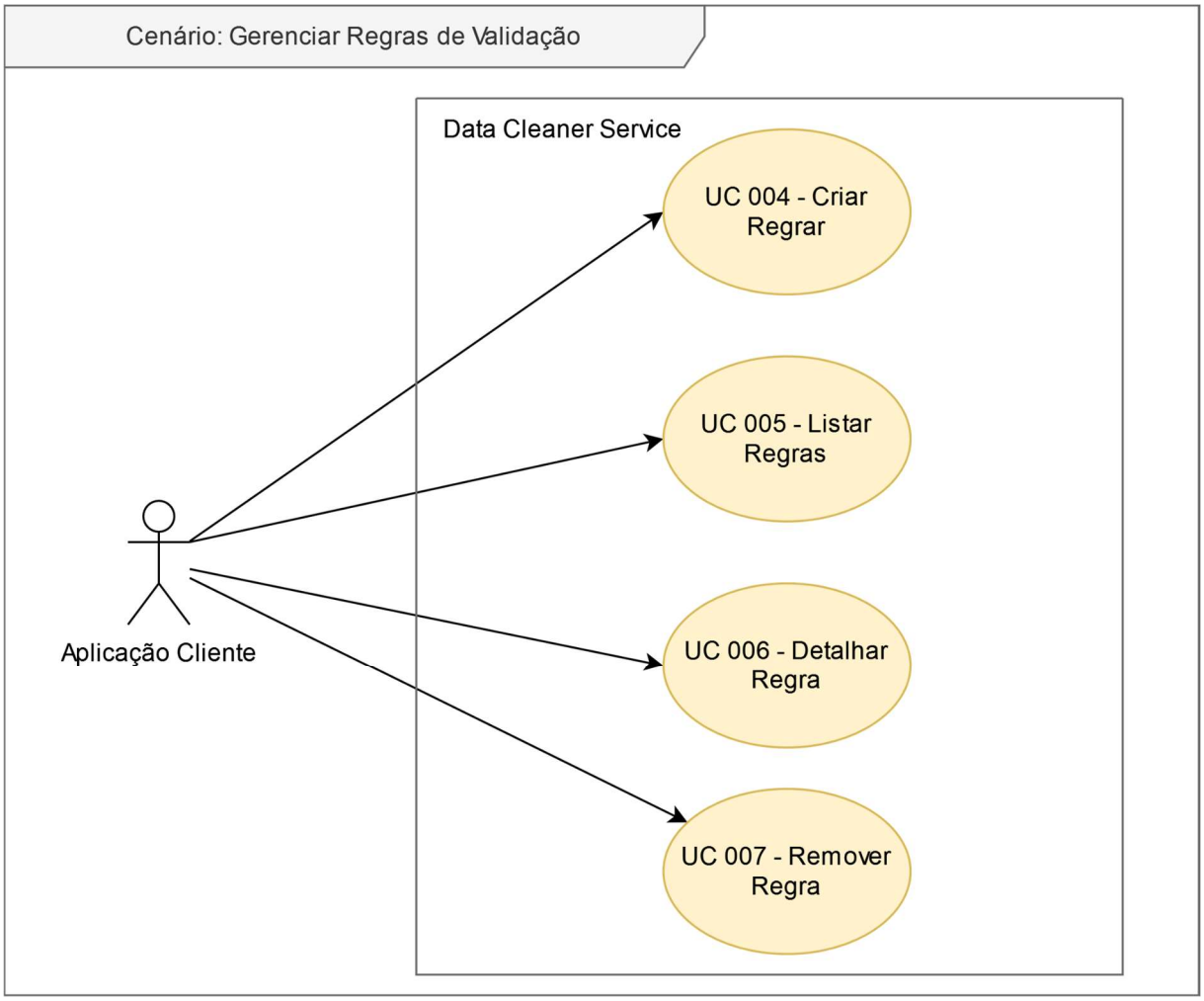
RNF005 - Manutenibilidade:

- O código do serviço web deve ser bem estruturado, modular e de fácil manutenção.

3.4. Modelagem de Software

3.4.1. Casos de Uso

FIGURA 06 - Gerenciar Regras de Validação.



Fonte: Próprio autor (2023).

UC 004 - Aplicação Cliente cria uma nova Regra. Para realizar essa ação, é necessário enviar uma requisição POST para a URL: "http://127.0.0.1:8000/regras/" com um formato JSON contendo os seguintes campos obrigatórios:

