THIAGO CARDOSO

ALGORITMO DE CONTROLE DE LIGAÇÕES SIMULTÂNEAS EM UM DISCADOR PREDITIVO

Joinville

2021

THIAGO CARDOSO

ALGORITMO DE CONTROLE DE LIGAÇÕES SIMULTÂNEAS EM UM DISCADOR PREDITIVO

Trabalho de Conclusão de Curso apresentado no curso de Bacharelado em Engenharia de Software do Centro Universitário Católica de Santa Catarina como requisito parcial para obtenção do certificado do curso.

Católica de SC – Joinville – Programa de Graduação

Orientador: MSc. Diogo Vinícius Winck

Joinville

2021

THIAGO CARDOSO
ALGORITMO DE CONTROLE DE LIGAÇÕES SIMULTÂNEAS EM UM DISCADOR PREDITIVO/ THIAGO CARDOSO. – Joinville, 2021- 81 p. : il. (algumas color.) ; 30 cm.

Orientador: MSc. Diogo Vinícius Winck

Trabalho de Conclusão de Curso – Católica de SC – Joinville – Programa de Graduação, 2021.

1. Discador 2. Telefonia 3. Algoritmo I. MSc. Diogo Vinícius Winck II. Centro Universitário Católica de SC – Joinville III. Algoritmo de Controle de Ligações Simultâneas em um Discador Preditivo CDU 02:141:005.7

THIAGO CARDOSO

ALGORITMO DE CONTROLE DE LIGAÇÕES SIMULTÂNEAS EM UM DISCADOR PREDITIVO

Trabalho de Conclusão de Curso apresentado no curso de Bacharelado em Engenharia de Software do Centro Universitário Católica de Santa Catarina como requisito parcial para obtenção do certificado do curso.

MSc. Diogo Vinícius Winck
Orientador

Professor
MSc. Mauricio Henning

Professor
MSc. Glauco Vinicius Scheffel

Joinville
2021

Este Trabalho de Conclusão de Curso é dedicado à minha família, principalmente meus pais e minha namorada, que foram fundamentais para que isso acontecesse, serei enternamente grato por todo o apoio e suporte fornecido.

Agradecimentos

Agradeço de forma especial meus pais, que nunca mediram esforços e sempre me incentivaram a estudar, à minha namorada, que esteve me apoiando nos momentos de dificuldade e me incentivou a continuar desenvolvendo o presente trabalho.

À todos, eterna gratidão!

"O insucesso é apenas uma oportunidade para recomeçar com mais inteligência."

Henry Ford

Resumo

A tecnologia tem se tornado cada vez mais essencial no mundo corporativo e vem automatizando boa parte dos trabalhos manuais que existiam no mundo. As rotinas de contato com o cliente estão tornando-se cada vez mais automatizadas, sem deixar de serem humanas, trazendo ganho para ambas as partes. Por outro lado, clientes estão cansados de serem metralhados de e-mails, mensagens e ligações indesejadas realizadas por robôs, por isso é necessário encontrar um meio termo, que esteja de acordo para ambas as partes. Pensando em atender ambas às expectativas, o presente trabalho tem o objetivo de construir um algoritmo que calcula quantas discagens são realizadas por um discador preditivo de uma empresa, automatizando um processo realizado manualmente, preservando a experiência do cliente. O algoritmo desenvolvido no presente trabalhado manipula dados para gerar inteligência e utiliza tecnologias como: Python, SQL e Airflow.

PALAVRAS-CHAVES: Discador, Telefonia, Algoritmo.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
JSON	*Java Script Object Notation*
KPI	*Key Performance Indicator*
SQL	*Structured Query Language*
TI	*Tecnologia da Informação*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Uma empresa de tecnologia, comercializa um *software* como serviço e concede aos clientes a oportunidade de testar o *software* durante três dias. Nesse período de testes, o cliente é enviado para um processo de discagem automatizada para seu telefone, com base no telefone preenchido em seu cadastro, onde a empresa realiza uma abordagem de qualificação de vendas, através do time comercial. A tecnologia do discador preditivo, tem auxiliado a empresa na abordagem aos clientes, mas gera dúvidas na gestão das discagens, principalmente no momento de definir a quantidade de ligações simultâneas que devem ser realizadas pelo discador.

Segundo Júnior et al. (2015), no início dos anos 2000 a internet chegou aos *call center*s brasileiros e com o objetivo de ganhos de escala, a tecnologia de ligações via Voip (Voz por IP, que utiliza fibras óticas para ligações via internet) foi introduzida, ocorrendo uma revolução no setor e reduzindo custos de chamadas preditivas, no qual um algoritmo de discador realiza ligações simultâneas e reduz o tempo ocioso do operador de *telemarketing*, aumentando massivamente a quantidade de ligações que o operador realiza.

O discador preditivo é um *software* que procura aumentar a produtividade dos agentes de um *call center*s através de lógica própria que decide quando deve ser iniciada uma nova discagem e quantas tentativas simultâneas devem ser realizadas (YONAMINE et al., 2007).

O presente trabalho de conclusão de curso, busca demonstrar a construção de um algoritmo, que visa automatizar o processo de controle de ligações simultâneas utilizado no discador preditivo da empresa de tecnologia Conta Azul.

O controle de ligações simultâneas, denominado fator de aceleração do discador, tem se demonstrado forte aliado no aumento do percentual de atendimento das ligações do discador e por consequência na conversão de novos clientes. A dependência diária de um controle manual realizado por uma pessoa responsável pela operação, tem se tornado crítico e falho, além da geração de custo adicional para a empresa.

A pesquisa será focada em estatística, análise e ciência de dados, buscando produtização de forma escalável e correlações com temas da área de tecnologia. A análise e ciência de dados irão contribuir para análise do cenário atual, visando um cenário futuro automatizado e produtizado. A estatística entra como base para criar o modelo estatístico a ser utilizado na produtização, além de contribuir para as análises do cenário atual.

Espera-se que o produto final, entregue uma automatização baseada em regras com base sólida estatística e que além da automatização do processo, contribua de forma

positiva para o percentual de atendimento das ligações realizadas pelo discador.

1.1 DEFINIÇÃO DO PROBLEMA

Em maio de 2019, a empresa de *software* Conta Azul contratou uma nova ferramenta de telefonia, denominada Hosanna. Essa ferramenta possui uma funcionalidade que permite a realização de discagens automatizadas para telefones. A contratação visava o aumento do percentual de atendimento das ligações, pois a abordagem via telefone é a principal fonte de vendas.

Com o passar dos meses, a ferramenta foi sendo implementada e o time de vendas da Conta Azul começou a ter bons resultados. Durante a implementação, foi percebido que era possível realizar acelerações e desacelerações no processo de discagem, permitindo ligar para mais ou menos clientes simultâneamente por minuto. Essa aceleração e desaceleração do discador é controlada por um fator chamado "fator de aceleração", que determina a quantidade de discagens que o discador irá realizar por minuto, ou seja, quanto maior o fator, o discador passa discar para mais pessoas, quanto menor o fator, o discador passar a discar para menos pessoas. Vale ressaltar, que o aumento desse fator visa produzir uma espécie de fila, fazendo com que caso não haja agentes disponíveis na operação para todos os clientes, os mesmos ficam em uma fila de espera e caso desejado, desconectam a chamada.