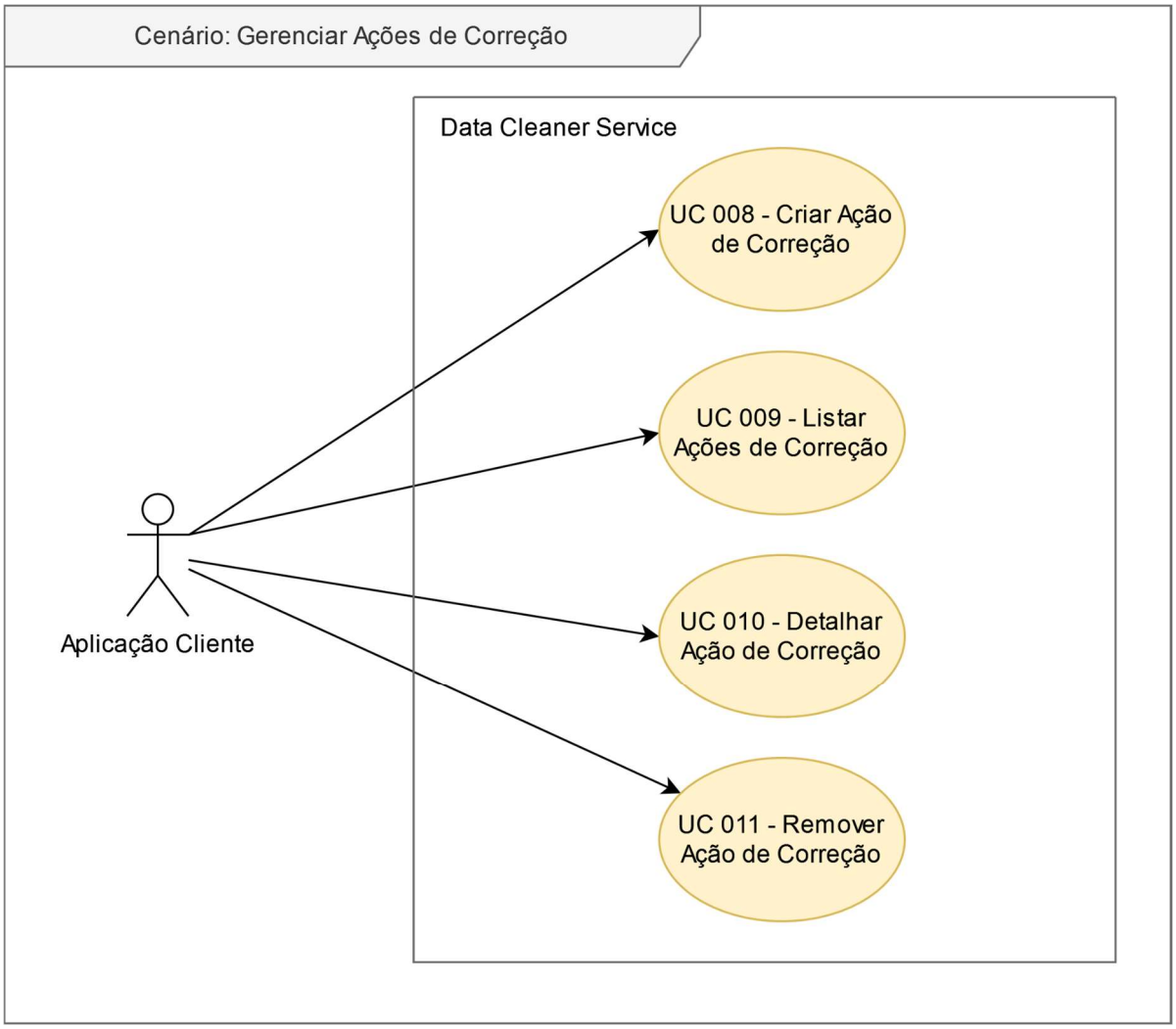
- "codigo": campo que representa o código da regra;
- "titulo": campo que descreve o título da regra;
- "descricao": campo que descreve a finalidade ou detalhes da regra;
- "tipo": campo que indica o tipo da regra (podendo ser nulo);
- "script": campo que contém o script ou a lógica da regra.

UC 005 - Aplicação Cliente visualiza a listagem de todas as regras cadastradas no banco de dados. Para acessar essa listagem, basta realizar uma requisição GET para a URL: "http://127.0.0.1:8000/regras/".

UC 006 - Aplicação Cliente visualiza detalhadamente uma regra específica, incluindo todos os seus atributos. Para obter esse detalhamento, é necessário realizar uma requisição GET para a URL correspondente à regra desejada, por exemplo: "http://127.0.0.1:8000/regras/1/".

UC 007 - Aplicação Cliente remove uma regra. Para isso, é necessário enviar uma requisição DELETE com a URL da regra que deseja excluir. Isso irá eliminar a regra do sistema.

FIGURA 07 - Gerenciar Ações de Correções



Fonte: Próprio autor (2023).

UC 008 - Aplicação Cliente cria uma nova Ação de Correção. Para executar essa ação, é necessário enviar uma requisição POST para a URL: "http://127.0.0.1:8000/acaodecorrecao/" com um formato JSON contendo os seguintes campos obrigatórios:

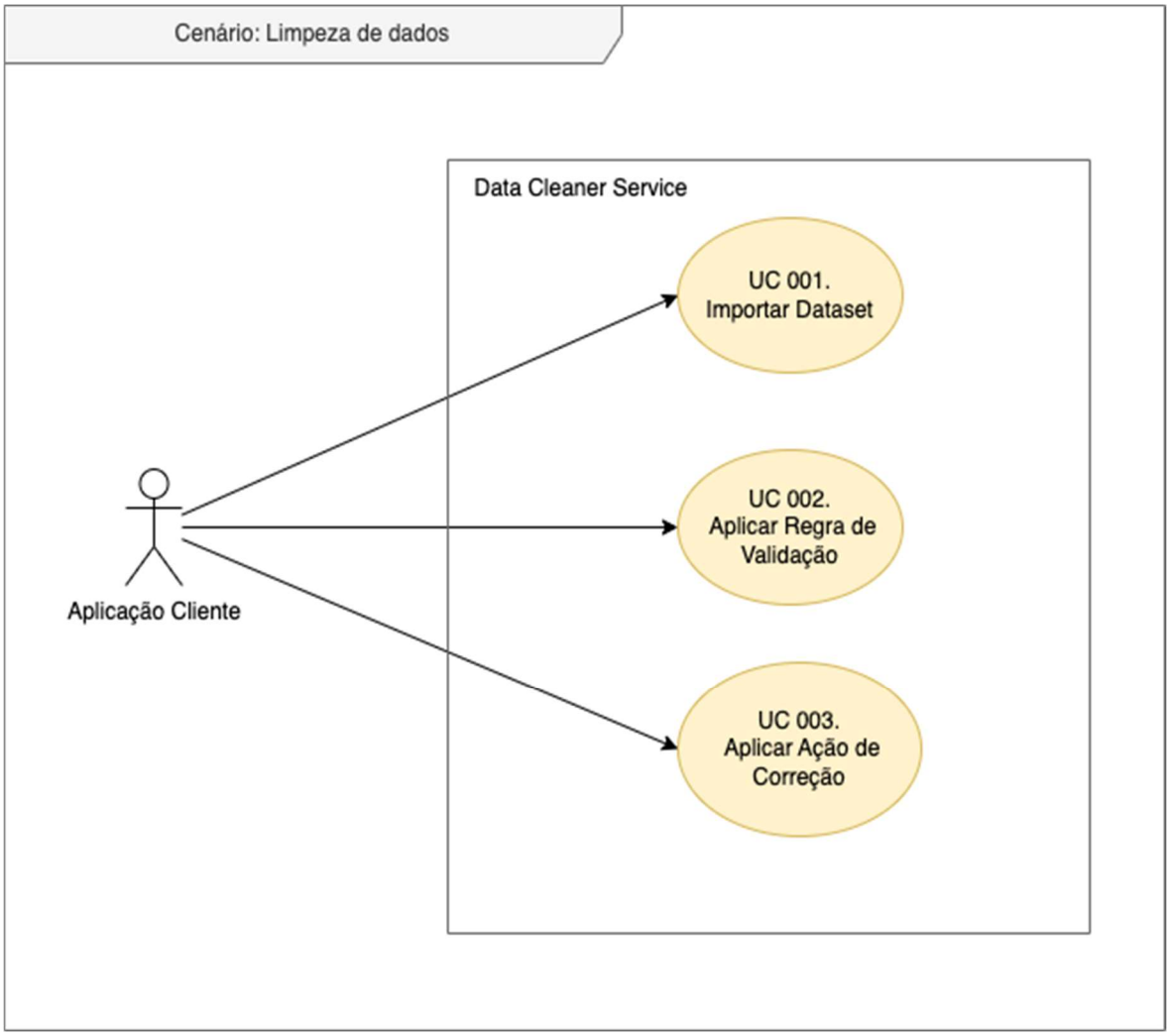
- "codigo": campo que representa o código da ação de correção;
- "titulo": campo que descreve o título da ação de correção;
- "descricao": campo que descreve a finalidade ou detalhes da ação de correção;
- "tipo": campo que indica o tipo da ação de correção (podendo ser nulo);
- "script": campo que contém o script ou a lógica da ação de correção.

UC 009 - Aplicação Cliente visualiza a listagem de todas as Ações de Correção cadastradas no banco de dados. Para acessar essa listagem, basta realizar uma requisição GET para a URL: "http://127.0.0.1:8000/acaodecorrecao/".

UC 010 - Aplicação Cliente visualiza detalhadamente uma Ação de Correção específica, incluindo todos os seus atributos. Para obter esse detalhamento, é necessário realizar uma requisição GET para a URL correspondente à Ação de Correção desejada, por exemplo: "http://127.0.0.1:8000/acaodecorrecao/1/".

UC 011 - Aplicação Cliente remove uma Ação de Correção. Para isso, basta enviar uma requisição DELETE com a URL da Ação de Correção que deseja excluir. Isso removerá a Ação de Correção do sistema.

FIGURA 08 - Limpeza de dados



Fonte: Próprio autor (2023).

UC 001 - Importar dataset: Neste caso de uso, o ator "Aplicação Cliente" utiliza a funcionalidade de importação para enviar os dados do dataset por meio de uma requisição POST. Além dos dados, as regras e ações de correção também são enviadas. A aplicação processa esses dados, aplicando as regras de validação e executando as ações de correção nos dados do dataset. Após esse processamento, os dados limpos são retornados como resultado da importação.

UC 002 - Aplicação das regras: Neste caso de uso, o Data Cleaner Service realiza a aplicação das regras de acordo com os dados importados. O serviço busca no banco de dados as regras especificadas no dataset e executa o script específico para validar os dados. Com base nas regras, o Data Cleaner Service determina se cada campo é considerado correto ou errado. Esse processo de validação identifica

possíveis inconsistências nos dados e fornece informações sobre sua qualidade e conformidade com as regras estabelecidas.

UC 003 - Aplicar Ação de Correção: Neste caso de uso, o Data Cleaner Service realiza a aplicação das ações de correção nos dados considerados não conformes. O serviço identifica os campos com valores incorretos e executa o script das ações de correção especificadas no dataset para corrigir esses dados. As ações de correção podem incluir transformações, substituições ou outras operações necessárias para ajustar os valores dos campos não conformes. Após a aplicação das ações de correção, os dados são atualizados e considerados corrigidos.

3.4.2. Modelagem de Dados

3.4.2.1. Diagrama de Classe

O diagrama de classes representa a estrutura das classes e suas relações no sistema, fornecendo uma visão geral das entidades envolvidas e como elas se relacionam entre si.

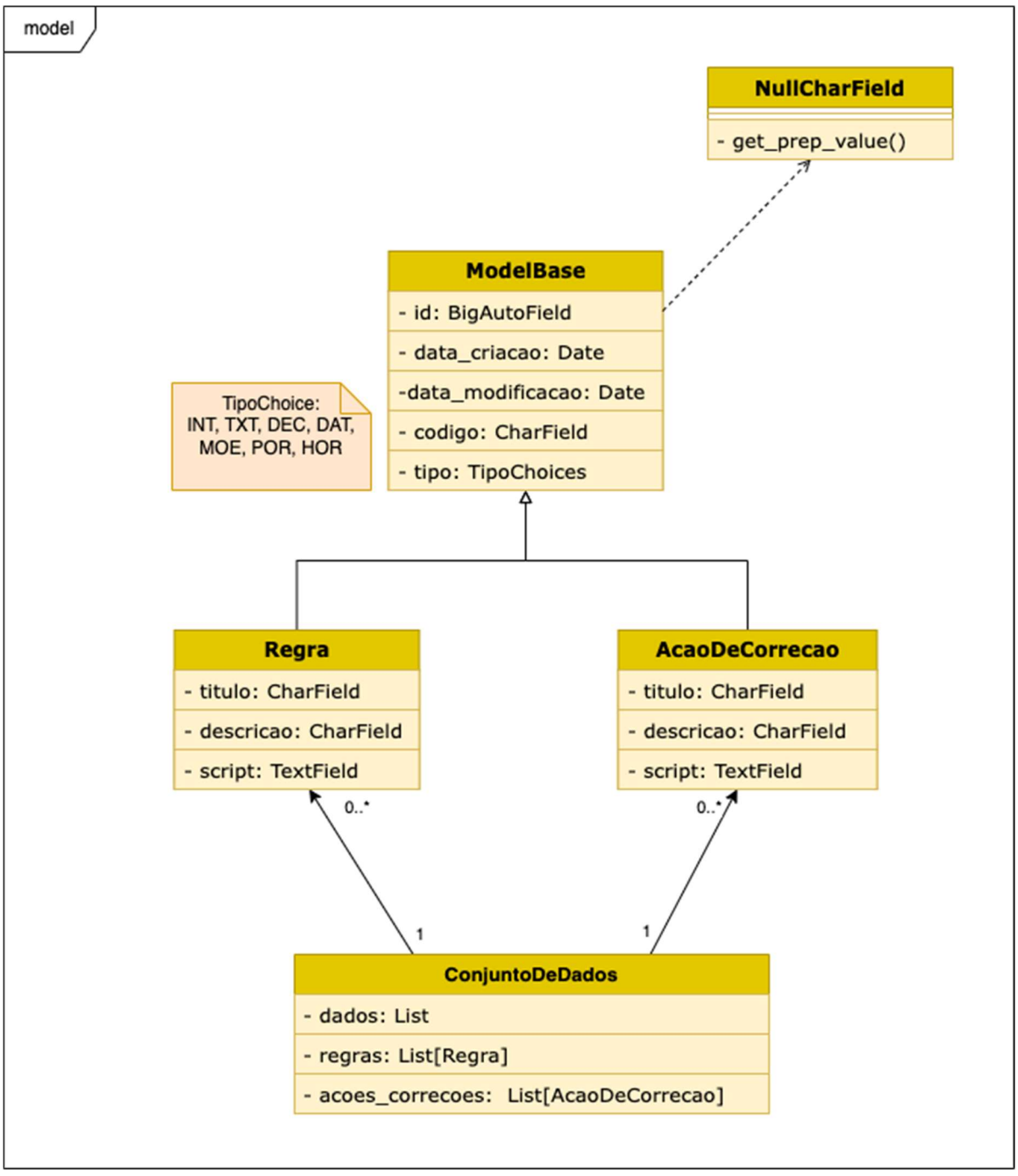
A classe "NullCharfield" é uma subclasse de "models.CharField" no Django que foi personalizada para lidar com campos de caracteres que podem ter valores vazios. No Django, quando um campo de caractere é definido como obrigatório (“null= False” e “black=False”) uma *string* vazia (“ ‘ ’ ”) não é considerada um valor válido. No entanto, em certos casos, pode ser útil permitir que um campo de caractere seja vazio, tratando-o como “None” no banco de dados.