Foi percebido que era possível aprimorar ainda mais o processo. Controlando o fator de aceleração, era possível aumentar ainda mais o percentual de ligações atendidas durante as tentativas do discador e consequentemente o faturamento da empresa, visto que conversão continuava sendo a mesma. A liderança responsável pelo setor de vendas da empresa decidiu então, alocar uma pessoa dedicada para realizar a alteração do fator de aceleração do discador no processo de discagens, visando o aumento dos indicadores.

De forma desestruturada, os indicadores realmente tiveram um pequeno ganho, porém ainda com a necessidade de possuir uma pessoa alocada na posição, com a responsabilidade por controlar o fator. O tema de discussão então, passou a ser o custo de aquisição por cliente (CAC), fazendo o corpo de liderança refletir se a o custo de fato pagava-se quando a contratação era realizada.

O custo de aquisição por cliente (CAC) começa a ser estudado e um dos temas de discussão é a alocação de uma pessoa dedicada para controle do fator e se esse retorno cobria essa despesa, além das demais. O faturamento de cada cliente paga essa despesa? Manter uma pessoa controlando o fator é escalável para a empresa? Faz realmente sentido controlar esse parâmetro? Há uma forma de automatizar esse controle?

1.2 JUSTIFICATIVA

O presente estudo tem como objetivo avaliar e automatizar o processo de controle do fator aceleração do discador. Este fator é importante aliado na abordagem direta ao possível cliente, principal fonte de abordagem da empresa.

O projeto é relevante pois vai automatizar um processo totalmente manual, que exige de controle e monitoramento constante. Além disso, o projeto vai abordar tópicos em alta no mercado, como a cultura *data driven*, ciência e análise de dados e estatística. A solução final pode servir de ponto de partida para futuras pesquisas relacionadas e abordadas em projetos correlacionados.

O impacto esperado gira em torno principalmente do aumento do percentual de atendimento das ligações, leve diminuição do custo de aquisição por cliente (CAC) e pequeno aumento na conversão dos clientes abordados através desse canal. Indiretamente, o produto final irá impactar positivamente em métricas de acompanhamento de produtividade internas do time e experiência do cliente.

Atualmente, a tecnologia dominante de discagem ativa é conhecida como preditiva. Calcula o tempo médio de atendimento, o nível de abordagem (percentual de clientes alvo encontrados em relação ao número de ligações), a quantidade de trabalhadores e com base em um algoritmo faz a discagem, filtra as ligações não atendidas, identifica tons de ocupado, mensagens das operadoras telefônicas, reconhece o ruído ambiente e a voz humana e somente nesse momento transfere para o operador. O que reduz de forma drástica o tempo improdutivo e aumenta o tempo falado do trabalhador (JÚNIOR et al., 2015).

As inovações tecnológicas desde a discagem eletrônica, discagem múltipla, Voip, os filtros de discagem que reduzem as chamadas "não humanas" até a combinação de algoritmos e reconhecimento de voz e padrões de contato só aumentam o tempo efetivamente trabalhado, reduzem o tempo que o trabalhador fica sem falar, contraindo o tempo entre uma chamada e outra de minutos até poucos segundos (JÚNIOR et al., 2015).

Além disso, conforme conclusão realizada na avaliação no artigo de Yonamine et al. (2007), indica que apenas com a implementação de um discador preditivo (discagens ativas automatizadas) e com tentativas de ligações simultâneas, é notado uma influência no ganho de produtividade. O presente projeto visa desta forma, atacar mais uma frente, que é a automatização do controle dessas ligações simultâneas, que pode gerar um impacto ainda maior no ganho de produtividade.

Portanto, o trabalho buscará apontar falhas no processo realizado de forma manual e a ineficácia do modelo de controle do fator de aceleração adotado. Além disso, tentará provar com base em dados, que o sistema atual gera prejuízo e como o novo cenário pode contribuir positivamente.

1.3 OBJETIVO GERAL

O objetivo geral do presente projeto é elaborar um algoritmo, que realize com efetividade, a tarefa de automatizar o processo de atualização do fator de aceleração do discador na Conta Azul, baseado em estatística e dados, para que desta forma torne o processo de discagens mais eficiente e o mais próximo possível do ideal.

1.4 OBJETIVOS ESPECÍFICOS

Diante do objetivo geral estabelecido anteriormente, busca-se atingir os objetivos abaixo:

- Pesquisar trabalhos correlacionados com o tema;
- Identificar como o processo é realizado atualmente;
- Analisar dados de discagens;
- Criar método de cálculo para utilização no algoritmo;
- Criar fluxo de extração de dados de discagens;
- Desenvolver protótipo do algoritmo;
- Implementar protótipo na operação;
- Comparar comportamento pré e pós implementação.

1.5 METODOLOGIA

Nesta seção serão abordados os metódos utilizados para definir e atingir os objetivos específicos e geral do trabalho. As necessidades serão levantadas conforme o andamento do projeto e as metologias visam suprir essas necessidades. A metodologia científica, portanto, deve abordar todos os passos descritos de forma escrita, que serão necessários para atingimento dos objetivos do projeto. Segundo Reis (2009), o trabalho científico é uma atividade intencional, processual e complexa de produção de conhecimentos para a interpretação da realidade.

No Capítulo 1 (introdução) é definido quais são as etapas necessárias para se atingir os objetivos. Cada etapa tem sua importância e tem ligação direta com o atingimento de objetivos específicos. Segundo Castro (1978), não há um roteiro exato a ser seguido, mas sim se cada etapa contribui para o desenvolvimento total do projeto.

Na primeira etapa é definido o motivo do trabalho e deve ser levantado o problema a ser abordado. Juntamente com o problema é necessário demonstrar o valor agregado a este estudo.

Na segunda parte do método é necessário responder o motivo levantado na primeira etapa, aplicando os conhecimentos na definição do objetivo geral e objetivos específicos do projeto.

Seguindo para a terceira etapa, é então realizada uma análise de viabilidade do projeto, juntamente com os resultados de dados extraídos das pesquisas é então determinado quais passos são necessários para a realização do projeto.

No Capítulo 2 é abordada a fundamentação teórica do projeto, onde foi escolhido o método de pesquisa bibliográfica. Conforme Macedo (1995), é a busca de informações bibliográficas, seleção de documentos que se relacionam com o problema de pesquisa, como livros, verbetes de enciclopédia, artigos de revistas, trabalhos de congressos, teses, etc. Na aplicação do método de pesquisa bibliográfica são levantadas fontes de pesquisas relacionadas ao tema do trabalho e são utilizadas técnicas de validação de conteúdo a ser utilizado na fundamentação do presente trabalho.

Posteriormente, no Capítulo 3, é aplicado o planejamento de execução do projeto, abordando todas as partes do trabalho. Nesse capítulo, são levantadas as tarefas, que são distribuídas em um cronograma de execução, até que todas sejama concluídas completamente. O plano indica os elementos que aparecerão nas diferentes partes da redação. Todavia, o planejamento pode ser mudado no decorrer da elaboração do trabalho, de acordo com novas necessidades que venham a surgir (ANDRADE, 2010).

No Capítulo 4, é apresentado o processo de desenvolvimento do projeto, trazendo exemplos de técnicas estatísticas abordadas ao longo do algoritmo, bem como trechos de código fonte utilizando as lógicas para devida automatização. Para organização dessa etapa foi utilizado os conceitos da metodologia ágil *Scrum*.

O projeto por sua vez, é finalizado no Capítulo 5 (resultados), onde são trazidos os resultados da pesquisa, bem como conclusões obtidas ao longo do desenvolvimento de todo o projeto. Nesse capítulo também é possível concluir se o projeto de fato atigiu seus objetivos e quais melhorias ainda podem ser tratadas futuramente, caso hajam. Segundo Colzani (2002), é nessa etapa que se apresentam os resultados finais obtidos com todo o desenvolvimento do projeto em questão, onde, de modo sucinto, é mostrada uma análise interpretada dos argumentos lançados no decorrer do presente trabalho. Concluindo o projeto, é necessário que nesse capítulo a solução seja comprovada ou descartada.

2 FUNDAMENTAÇÃO

Neste capítulo são abordados assuntos que fundamentam a parte bibliográfica do projeto, onde é realizado o levantamento de informações que visam contribuir para o desenvolvimento do projeto como um todo. Este capítulo é divido em duas seções: trabalhos relacionados e pilha tecnológica.

Na seção 2.1 (trabalhos relacionados) são encontrados trabalhos já desenvolvidos e documentados, que tem relação direta ou indireta com o presente projeto.

Na seção 2.2 (pilha tecnológica) são documentadas todas as tecnologias utilizadas no projeto, bem como a justificativa do por quê cada tecnologia foi escolhida e particularidades relevantes de cada tecnologia.

2.1 TRABALHOS RELACIONADOS

Nesta seção, serão abordados todos trabalhos selecionados que possuem escopo correlato com o presente trabalho. Os trabalhos relacionados foram pesquisados e selecionados entre simpósios, livros que tratam de assuntos correlatos, aplicações que já existam no mercado e tenham escopo relacionado, além de artigos encontrados na *web*. Para encontrar as melhores fontes, para melhor seleção dos trabalhos, foi utilizado como critério principal o nível de proximidade com o presente trabalho.

2.1.1 O Processo de Trabalho e Criação de Valor nos Callcenters Brasileiros

O artigo escrito pelo autor Júnior et al. (2015), da Universidade Federal de Santa Catarina (UFSC), aborda o processo de evolução dos *call center*s localizados no Brasil e como o uso de tecnologias tem contribuído historicamente para a evolução dos mesmos. É ressaltado também, como a tecnologia tem tornado o dia a dia dos profissionais de telemarketing cada vez mais produtivas e otimizadas, principalmente em operações preditivas onde são executadas ligações ativas, reduzindo a entrega de chamadas com falha e consequentemente o tempo ocioso dos profissionais, além de aumentar o lucro para o setor.

As inovações tecnológicas desde a discagem eletrônica, discagem múltipla, Voip, os filtros de discagem que reduzem as chamadas "não humanas" até a combinação de algoritmos e reconhecimento de voz e padrões de contato só aumentam o tempo efetivamente trabalhado, reduzem o tempo que o trabalhador fica sem falar, contraindo o tempo entre uma chamada e outra de minutos até poucos segundos (JÚNIOR et al., 2015).

O artigo de Júnior et al. (2015), também evidencia o conceito de "mais valia", que é famosamente empregado por Karl Marx. O autor relaciona o termo "mais valia" com a realidade dos *call center*s brasileiros, onde cuidadosamente as empresas precisam comparar esforços *versus* resultados, de acordo com o que o termo abrange.

De forma geral, o artigo de Júnior et al. (2015) evidência, de forma embasada, a dificuldade de se encontrar um equilíbrio para o sucesso das operações dos *call center*s brasileiros e como a tecnologia tem contribuído de forma positiva para o histórico de evolução e cenário atualmente.

2.1.2 Simulação de um Call Center para Avaliação do Impacto da Adoção de um Discador Preditivo

O artigo desenvolvido pelo autor Yonamine et al. (2007) no Simpósio Brasileiro de Pesquisa Operacional, aborda uma simulação realizada para avaliar o impacto da adoção de um discador preditivo (discagens ativas automatizadas) em uma operação de *call center*. No estudo, é abordado a aplicação de uma metodologia que consiste em simular a implementação, em uma empresa de pequeno porte, afim de avaliar principalmente os impactos na produtividade da empresa.

Foram testadas lógicas em relação ao número de chamadas simultâneas realizadas e ao momento de início da discagem. Adicionalmente, avaliou-se o impacto na produtividade do *call center* variando-se o número de funcionários, o tempo médio de duração da ligação, a variabilidade da distribuição de tempo das chamadas e a taxa de sucesso para se completar uma ligação (YONAMINE et al., 2007).

O trabalho de Yonamine et al. (2007), aborda também aborda análises de vários cenários que foram sendo testados durante a simulação. Um dos cenários testados na simulação, foi o número de ligações simultâneas, onde foi simulado com o cenário base, onde as ligações eram executadas manualmente pelos pesquisadores e o cenário de discador preditivo (discagens ativas automatizadas) com 1 (uma) e 2 (duas) chamadas simultâneas. Conforme a Tabela 1 abaixo, é possível perceber que a alteração no formato de discagem já trás uma ganho de produtividade dos operadores, onde conclui-se que simplesmente com a adoção da discagem automática permitiria executar a mesma pesquisa em um prazo menor ou utilizando um número menor de operadores.

Tabela 1 – Resultado da simulação do número de tentativas simultâneas de chamadas

Chamadas simultâneas	*Tempo de discagem*	*% Abandono*	*% em Pesquisa*	*% Falando*	*tempo em pesquisa (min)*	*tempo falando (min)*
Base	Base	0,00%	51,24%	66,76%	30,7	40,1
1	Base	0,00%	60,48%	78,72%	36,3	47,2
2	Base	10,67%	70,88%	92,39%	42,5	55,4

Fonte: (YONAMINE et al., 2007)

Resumidamente, o artigo de Yonamine et al. (2007), trás uma panorama detalhado de cada cenário sendo automatizado através do discador automatizado que a tecnologia nos proporciona. Além disso, o artigo conclui que a automatização contribuiu de forma positiva para o cenário de produtividade da empresa avaliada, o que reforça a necessidade de automatização de operações de *call center* e também o tema do presente trabalho.

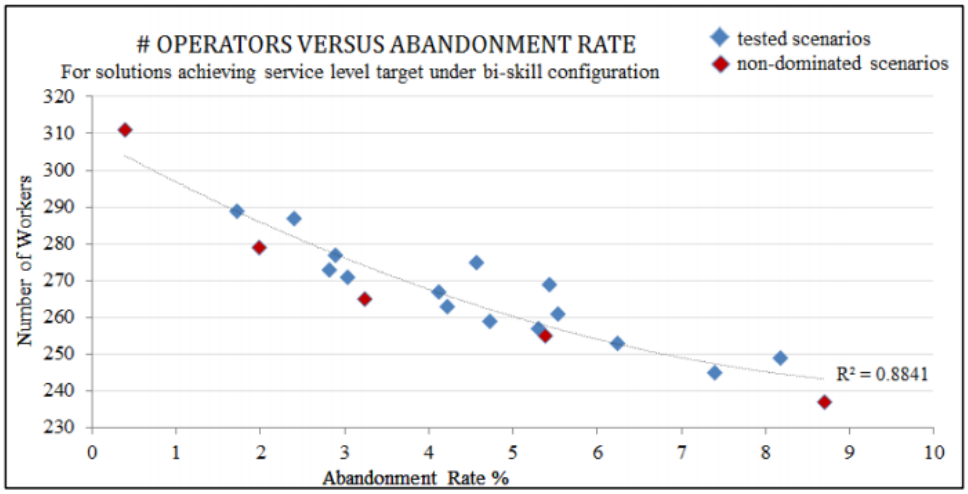
2.1.3 Understanding the Trade-offs in a Call Center

O artigo de Munoz e Brutus (2013), da Universidade Estadual da Pensilvânia nos Estados Unidos (EUA), descreve a dificuldade na tomada de decisão no momento de definir a força de trabalho adequada para atingimento de metas em operações de *call center*. Durante o artigo, os autores ainda demostram uma simulação que pode ser avaliada para reduzir custos em operações de *call center*.