A classe ModelBase é uma classe abstrata que funciona como uma base para todas as outras classes do sistema. Ela define os atributos comuns a todos os modelos, como "id", "data_criacao", "data_modificacao" e "codigo".

A classe Regra herda da classe ModelBase e representa uma regra no sistema. Ela possui atributos específicos, como "titulo", "descricao", “tipo” e "script", que descrevem a regra em detalhes.

Similarmente, a classe AcaoDeCorrecao também herda da classe ModelBase e representa uma ação de correção no sistema. Ela possui atributos como "titulo", "descricao", “tipo” e "script", que definem a ação de correção a serem executadas após a validação das regras.

FIGURA 09 - Diagrama de classes



Fonte: Próprio autor (2023)

Por fim, a classe ConjuntoDeDados tem uma associação com as classes Regra e AcaoDeCorrecao. Essa classe representa um conjunto de dados e é usada para representar os dados, regras e ações de correção. Ela permite que os dados sejam

associados às regras e ações de correção relevantes para um processamento conjunto.

Dessa forma, o diagrama de classes mostra a estrutura do sistema, com uma classe base compartilhada, classes específicas para regras e ações de correção, e uma classe que organiza e relaciona conjuntos de dados, regras e ações de correção. Isso proporciona uma estrutura coesa e modular para o sistema, facilitando a compreensão e a manutenção do código.

3.4.2.2. Diagrama de Banco de Dados

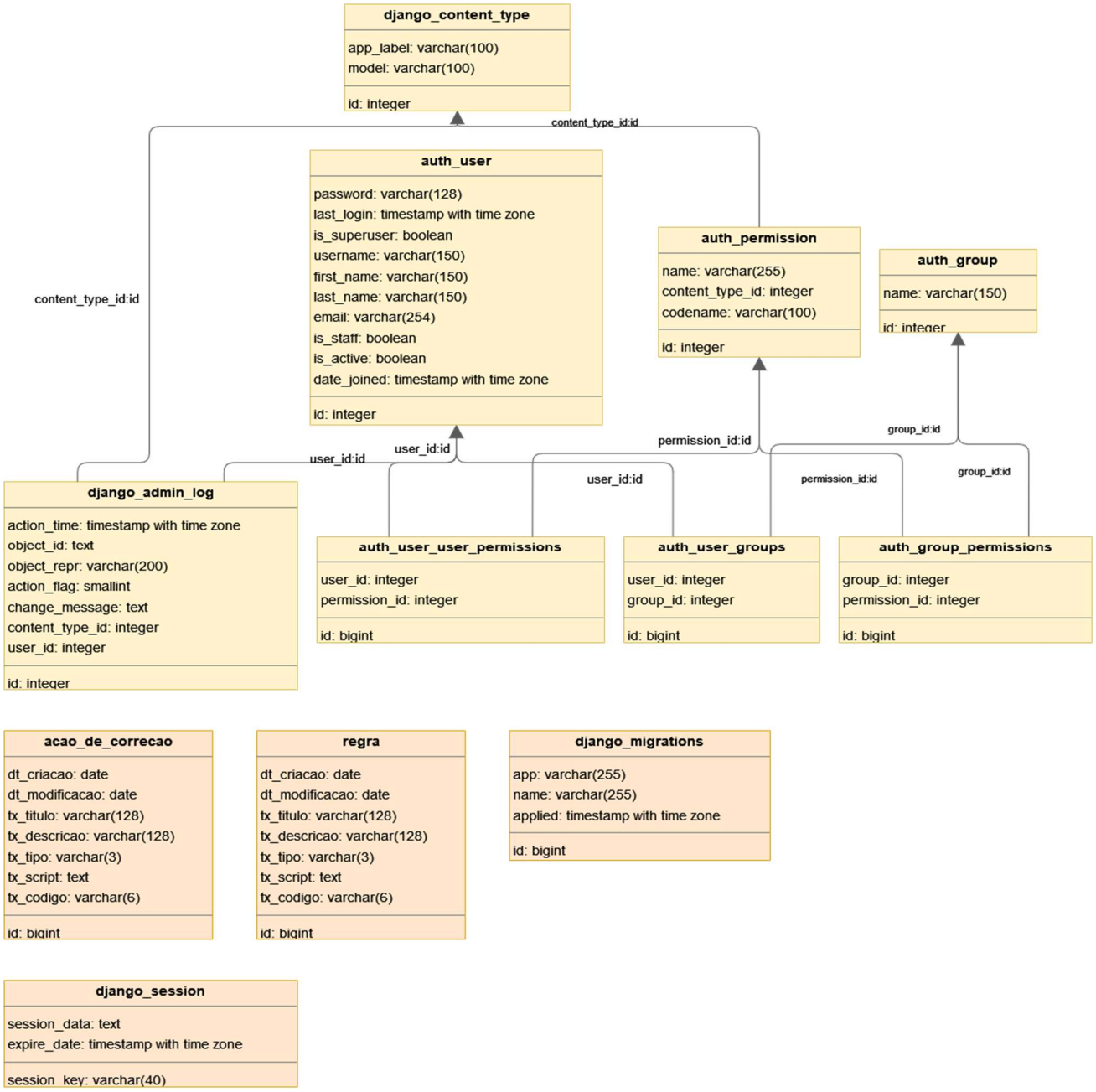
O diagrama de Banco de dados é uma representação visual onde mostra como as classes de entidades do sistema serão mapeadas e armazenadas no banco de dados relacional. Ele fornece uma visão clara das tabelas, relacionamentos e atributos que compõem o esquema de dados.