A simulação de eventos discretos representa uma boa alternativa não apenas para determinar o número adequado de trabalhadores necessários para atingir as metas, mas também como forma de visualizar a relação entre os objetivos e metas. A viabilidade econômica de relaxar um ou mais alvos pode ser facilmente determinada após a compreensão de suas compensações. No entanto, os custos de oportunidade ocultos devem ser considerados cuidadosamente (MUNOZ; BRUTUS, 2013).

Ainda segundo Munoz e Brutus (2013), durante o processo de simulação, os testes foram realizados em uma empresa chilena com 600 funcionários. A simulação constatou que apenas sacrificando a taxa de abandono de (5%) para (8%), reduziu-se para 237 trabalhadores necessários para o mesmo atingimento de nível de serviço, conforme o gráfico da Figura 1 abaixo.

Figura 1 – Número de operadoras versus taxa de abandono para (85%) do nível de serviço



Fonte: (MUNOZ; BRUTUS, 2013)

Os autores Munoz e Brutus (2013) ainda concluem que, com apenas uma única modificação, essa configuração alternativa representa uma economia operacional de quase US $200 mil por ano, se comparada com o cenário atual. Em resumo, o artigo trás uma visão simplificada, de quão pequenas ações podem influenciar de forma positiva no cenário de um *call center*, sendo de suma importância a realização de simulações para tal validação.

2.1.4 Influence of the Contact Center Systems Development on Key Performance Indicators

O artigo de Płaza e Pawlik (2021), do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) nos Estados Unidos (EUA), destaca os principais *KPIs* que devem ser acompanhados em uma operação de *call center*, como: nível de serviço, custo por contato, satisfação do cliente, tempo médio de tratamento, resolução na primeira chamada, taxa de abandono, tempo médio de espera e taxa de ocupação, sendo analisado de forma crítica o desempenho de cada um. Vale ressaltar que o artigo foi publicado em 2021 e por isso já contempla alguns cenários em meio à pandemia.

O futuro dos sistemas de *contact center* está associado principalmente às amplas possibilidades de implantação de métodos de inteligência artificial. Novas implementações de métodos de classificação de múltiplos critérios de mensagens de texto terão, sem dúvida, um forte impacto na qualidade do serviço e na otimização de custos. Além disso, uma combinação de métodos de reconhecimento de emoção com métodos de reconhecimento de intenção, que agora são a base para o desenvolvimento de *chatbot*s e *voicebot*s, pode melhorar a qualidade do atendimento ao cliente e a satisfação do cliente. Pode-se esperar que o rápido progresso tecnológico em breve traga o uso generalizado de *videobot*s,

agregando técnicas de reconhecimento facial às ferramentas de identificação de emoções (PŁAZA; PAWLIK, 2021).

Ainda segundo os autores Płaza e Pawlik (2021), essas tecnologias podem afetar os indicadores analisados, por isso foi definida uma tabela base com valores considerados padrões para os indicadores de *call center*, sendo alguns valores baseados em dados estatísticos, outros resultantes de contratos celebrados entre empresas prestadoras de serviços de *call center*, onde é indicado o valor ideal para cada *KPI* em uma operação de *call center*, conforme a Tabela 2 abaixo.

Tabela 2 – Principais indicadores de desempenho para sistemas CC

Symbol	Name of indicator	Industry values	Reference
SL	Service Level	80%	[4]
CPC	Cost Per Contact	$8-13$	[5]
C-SAT	Customer Satisfaction	90%	[6]
AHT	Average Handle Time	6 minutes	[4]
FCR	First Call Resolution	70-75%	[4, 7]
AR	Abandon Rate	5-8%	[6]
AWT	Average Waiting Time	17.28 seconds	[8]
OR	Occupancy Rate	60-80%	[6]

Fonte: (PŁAZA; PAWLIK, 2021)

Em suma, o artigo de Płaza e Pawlik (2021), conclui que a introdução das novas tecnologias e soluções garante uma melhor utilização do tempo de trabalho dos agente e também mostra que tanto o tempo médio de espera quanto o tempo médio de conversa são mais curtos hoje. O artigo ainda mostra que as mudanças são benéficas pois contribuem diretamente para a redução dos custos do serviço.

2.1.5 Efficient Data Analysis Pipelines

O artigo *Efficient Data Analysis Pipelines* escrito por Koivisto (2019), da Univerdade de Helsinque na Finlândia, trás uma visão detalhada sobre a construção de um *pipeline* eficaz e escalável de consumo de dados para análise de dados. As seções do artigo passam por todas as camadas necessárias para a construção deste *pipeline* e também descreve o uso do Apache Airflow como ferramenta de orquestração e o PostgreSQL como ferramenta de banco de dados, tornando-se ainda mais interessante, pois ambas as ferramentas fazem parte da pilha tecnológica do presente trabalho.

O *pipeline* em si pode ser bom para descrever o uso de uma estrutura de nível superior, como Apache Airflow, que torna o desenvolvimento de *pipeline*s maiores mais fácil sem sacrificar a flexibilidade. Todo o *pipeline* e suas camadas: fonte de dados, coleta,

armazenamento, pré-processamento e modelagem devem ser pensados como um todo para maximizar a eficiência do *pipeline* (KOIVISTO, 2019).

No decorrer do artigo de Koivisto (2019), são abordadas as camadas para a construção do fluxo. Na camada de fonte de dados, é onde ficam localizadas as origens dos dados, ainda sem tratamento algum. O dado então passa pela camada de coleta de dados, onde os dados são extraídos das fontes. Em seguida o dado passa pela camada de persistência, onde os dados são inseridos em um banco de dados, no artigo é sugerido o PostgreSQL. Em seguida, na camada de pré-processamento, o dado das diversas fontes é padronizado em um padrão único. Por fim, na camada de modelagem, o dado é de fato modelado e se necessário consolidado, a fim de servir como base para análises.

Em suma, o artigo de Koivisto (2019) trás um panorama geral de como deve ser aplicado um fluxo coerente para coleta, tratamento e análise de dados, afim de que se tornem úteis para tomada de decisão. Este artigo colabora de forma importante para o presente trabalho, pois o tratamento dos dados será essencial para a utilização dos dados no algoritmo de automatização do fator de aceleração do discador.

2.1.6 Estatística Básica Simplificada

O artigo Estatística Básica Simplificada, escrito pelos autores Carvalho e Campos (2008), aborda o âmbito da estatística básica, trazendo conceitos e noções básicas da disciplina. O artigo ainda, trás de forma clara e objetiva, o processo completo desde a pesquisa, coleta dos dados, até a escolha do melhor método estatístico para melhor reaproveitamento na análise dos dados.