Ao utilizar o Django, algumas tabelas adicionais são criadas no banco de dados automaticamente. Essas tabelas são responsáveis pelo gerenciamento de recursos importantes, como autenticação de usuários, permissões e grupos.

No diagrama de classe, essas tabelas são representadas em cor amarela para indicar a diferença. Essas funcionalidades extras são criadas para fornecer funcionalidades essenciais de segurança e controle de acesso ao sistema. Elas permitem que os desenvolvedores implementem recursos como registro de usuário, autenticação de login, gerenciamento de permissões de acesso e organização de usuários em grupos.

Essa abordagem do Django de criar tabelas adicionais no banco de dados ajuda a simplificar e acelerar o desenvolvimento de aplicativos web, pois as funcionalidades básicas de autenticação e autorização são tratadas automaticamente pelo framework. Dessa forma, os desenvolvedores podem se concentrar em outras partes do sistema, sem precisar se preocupar com a implementação desses recursos complexos do zero.

FIGURA 10 - Diagrama do Banco de dados



Fonte: Próprio autor (2023).

As tabelas acao_de_correcao e regra são definidas para representar as entidades específicas do sistema, elas possuem campos correspondentes aos atributos dos modelos ‘AcaoDeCorrecao’ e ‘Regra’. Os campos representam as propriedades dessas entidades, como data de criação, data de modificação, título, descrição, tipo, script e código.

Em resumo, no diagrama de banco de dados, as tabelas destacadas em amarelo representam as tabelas geradas automaticamente pelo Django para recursos

internos, como migrações e sessões de usuário. Além disso, temos as tabelas personalizadas que correspondem aos modelos acao_de_correcao e regra, que armazenam os dados das ações de correção e regras do sistema.

3.5. Implementação do Serviço Web

A implementação é a etapa crucial do desenvolvimento deste trabalho, permitindo a disponibilização de funcionalidades e recursos acessíveis para a utilização do serviço. Neste tópico abordaremos as principais etapas e componentes relacionados à implementação do serviço web.

3.5.1. Cadastrar Regra de Validação

As regras de validação são adicionadas ao sistema seguindo alguns requisitos, de ser feito uma requisição POST para a URL: http://localhost:8000/regras/, e montar um JSON com os seguintes parâmetros obrigatórios:

- Código, é um campo texto de no máximo 5 caracteres
- Título, é o campo texto de no máximo 128 caracteres
- Descrição, é o campo texto de no máximo 128 caracteres
- Tipo, é o campo texto de no máximo 3 caracteres
- Script é um campo de texto sem limite de caracteres, algumas regras devem ser definidas, o script só aceita operadores lógicos, laço de petições, não aceita definições de novas funções com ‘*def*’ ou ‘*return’*. As variáveis definidas são *value* e *elemento*.

3.5.2. Cadastrar Ação de Correção

As ações de correção são adicionadas ao sistema seguindo alguns requisitos, fazer uma requisição POST para a URL: (http://localhost:8000/acaodecorrecao/), e montar um objeto no formato JSON com os seguintes parâmetros obrigatórios

- Código, é um campo texto de no máximo 5 caracteres
- Título, é o campo texto de no máximo 128 caracteres
- Descrição, é o campo texto de no máximo 128 caracteres
- Tipo, é o campo texto de no máximo 3 caracteres

- Script é um campo de texto sem limite de caracteres. Algumas regras devem ser definidas, o script só aceita operadores lógicos, laço de repetição, não aceita definições de novas funções com ‘*def*’ ou ‘*return’*. As variáveis definidas são *value* e *elemento*.

3.5.3. Aplicar Regra de Validação e Ação de Correção

Após a definição das regras de validação, é necessário fazer uma requisição POST, na qual os dados enviados devem conter a URL das regras, URL das Ações de correção e os dados. Abaixo um exemplo de JSON.

```
{
        “regras”: “http://127.0.0.1:8000/regras/1/”,
        “acao_correcao”: http://127.0.0.1:8000/acaodecorrecao/1/,
        “dados”: [0, 1, 2, 3, 4, 5, 6]
}
```

Caso as regras não sejam enviadas, considera-se que todos os dados estão em não conformidade com o padrão, e em seguida, é realizada a ação de correção. As regras enviadas no JSON são recuperadas do banco de dados, montando e executando o script para validar os dados.

Aqui está o código como referência para explicar como é feita a aplicação das regras e ação de correção:

FIGURA 11 - Classe Conjunto de Dados do Data Cleaner Service

```python
class ConjuntoDeDados(viewsets.ModelViewSet):
    serializer_class = serializer.ConjuntoDeDadosSerializer
    permission_classes = [permissions.AllowAny]

    def list(self, request, *args, **kwargs):...

    def retrieve(self, request, *args, **kwargs):...

    def update(self, request, *args, **kwargs):...

    def partial_update(self, request, *args, **kwargs):...

    def destroy(self, request, *args, **kwargs):...

    def create(self, request, *args, **kwargs):
        dados = request.data.get('dados', None)
        regras_urls = request.data.get('regras', None)
        acao_correcoes_urls = request.data.get('acao_correcao', None)

        # Verifica se o campo 'dados' está presente nos dados recebidos
        if dados is None:
            return Response({'error': 'O campo "dados" é obrigatório.'}, status=status.HTTP_400_BAD_REQUEST)

        regras = self.get_related_objects(self, urls=regras_urls, model=Regra)
        acao_correcoes = self.get_related_objects(self, urls=acao_correcoes_urls, model=AcaoDeCorrecao)

        conjunto_de_dados = models.ConjuntoDeDados(
            dados=dados,
            regras=regras,
            acoes_correcoes=acao_correcoes
        )

        behavior = bv.ExecutarLimpeza()
        resultado = behavior.run(conjunto_de_dados)
        return Response(resultado)
```

Fonte: Próprio autor (2023)

O código acima é uma visão geral do método *create* em uma classe de visão do Django. Ele extrai os dados, as URLs das regras e das ações de correção da requisição POST. Em seguida, obtém as regras e as ações de correção relacionadas com base nas URLs fornecidas.

O objeto “conjunto_de_dados” é criado com os dados, regras e ações de correção obtidas. Em seguida, é instanciada a classe “ExecutarLimpeza” do módulo “behavior.py”.