É imprescindível sabermos que se trata de um ramo da Matemática Aplicada, uma metodologia, uma técnica científica, adotada para se trabalhar com dados, ou seja, com elementos de pesquisa. Esta metodologia, este método, consiste em uma série de etapas, iniciando pela coleta das informações (dos dados) que, após coletadas, passarão por uma organização e apresentação. Chegamos, daí, a uma fase complementar, na qual se dará a análise daqueles dados (já organizados e descritos). Ora, esta análise dos dados coletados funcionará como um meio, pelo qual chegaremos a uma conclusão. Esta, por sua vez, ensejará uma tomada de decisão (CARVALHO; CAMPOS, 2008).

Um tópico do artigo de Carvalho e Campos (2008), importante para o presente trabalho, é a abordagem de variáveis. Segundo Carvalho e Campos (2008), a variável é o que se está sendo investigado, podendo ser quantitativa ou qualitativa, onde o autor ressalta que a variável quantitativa pode ser atribuída a um valor númerico e a variável qualitativa pode ser definida por tratar-se de um valor em formato de texto.

Podemos considerar, que o artigo de Carvalho e Campos (2008) é um ponto de partida quando trata-se de estatística. Os autores fornecem pilares importantíssimos para

a iniciação de uma análise confiável de dados.

2.1.7 Ciência de Dados

O artigo dos autores Porto e Ziviani (2014), do Laboratorio Nacional de Computação Científica (LNCC), ressalta a importância da ciência de dados para todas as áreas de estudo da ciência. No decorrer do artigo são abordados os principais pilares da ciência de dados e também os principais desafios enfrentados pela área, em especial no Brasil.

O avanço tecnológico das últimas décadas culminou com a capacidade de obtenção e geração de imensos volumes de dados, tanto de fenômenos naturais quanto de sistemas artificiais. O novo cenário delineado nesse contexto abre em realidade novas necessidades, perspectivas e oportunidades de avanços tecnológicos relacionados ao desenvolvimento de técnicas, metodologias, modelos, algoritmos e arquiteturas para se fazer frente ao desafio de analisar e interpretar esses imensos volumes de dados que emergem em aplicações de diversas áreas do conhecimento. Há, portanto, um grande potencial tecnológico na pesquisa básica e aplicada em ciência de dados, tal como aqui discutido, dado o foco nos aspectos fundamentais da análise de dados em larga-escala com impacto em diferentes áreas de conhecimento básico bem como cenários de aplicação (PORTO; ZIVIANI, 2014).

Além disso, Porto e Ziviani (2014), reforçam a tese de que a ciência de dados pode ser utilizada em qualquer área de atuação e está no eixo das demais áreas de estudo, incluindo o governo federal, como mostra a Figura 2 abaixo.

Figura 2 – Desafios de pesquisa básica em ciência de dados para diversos cenários de aplicação no eixo ciência-indústria-governo

Fonte: (PORTO; ZIVIANI, 2014)

Em síntese, o artigo de Porto e Ziviani (2014) abstrai diversas dificuldades na estratégia de ciência de dados, como a falta de pesquisas na área, dificuldade com gestão e processamento de dados. Além disso o artigo ainda aborda fatores importantes para a análise de dados, destacando que o produto final sempre deve ser a produção de conhecimento. Os autores também destacam que a análise sempre deve partir de uma hipótese, onde então entra a busca dos dados e análise para validação. O artigo torna-se importante aliado para o presente trabalho, pois reforça a tese de validação de hipóteses e decisões baseadas em dados.

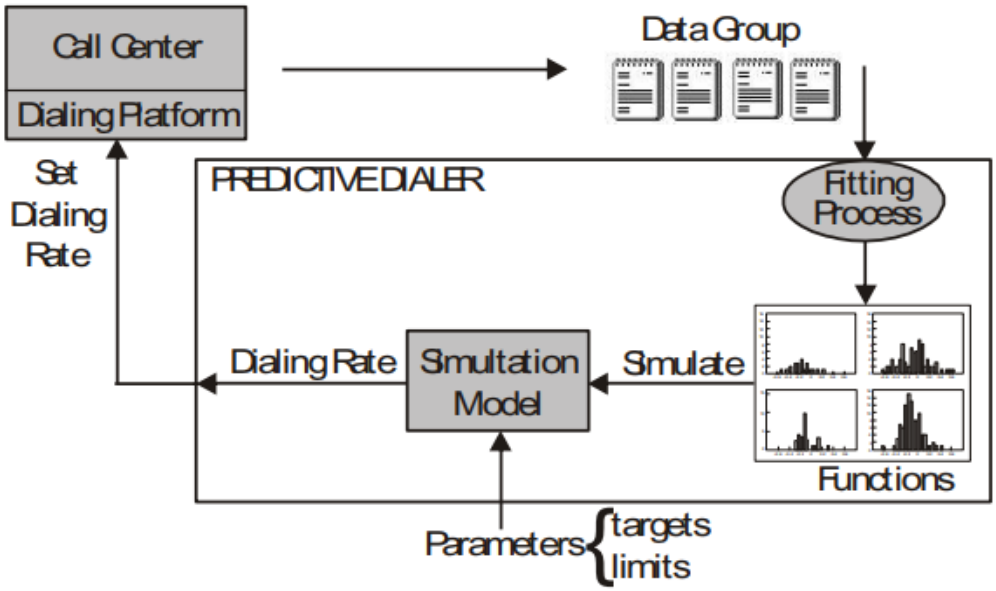
2.1.8 Using Simulation to Predict Market Behavior for Outbound Call Centers

O artigo de Filho et al. (2007), da Conferência de Simulação de Inverno de 2007 realizada em Washington nos Estados Unidos (EUA), fala sobre a importância de programar corretamente as discagens de um discador preditivo, a fim de que atinja os níveis de performance das operações de *call center*, respeitando os limites de taxas de abandonos das ligações.

A busca pela automatização da tarefa de discagem não é um assunto novo, tendo começado na década de 80. Mais recentemente, antes do discador preditivo, tínhamos o discador automático. Um discador automático é um dispositivo eletrônico que pode discar automaticamente números de telefone para se comunicar entre quaisquer dois pontos nas redes de telefonia e celular. Enquanto o discador automático básico apenas disca automaticamente para o telefone chamado. A função dos algoritmos preditivos do discador é, portanto, obter tempos de espera baixos, enquanto mantém as chamadas abandonadas em níveis aceitáveis (FILHO et al., 2007).

Segundo Filho et al. (2007), O discador preditivo deve capturar todos os dados dos eventos para simular o comportamento do ambiente. Os dados são capturados, analisados e classificados em classes ou grupos de variáveis permitindo que o modelo simule as ações dos agentes e todas as atividades temporais, como horários de discagem, horários de toque, horários de conversação, etc. O fluxo é demonstrado na Figura 3 abaixo.

Figura 3 – Diagrama de blocos do processo de adaptação do discador preditivo



Fonte: (FILHO et al., 2007)

Os autores Filho et al. (2007) do artigo, concluem que segundo resultados, o processo de consumo dos dados da própria operação do sistema para controles do fator de discagens, trás a vantagem de rápido ajuste, representando o comportamente atual, de acordo com as condições do *call center*. Além disso, os testes trouxeram resultados aceitáveis de desempenho durante o período da campanha. Em resumo, o artigo trouxe uma visibilidade importante quando abordamos fluxos para consumo de dados à serem utilizados para controle de discagens.