A função “run” na classe “ExecutarLimpeza” executa as etapas de aplicação das regras e ação de correção nos dados. Primeiro, as regras são aplicadas aos dados usando o método privado “__aplica_regra”, que percorre as regras e executa os

scripts de validação. Os dados que passarem nas regras são considerados corretos, enquanto os que falharem são considerados errados.

Se houverem ações de correção definidas, o método privado “__aplica_acao_de_correcao” é chamado para aplicar essas ações nos dados errados. O método percorre as ações de correção e executa os scripts correspondentes para cada dado errado. Os dados corrigidos são adicionados à lista de dados corrigidos, e os não corrigidos permanecem na lista de dados errados.

No final, o resultado contendo as listas de dados corretos, dados errados, dados corrigidos e dados não corrigidos é retornado como resposta.

3.6. Disponibilização do Código-fonte do Projeto

O código-fonte do projeto está disponível no seguinte repositório: Github: https://github.com/grpicanco/DataCleanerService

O código fonte da demonstração desenvolvida está disponível no seguinte repositório:

Github: https://github.com/grpicanco/Componente_DataCleanerService

3.7. Demonstração de Uso da Solução

Foram criados 2 exemplos para demonstrar o uso do serviço web Data Cleaner. O primeiro é o uso de uma biblioteca que facilita o uso do serviço em um script de limpeza de dados, e o segundo é o uso com aplicação frontend, onde é possível visualizar por meio de uma interface gráfica do usuário.

Esses exemplos têm o objetivo de mostrar as diferentes formas de utilizar o serviço Data Cleaner, destacando sua versatilidade e utilidade tanto em ambientes de *backend* quanto de *frontend*.

3.7.1. Biblioteca em Python para Uso em Scripts

O código-fonte abaixo representa um método "corrigir_dados" do componente que recebe os dados a serem corrigidos, uma lista de códigos de regras a serem aplicadas e uma lista de códigos de ações de correção a serem aplicadas.

FIGURA 12 - Componente Método Corrigir Dados.

```python
def corrigir_dados(self, dados, cod_regras, cod_acao_de_correcao):
    """
    Corrige os dados com base nas regras e ações de correção fornecidas.

    :param dados: Dados a serem corrigidos.
    :param cod_regras: Lista de códigos das regras a serem aplicadas.
    :param cod_acao_de_correcao: Lista de códigos das ações de correção a serem aplicadas.
    :return: Resposta da requisição.
    :raises Exception: Erro na requisição.
    """
    regras = []
    acaos = []
    url = 'http://localhost:8000/correcaodedados/'
    headers = {'Content-Type': 'application/json'}
    for item in cod_regras:
        regra = self.detalhar_regra(item)
        regras.append(regra.url)

    for item in cod_acao_de_correcao:
        acao = self.detalhar_acao_de_correcao(item)
        acaos.append(acao.url)

    data = {
        'regras': regras,
        'acao_correcao': acaos,
        'dados': dados
    }
    data = json.dumps(data, separators=(",", ":"))
    response = requests.post(url, data=data, headers=headers)
    if response.status_code == 200:
        return response
    else:
        raise f'Erro na requisição: {response.status_code}'
```

Fonte: Próprio autor (2023)

Primeiro, o código itera sobre os códigos de regras e ações de correção fornecidos, obtendo as informações completas de cada regra e ação por meio dos métodos "detalhar_regra" e "detalhar_acao_de_correcao". Em seguida, as URLs das regras e ações são armazenadas nas listas de regras e ações, respectivamente.

Após isso, é construído o *payload* (dados a serem enviados) da requisição POST, contendo as URLs das regras e ações, juntamente com os dados a serem

corrigidos. O *payload* é convertido em formato JSON e enviado para a URL http://localhost:8000/correcaodedados/ utilizando a biblioteca requests.

A resposta da requisição é verificada e retornada se o status for 200 (OK). Caso contrário, uma exceção é lançada com a mensagem de erro correspondente.

Esse código exemplifica como fazer uma requisição para corrigir os dados utilizando as regras e ações de correção fornecidas, integrando o componente com o serviço Data Cleaner.

3.7.2. Uso no Script

O script tem como principal função realizar a importação e limpeza de dados de um arquivo de tabela (CSV, Excel, ODT ou ODS). Ele utiliza o Data Cleaner Service Component para aplicar regras de validação e ações de correção nos dados, conforme especificado no dataset.

FIGURA 13 - Classe Main do Script

```
def main():
    path = 'arquivos//Untitled 1.csv'
    df = ler_tabela(path)
    regras = {
        0: ["RG001"],  # Id
        1: ['RG006'],  # Nome
        2: ['RG002'],  # Sexo
        3: ['RG001'],  # Altura(cm)
        4: ['RG004']  # Dt nascimento
    }
    acao = {
        0: ['AC001', 'AC002'],
        1: ['AC003'],
        2: ['AC004', 'AC005'],
        3: ['AC006', 'AC009', 'AC007'],
        4: ['AC008']
    }
    corrigido = []
    lista_dados, colunas = obter_lista_colunas(df)
    item = 0
    for dado in lista_dados:
        dado = componente.corrigir_dados(dados=dado, cod_regras=regras.get(item), cod_acao_de_correcao=acao.get(item))
        corrigido.append(dado.json())
        item += 1
    print(corrigido)
    print(colunas)
    df = pd.DataFrame(columns=colunas)
    coluna_index = 0
    for item in corrigido:
        for row in item['dados']:
            index, values = row
            if len(df) <= index:
                df.loc[index] = [None] * len(colunas)
            df.iloc[index, coluna_index] = values if values is not None and values != '' else '-'
        coluna_index += 1

    print(type(df))
    nome_arquivo = 'Novo.csv'
```

Fonte: Próprio autor (2023)

O script lê o arquivo, transforma as colunas em linhas, aplica as correções necessárias nos dados e gera um novo arquivo com os dados limpos. A função principal, “main()”, coordena todo esse processo, utilizando as funções auxiliares como “ler_tabela()”, “obter_lista_colunas()”, “ordena_dados()”, entre outras. No final, o script gera um novo arquivo CSV chamado "Novo.csv" com os dados limpos e corrigidos.