2.1.9 Optimization of a Predictive Dialing Algorithm

O artigo de Fourati e Tabbane (2010), fruto da Sexta Conferência Internacional Avançada de Telecomunicações de 2010, propõe uma melhoria no processo de discagens de um discador preditivo, utilizando um algoritmo que automatiza a quantidade de ligações simultâneas, baseado na cadeia de Markov de tempo contínuo (CTMC).

O objetivo principal é considerar a capacidade de prever de forma otimizada a melhor taxa de discagem, permitindo a compensação entre o número de chamadas abandonadas e a taxa de ocupação do agente (taxa de produtividade). As chamadas abandonadas são geradas quando o sistema de discagem preditiva obtém uma resposta mais humana nas tentativas de chamada do que os agentes disponíveis para atender essas chamadas. Em alguns países, como o Reino Unido e os EUA, existem regras que limitam o número de chamadas abandonadas, que um *call center* pode fazer em um determinado período de tempo, com a ameaça de grandes sanções para os centros que abusam desses sistemas (FOURATI; TABBANE, 2010).

O algoritmo proposto no artigo de Fourati e Tabbane (2010), utiliza principalmente como base os seguintes parâmetros: nível de serviço, taxa de abandono e ocupação dos agentes. O nível de serviço é estabelecido pela quantidade de ligações aceitáveis para que os atendimentos tenham qualidade adequada. A taxa de abandono é calculada a partir das chamadas que ficam na fila de espera, por não possuir agente disponível naquele momento e acabam abandonando a chamada. Já a ocupação dos agentes, é o tempo em que os agentes estão produzindo, ou seja, em atendimento. De acordo com estes parâmetros, o algoritmo proposto visa aumentar a taxa de ocupação dos agentes, respeitando a máxima taxa de abandono (5%).

Os resultados do algoritmo proposto por Fourati e Tabbane (2010), segundo o artigo, tem resultados positivos. Os autores concluem que o algortimo obteve um ótimo desempenho respeitando o limite de taxa de abandono e aumentando a ocupação dos agentes, causando impacto positivo na produtividade na estratégia de *call center*. O artigo torna-se peça importante para comparações futuras com o presente trabalho, que usará técnicas semelhantes em seu desenvolvimento.

2.1.10 Methods for Controlling the Call Placemant Ratio for Outbound Dialing of a Call Center

O artigo de Dilini et al. (2011), do Departamento de Estatística da Universidade de Colombo no Sri Lanka, fala sobre um possível modelo para automatização do parâmetro que indica quantas ligações simultâneas o discador deve realizar, utilizando como base a Cadeia de Markov de Tempo Contínuo (CTMC). O artigo testa três cenários diferentes e aborda os resultados de cada cenário no decorrer do artigo.

O discador preditivo do sistema em estudo funciona como um discador automático, que disca uma chamada assim que é informado que um agente está aguardando uma chamada e que o agente deve esperar até que uma chamada seja atendida fornecendo um tempo muito alto ocioso para o agente enquanto garante o tempo de espera do cliente a zero. O foco principal deste estudo é fornecer previsões usando dados históricos que foram registrados para várias campanhas de saída para fazer a discagem preditiva de forma eficaz. Estratégias de simulação e metodologias de ajuste de distribuição com abordagem analítica têm sido utilizadas para construir o ambiente real que tem a capacidade de prever a rotina de discagem otimizada (DILINI et al., 2011).

Para a simulação, o autor Dilini et al. (2011) determinou, resumidamente, as seguintes fases:

- Fase 1: extrair a média do tempo de estabelecimento da chamada como parâmetro de atraso para haver uma compensação entre a porcentagem de espera do cliente (CWP) e porcentagem de ocupação do agente (ABP);

- Fase 2: descobrir os valores para a ocupação do agente e a taxa de abandono do cliente em relação às mudanças na taxa de discagem e tempo médio de serviço.

Por fim, Dilini et al. (2011), conclui que para que o gerente do sistema consiga ter uma compensação entre a ociosidade do agente e a espera do cliente, precisa seguir diariamente os seguintes passos:

- No início do dia fazer a simulação da fase 1, para todas as horas do mesmo dia de semana anterior;
- Se o sistema funcionar por 10 horas deve-se calcular os 10 valores da mediana de tempo entre as ligações, por meio da simulação feita na etapa 1;
- Fazer a discagem paralela para todos os seis agentes, mantendo o valor correspondente resultante da etapa 2, para cada hora em que a discagem de saída for realizada.

2.2 PILHA TECNOLÓGICA

Nesta seção, serão abordadas todas as tecnologias selecionadas e utilizadas no desenvolvimento do projeto de implementação do algoritmo do fator de aceleração do discador na Conta Azul. Em cada subseção é realizada uma introdução sobre cada tecnologia, para que facilite o entendimento, além de abordar qual será a função da tecnologia e participação no projeto.

2.2.1 Apache Airflow

Apache Airflow é uma ferramenta de código aberto que serve para gerenciar fluxos de extração, transformação e carregamento de dados. Ela foi criada pelo time de desenvolvedores do AirBnB, com o intuito de definir fluxos de trabalho que envolve consulta de dados de inúmeras fontes, tratamento e mineração de dados, de forma que possa ser feito de forma periódica ou não, ou seja, um *pipeline* de dados. Ela se tornou *open source* em meados de 2015, sendo divulgado por um post no *blog* aonde os engenheiros e cientistas de dados da empresa compartilham suas experiências. Depois de algum tempo, ela foi cedida para o Apache em que se tornou um dos inúmeros projetos que ela mantém, hoje chamado de apache-airflow (FILHO, 2018).

Com ajuda do Airflow você pode rápida e facilmente criar *pipeline*s de dados para resolver uma boa parte dos problema de integração e tratamento de dados que existem no dia a dia (BRENNER, 2019).

No projeto, o Apache Airflow será utilizado para gerenciar os *pipeline*s de dados responsáveis por extrair os dados necessários para uso do modelo estatístico, bem como

executar as classes responsáveis pelos cálculos de fator de aceleração ideal e também realizar a inserção do dado já tratado com o fator ideal de aceleração na *API* da ferramenta de telefonia.

2.2.2 Python

Python é uma linguagem de altíssimo nível orientada a objeto, de tipagem dinâmica e forte, interpretada e interativa. O Python tem uma sintaxe clara e concisa que favorece a legibilidade do código-fonte, tornando a linguagem mais produtiva (BORGES, 2014).

A linguagem inclui diversas estruturas de alto nível (listas, dicionários, data/hora, complexos e outras) e uma vasta coleção de módulos prontos para uso, além de *framework*s de terceiros que podem ser adicionados. Também inclui recursos encontrados em outras linguagens modernas, tais como geradores, introspecção, persistência, metaclasses e unidades de teste. Multiparadigma, a linguagem suporta programação modular e funcional, além da orientação a objetos. Mesmo os tipos básicos no Python são objetos. A linguagem é interpretada através de *bytecode* pela máquina virtual Python, tornando o código portável. Com isso é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar direto do código-fonte (BORGES, 2014).

O Python será utilizado no projeto como linguagem de programação base para a criação de todos os *pipeline*s de extração, transformação e carregamento dos dados. Os pipelines escritos em Python serão executados dentro do ambiente do Apache Airflow.

2.2.3 PostgreSQL

PostgreSQL é um sistema de banco de dados relacional de objeto de código aberto que usa e estende a linguagem SQL combinada com muitos recursos que armazenam e escalam com segurança as cargas de trabalho de dados mais complicadas. As origens do PostgreSQL remontam a 1986 como parte do projeto POSTGRES na Universidade da Califórnia em Berkeley e tem mais de 30 anos de desenvolvimento ativo na plataforma central (POSTGRESQL, 1996).

PostgreSQL vem com muitos recursos destinados a ajudar os desenvolvedores a construir aplicativos, administradores para proteger a integridade dos dados e criar ambientes tolerantes a falhas, e ajudá-lo a gerenciar seus dados, não importa o tamanho do conjunto de dados. Além de ser gratuito e de código aberto , o PostgreSQL é altamente extensível (POSTGRESQL, 1996).

No projeto, será necessário a criação de algumas tabelas de dados relacionais, que gravarão dados necessários para cálculo ideal do fator de aceleração, fazendo parte de todo

o *pipeline*. Desta forma, o PostgreSQL será utilizado para armazenar toda essa estrutura de dados relacionais do projeto.

2.2.4 Amazon Web Services

Em 2006, a Amazon Web Services (AWS) começou a oferecer serviços de infraestrutura de TI para empresas por meio de serviços *web* – hoje conhecidos como computação em nuvem. Um dos principais benefícios da computação em nuvem é a oportunidade de substituir diretamente gastos com a infraestrutura principal por preços variáveis baixos, que se ajustam de acordo com sua empresa. Com a Nuvem, as empresas não precisam mais planejar ou adquirir servidores, assim como outras infraestruturas de TI, com semanas ou meses de antecedência. Em vez disso, podem instantaneamente rodar centenas de milhares de servidores em minutos e oferecer resultados mais rapidamente. (AMAZON, 2003).

Segundo Veras, o *cloud computing* ou computação em nuvem pode ser considerada uma nova arquitetura de TI, uma evolução natural dos modelos baseados em *mainframe* e cliente/servidor, ou seja, agora paga-se pelo uso dos recursos em um modelo que se adequa à demanda, trazendo a ideia de redução de custos, minização de riscos e maximização das oportunidades.

Os servidores em nuvem da Amazon Web Services, serão utilizados no projeto para alocação do Apache Airflow, bem como toda a estrutura de banco de dados utilizada no projeto.

2.2.5 Github

Considerado uma rede social para desenvolvedores, o GitHub é uma plataforma de código aberto que possibilita a hospedagem de arquivos versionados, sendo possível consultá-los num outro momento, contribuir com novas versões, sugestões e até mesmo correções de *bug*s. Ele também facilita o trabalho em equipe e de organizações, simplificando as revisões de códigos, as responsabilidades e divisões de trabalho (CHRISTINE, 2019).

Ao hospedar um projeto no Github, possibilitamos que todos os envolvidos ou interessados no projeto tenham um acesso mais fácil e trabalhem de forma centralizada e organizada. Além disso, o github nos concede diversas ferramentas para um melhor controle do projeto como, por exemplo,quais usuários alteraram, o que foi alterando, quando foi alterado, possibilita também que os usuários relatem problemas e muito mais (NSTECH, 2019).

O Github será utilizado no projeto como repositório de todo o código Python gerado, facilitando a contribuição de mais integrantes do time, bem como controle de

versão do projeto.

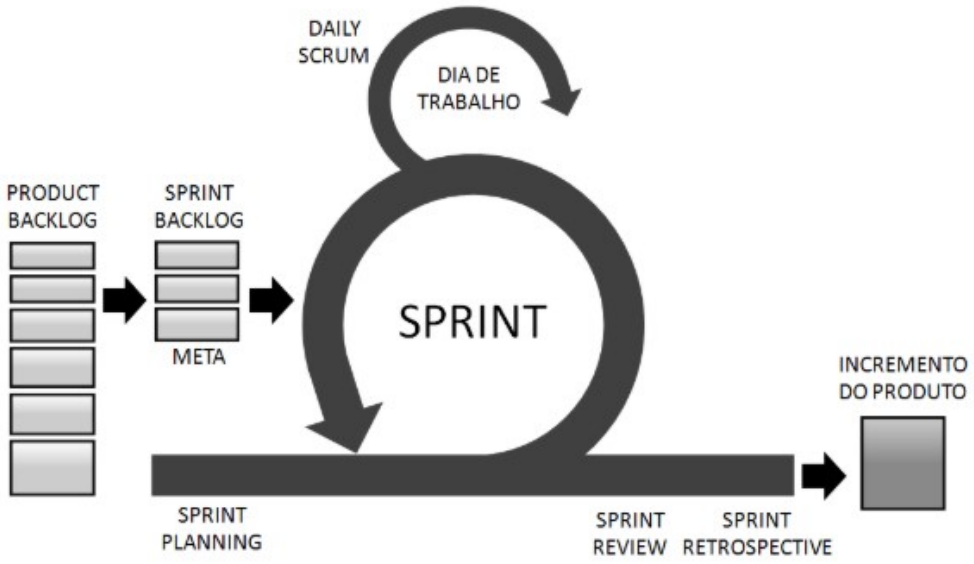
3 PROJETO

Neste capítulo é apresentado como foi estruturado o projeto do presente trabalho, abordando tudo que é necessário e planejado para a execução do mesmo. Através da metodologia *Scrum* e com base nas cerimônias do ciclo de desenvolvimento do *Scrum*, foi possível definir as necessidades, riscos e ações que deverão ocorrer durante o desenvolvimento. Além do *Scrum*, também foram utilizadas técnicas relevantes de engenharia de *software* para entendimento completo do escopo, através dos requisitos, casos de uso e fluxograma.

A melhor forma de ser ágil é construir somente o que o cliente valoriza e não mais que isto. O excesso de formalidade pode limitar o progresso do projeto, mas por outro lado, o caos total, sem a utilização de processos pode impedir que se alcancem os objetivos do projeto. *Scrum* permite criar produtos melhor adaptados à realidade do cliente de forma ágil. Além do mais, praticar *Scrum* nos projetos traz grandes benefícios para a equipe como comprometimento, motivação, colaboração, integração e compartilhamento de conhecimento, o que facilita em muito o gerenciamento e sucesso dos projetos (PEREIRA et al., 2007).

Abaixo a Figura 4 mostra, de forma gráfica, um panorama geral de como funciona um ciclo de desenvolvimento *Scrum*, que ao longo deste capítulo, será desmistifado e aplicado ao presente projeto.

Figura 4 – Ciclo do *Scrum*



Fonte: (OLIVEIRA, 2020)

Este capítulo está dividido em cinco partes. Na seção 3.1 (requisitos), são descritos os requisitos funcionais e não funcionais do projeto. Após, na seção 3.2 (casos de uso) são apresentados, de forma gráfica, os casos de uso referentes ao projeto. Na seção 3.2 (fluxograma), é apresentado o fluxograma do processo de discagem que envolve o presente projeto. Em seguida, na seção 3.3 (escopo), entra-se no escopo do projeto, detalhando atividades e impeditivos do projeto. Na seção 3.4 (execução), é apresentado o ciclo de desenvolvimento para execução do projeto, baseado no ciclo de desenvolvimento do *Scrum*. Por fim, na seção 3.5 (considerações finais), o capítulo é concluído com as considerações finais.