3.7.3. Aplicação Frontend com Interface Gráfica

Neste trabalho, realizamos uma demonstração visual de limpeza de dados, utilizando a biblioteca Flask para auxiliar no desenvolvimento. Exploramos o processo

de limpar e organizar conjuntos de dados, conseguimos criar uma interface intuitiva e interativa para visualizar e interagir com os dados limpos

FIGURA 14 - Função “tela_3”.

```python
@main_blueprint.route('/tela3')
def tela_3():
    # Obter os dados corrigidos da requisição anterior
    dados_corrigidos_json = request.args.get('dados_corrigidos')
    # Decodificar a string de consulta para obter os dados como uma lista de dicionários
    dados_decodificados = urllib.parse.parse_qs(dados_corrigidos_json)
    colunas = ast.literal_eval(dados_decodificados['colunas'][0])
    dados_corrigidos = ast.literal_eval(dados_decodificados['dados'][0])
    tabela_errada = []
    tabela_corrigida = []
    for item in dados_corrigidos:
        item['correto'] = [[index, value, 'correto'] for index, value in item['correto']]
        item['errado'] = [[index, value, 'errado'] for index, value in item['errado']]
        item['corrigido'] = [[index, value, 'corrigido'] for index, value in item['corrigido']]
        item['nao_corrigido'] = [[index, value, 'errado'] for index, value in item['nao_corrigido']]
        valores = item['correto'] + item['errado']
        tabela_errada.append(sp.ordena_dados(valores))
        valores = item['correto'] + item['corrigido'] + item['nao_corrigido']
        tabela_corrigida.append(sp.ordena_dados(valores))

    # Transformar os dados em um dicionário com as colunas como chaves
    tabela_errada_dict = list(map(list, zip(*tabela_errada)))
    tabela_corrigida_dict = list(map(list, zip(*tabela_corrigida)))

    # Cria dataframe Partir dos dicionarios.
    df_errada = pd.DataFrame(tabela_errada_dict, columns=colunas)
    df_corrigida = pd.DataFrame(tabela_corrigida_dict, columns=colunas)

    # Definir a função para formatar as células com a classe desejada

    # Aplicar a formatação às células do dataframe
    df_estilizado_errada = df_errada.applymap(lambda valor: formatar_celula(valor[1], valor[2]))
    df_estilizado_corrigido = df_corrigida.applymap(lambda valor: formatar_celula(valor[1], valor[2]))

    # Converter o dataframe estilizado em uma tabela HTML
    tabela_html_errada = df_estilizado_errada.to_html(index=False, escape=False)
    tabela_html_corrigido = df_estilizado_corrigido.to_html(index=False, escape=False)

    # Remover a classe "dataframe" e substituí-la por "table table-bordered table-dark"
```

Fonte: Próprio autor (2023)

FIGURA 15 – Continuação da Função “tela_3”

```
# Converter o dataframe estilizado em uma tabela HTML
tabela_html_errada = df_estilizado_errada.to_html(index=False, escape=False)
tabela_html_corrigido = df_estilizado_corrigido.to_html(index=False, escape=False)

# Remover a classe "dataframe" e substitui-la por "table table-bordered table-dark"
tabela_html_errada = tabela_html_errada.replace('class="dataframe"', 'class="table table-bordered niceTable"')
tabela_html_corrigido = tabela_html_corrigido.replace('class="dataframe"', 'class="table table-bordered niceTable"')

# Remover células vazias
tabela_html_errada = tabela_html_errada.replace('<td><td', '<td').replace('</td></td>', '</td>')
tabela_html_corrigido = tabela_html_corrigido.replace('<td><td', '<td').replace('</td></td>', '</td>')

return render_template('tela_3.html', tabela_errada=tabela_html_errada, tabela_corrigida=tabela_html_corrigido)
```

Fonte: Próprio autor (2023)

1. Obtém os dados corrigidos da requisição anterior, que são passados como parâmetros na URL e recuperados utilizando o `request.args.get`.
2. Decodifica a *string* de consulta obtida para obter os dados corrigidos como uma lista de dicionários, utilizando a biblioteca `urllib.parse.parse_qs`.
3. Realiza a manipulação dos dados corrigidos para formatá-los corretamente, organizando os valores corretos, errados, corrigidos e não corrigidos em listas.
4. Transforma os dados corrigidos em dicionários, onde as colunas são as chaves.
5. Cria dois DataFrames do pandas com os dados corrigidos: um para a tabela errada (`df_errada`) e outro para a tabela corrigida (`df_corrigida`).
6. Define uma função para formatar as células das tabelas com a classe desejada.
7. Aplica a formatação às células dos DataFrames utilizando o método `applymap`.
8. Converte os DataFrames estilizados em tabelas HTML utilizando o método `to_html`.
9. Realiza ajustes nas tabelas HTML, como substituir a classe "dataframe" pela classe "table table-bordered niceTable" e remover células vazias.
10. Renderiza o template 'tela_3.html', passando as tabelas HTML como parâmetros para serem exibidas na página.

Em resumo, essa função recebe os dados corrigidos da requisição anterior, os processa, cria DataFrames, estiliza as células dos DataFrames, converte-os em tabelas HTML e renderizar uma página HTML com as tabelas exibidas. Essa página será usada para visualizar os resultados da correção de dados de forma tabular.

FIGURA 16 - Demonstração visual dos dataset.

Fonte: Próprio autor (2023).

3.8. Scripts de Limpeza de Dados

O sistema consiste em dois objetos principais: um objeto de classe para armazenar as regras e outro objeto de ação de correções. A seguir, apresentamos um exemplo de uma regra específica:

FIGURA 17 - Objeto da Classe Regra.

```
id: 4,
dt_criacao: 2023-06-26,
dt_modificacao: 2023-06-26,
codigo: RG004,
titulo: Validar data dia, mês e ano,
descricao: Válida se o dado recebido está no formato de
dia, mês e ano (DD/MM/YYYY),
tipo: DAT,
script:
"from datetime import datetime

dado = elemento

try:
     datetime.strptime(dado, '%d/%m/%Y')
     value = True
except ValueError:
     value = False"
```

Fonte: Próprio autor (2023).

Como já citado há dois objetos principais: um objeto de classe, que armazena as regras, e um objeto de ação de correção, responsável por executar as correções. A seguir, apresentamos um exemplo de um objeto do tipo "ação de correção":

FIGURA 18- Objeto da Classe Ação de Correção.

```
id: 8,
dt_criacao: 2023-06-26,
dt_modificacao: 2023-06-26,
codigo: AC008,
titulo: Transforma as datas em Dia/Mês/Ano,
descricao: "Transforma todas as datas validas em dia, mês e
ano (dd/mm/YYYY)",
tipo: DAT,
script:
"from dateutil import parser

try:
     data_parseada = parser.parse(elemento)
     data_formatada = data_parseada.strftime(""%d/%m/%Y"")
     elemento = data_formatada
     value = True
except:
     elemento = '-'
     value = True"
```

Fonte: Próprio autor (2023).

3.9. Dados de Exemplo Utilizados

Segue o CSV que será utilizado para demonstração.

QUADRO 02 - CSV usado para demonstração.

ID	Nome	Sexo	Altura(cm)	Data Nascimento
1	Antonio Gael Gael Ramos	Masculino	178	23-02-1990
2	Elisa Márcia Vera	F	asd	
3	Geraldo Martin Raimundo Ramos	M	1,5	31/03/1985
4	gabriel picanco	m	1,8	07/07/1997
5	Pietro André Julio Freitas		166	01/31/1969
6	Ester Larissa	Feminino	155	asdasd

Fonte: Próprio autor (2023).

4. CONSIDERAÇÕES FINAIS

Neste trabalho, foi desenvolvida uma ferramenta de limpeza de dados, baseada em um serviço web. O objetivo central consistiu em criar um serviço web capaz de realizar a limpeza de dados com base em regras predefinidas pelo usuário. O foco foi receber um Dataset com problemas de qualidade e fornecer ao usuário um novo Dataset validado e corrigido, levando em consideração as configurações personalizadas definidas por ele.

Para demonstrar a eficácia da nossa abordagem, desenvolvemos dois demonstrativos: o primeiro é uma biblioteca, e o segundo é um serviço frontend que consome a aplicação, tornando-a facilmente acessível e interativa para os usuários

Ao conceber a solução, foram exploradas diversas tecnologias relacionadas à limpeza de dados. Contudo, devido às restrições de tempo e recursos do projeto, optou-se por concentrar a implementação principalmente na tecnologia Python, utilizando arquivos CSV como formato de dados.

Durante o desenvolvimento, identificaram-se alguns riscos significativos. Destaca-se, entre eles, a complexidade de integrar diferentes tecnologias e ferramentas, como Python, Google Planilhas e Javascript. Adicionalmente, o desempenho da solução ao lidar com grandes volumes de dados revelou-se um desafio a ser enfrentado. Para mitigar esses riscos, priorizaram-se o desenvolvimento e a garantia da qualidade desses aspectos sempre que possível.

Os resultados obtidos até o momento demonstram-se altamente promissores. A solução revelou-se eficaz na limpeza de dados, oferecendo aos usuários conjuntos de dados validados e corrigidos. Contudo, é crucial reconhecer as limitações do software em sua versão atual: ainda não é possível interpretar outras linguagens de script, nem realizar a integração com outras ferramentas de limpeza de dados.

Como perspectiva para trabalhos futuros, almeja-se ampliar significativamente essa solução. Planeja-se desenvolver novos componentes, como a capacidade de interpretar outras linguagens de script e estabelecer integrações com diversas ferramentas de limpeza de dados. Além disso, pretende-se implementar um frontend otimizado para o serviço, aprimorando a interação dos usuários, tornando-o mais intuitivo para melhor aproveitamento da experiência do usuário.

5. REFERÊNCIA

ABIB, G. A qualidade da informação para a tomada de decisão sob a perspectiva do sensemaking: uma ampliação do campo. **Ciência da Informação**, v. 39, n. 3, p. 73–82, dez. 2010.

DE JONGE, E. An introduction to data cleaning with R. p. 53, 2013.

Google Sheets API Overview. Disponível em: <https://developers.google.com/sheets/api/guides/concepts>. Acesso em: Jun. 2022.

Introduction to TIBCO Clarity. concept. Disponível em: <https://docs.tibco.com/pub/clarity/2.0.1/doc/html/GUID-B611C543-3218-4EFF-BC79-C9A31BA5D670.html>. Acesso em: Jun. 2022.

Jinja Documentation (3.1.x). Disponível em: <https://jinja.palletsprojects.com/en/3.1.x/>. Acesso em: Jun. 2022.

MARINO, V. **Qual a melhor linguagem para Ciência de Dados?Medium**, 26 set. 2018. Disponível em: <https://medium.com/@vmarino/qual-a-melhor-linguagem-para-ci%C3%AAncia-de-dados-bc06606352fe>. Acesso em: Jun. 2022

Microsoft Excel. Disponível em: <https://www.microsoft.com/pt-br/microsoft-365/excel>. Acesso em: Maio. 2022.

Modular Applications with Blueprints - Flask Documentation (2.1.x). Disponível em: <https://flask.palletsprojects.com/en/2.1.x/blueprints/>. Acesso em: Jun. 2022.

OLIVEIRA, P.; RODRIGUES, F.; RANGEL HENRIQUES, P. **Limpeza de Dados -Uma Visão Geral**. Jun. 2022.

OpenRefine. Disponível em: <https://openrefine.org/>. Acesso em: Maio. 2022.

Pandas 1.4.3 Documentation. Disponível em: <https://pandas.pydata.org/docs/>. Acesso em: Jun. 2022.

PEP 8 – Style Guide for Python Code. Disponível em: <https://peps.python.org/pep-0008/#introduction>. Acesso em: Jun. 2022.

PRESS, G. **54 Predictions About The State Of Data In 2021**. Disponível em: <https://www.forbes.com/sites/gilpress/2021/12/30/54-predictions-about-the-state-of-

data-in-2021/?sh=5c600904397d>. Acesso em: Maio. 2022.

Python Geral - documentação Python. Disponível em: <https://docs.python.org/pt-br/3/faq/general.html#general-information>. Acesso em: Jun. 2022.

SQLAlchemy - The Database Toolkit for Python. Disponível em: <https://www.sqlalchemy.org/>. Acesso em: Jun. 2022.

WANG, R. Y.; ZIAD, M.; LEE, Y. W. **Data Quality**. Springer Science & Business Media, 2006.

Welcome to Flask - Flask Documentation (2.1.x). Disponível em: <https://flask.palletsprojects.com/en/2.1.x/>. Acesso em: Jun. 2022.

Werkzeug Documentation (2.1.x). Disponível em: <https://werkzeug.palletsprojects.com/en/2.1.x/tutorial/>. Acesso em: Jun. 2022.

Provost, F., & Fawcett, T. (2013). **Data Science para Negócios**. Alta Books.

Davenport, T. H., & Patil, D. J. (2012). **Data Scientist: O profissional mais cobiçado do século XXI**. Harvard Business Review Brasil.