3.1 REQUISITOS

Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado engenharia de requisitos (RE, do inglês *requirements engineering*) (SOMMERVILLE, 2012).

Ainda segundo Sommerville (2012), os requisitos precisam ser escritos em diferentes níveis de detalhamento, para que os leitores possam usá-los de diversas maneiras, leitores de requisitos de usuários não se preocupam com a forma que o sistema será desenvolvido, apenas estão interessados em como ele apoiará os processos de negócio.

Nesta seção, nos tópicos a seguir, serão descritos os requisitos funcionais (3.1.1) e os requisitos não funcionais (3.1.2) do projeto proposto no presente Trabalho de Conclusão de Curso.

3.1.1 Requisitos funcionais

Os requisitos funcionais de um sistema descrevem o que ele deve fazer. Eles dependem do tipo de software a ser desenvolvido, de quem são seus possíveis usuários e da abordagem geral adotada pela organização ao escrever os requisitos. Quando expressos como requisitos de usuário, os requisitos funcionais são normalmente descritos de forma abstrata, para serem compreendidos pelos usuários do sistema (SOMMERVILLE, 2012). Na Tabela 3 abaixo são descritos os requisitos funcionais do projeto.

Tabela 3 – Requisitos funcionais do projeto

Identificador	**Descrição**
RF001	O algoritmo deve atualizar o fator de aceleração do discador a cada 10 minutos.
RF002	O algoritmo deve atualizar o fator de aceleração do discador apenas em horário comercial.
RF003	O algoritmo deve aumentar ou diminuir o fator de aceleração do discador conforme regras e cálculos definidos.
RF004	O algoritmo deve atualizar o fator de aceleração do discador de todas as campanhas relacionadas à equipe de vendas.
RF005	O algoritmo deve guardar dados de cada atualização realizada para acompanhamento de dados e análises.
RF006	O algoritmo deve redefinir o fator de aceleração para o valor mínimo todos os dias no início das operações de discador.

Fonte: O Autor, 2021

3.1.2 Requisitos não funcionais

Os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área. Uma alternativa a esse cenário seria os requisitos definirem restrições sobre a implementação do sistema, como as capacidades dos dispositivos de E/S ou as representações de dados usadas nas interfaces com outros sistemas (SOMMERVILLE, 2012). Na Tabela 4 abaixo, são descritos os requisitos não funcionais do projeto.

Tabela 4 – Requisitos não funcionais do projeto

Identificador	**Descrição**
RNF001	O algoritmo deve gerar alertas em caso de erros durantea execução do fluxo.
RNF002	O algoritmo deve se comunicar através de ambiente seguro com o banco de dados MySQL disponibilizado pelo fornecedor da ferramenta de discador para extração de dados.
RNF003	O algoritmo deve se comunicar através de ambiente seguro com a *API* do fornecedor da ferramenta de discador para inserção do fator.
RNF004	O algoritmo deve ser disponibilizado em ambiente que permita acompanhamento de execuções de forma gráfica e funcional.
RNF005	O algoritmo deve ser disponibilizado em ambiente que permita ligar ou desligar o algoritmo a qualquer momento.

Fonte: O Autor, 2021

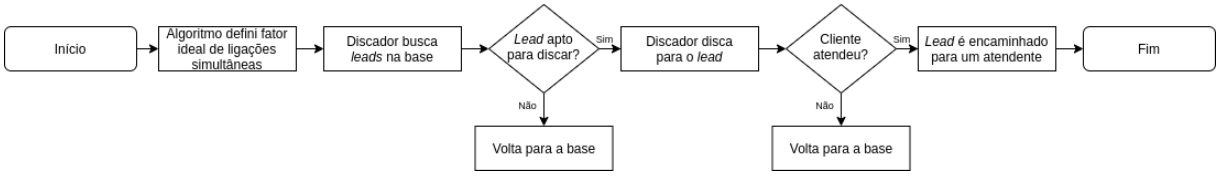
3.2 FLUXOGRAMA

O fluxograma representa um processo que utiliza símbolos gráficos para descrever passo a passo a natureza e o fluxo desse processo. Ele tem como objetivo mostrar, de forma simples e descomplicada, o entendimento do fluxo das informações e elementos evidenciando a sequência operacional que caracteriza o trabalho que está sendo executado tanto na empresa quanto no dia a dia. As etapas do fluxograma são apresentadas utilizando figuras geométricas como círculos, setas, triângulos, etc (MEDEIROS et al., 2018).

Ainda conforme (MEDEIROS et al., 2018), os fluxogramas podem ser utilizados quando se deseja entender o funcionamento de um processo de modo simples ou quando se precisa melhorar a comunicação entre pessoas envolvidas em um mesmo processo.

Na Figura 5 abaixo, é demonstrado o fluxograma do processo de discagem que envolve o presente projeto, sendo o passo de definição do fator ideal de ligações simultâneas proposto no algoritmo do presente projeto.

Figura 5 – Fluxograma do processo de discagem



Fonte: O Autor, 2021

3.3 ESCOPO

O escopo do projeto descreve o trabalho necessário para entregar um produto, um serviço ou um resultado tangível, é o mapeamento de todo o trabalho que será necessário para a conclusão do projeto (ESPINHA, 2021).

Ainda segundo Espinha (2021), o escopo do projeto contém todo o trabalho que é necessário para a realização do projeto, com os detalhes das atividades que serão desempenhadas ao longo do projeto.

Nesta seção, serão descritos os itens referentes ao *Product backlog* (3.3.1), onde são listadas as atividades do projeto e o *Impediment backlog* (3.3.2), onde são listados os impedimentos do projetos. Todos os itens fazem parte do escopo do presente projeto.

3.3.1 *Product Backlog*

O *Product backlog* contém uma lista de itens priorizados que incluem tudo o que precisa ser realizado, que possa ser associado com valor de negócio, para a finalização do projeto, sejam requisitos funcionais ou não. É importante ressaltar que cada item no *Backlog* do produto deve ter um valor de negócio associado (*Business value*), onde podemos medir o retorno do projeto e priorizar a realização dos itens (PEREIRA et al., 2007). Na Tabela 5 abaixo, são descritos os itens do *Product backlog* do projeto.

Tabela 5 – *Product Backlog* do projeto

Identificador	**Descrição**	***Business Value***
PB001	Analisar fatores ideais para automação.	Melhoria métricas discador preditivo.
PB002	Definir regras de cálculo para atualização do fator.	Melhoria métricas discador preditivo.
PB003	Definir arquitetura do projeto.	Melhoria métricas discador preditivo.
PB004	Criar estrutura de banco de dados do projeto.	Melhoria métricas discador preditivo.
PB005	Construir *Airflow Dag* responsável pela extração de dados base da ferramenta para os cálculos.	Melhoria métricas discador preditivo.
PB006	Construir *Airflow Dag* responsável pelos cálculos e definição do fator ideal.	Melhoria métricas discador preditivo.
PB007	Construir *Airflow Dag* responsável pelo envio do fator ideal para a *API* da ferramenta de discador.	Melhoria métricas discador preditivo.
PB008	Construir *Airflow Dag* responsável por redefinir fator mínimo no início da operação diariamente.	Melhoria métricas discador preditivo.

Fonte: O Autor, 2021

3.3.2 *Impediment Backlog*

O *Impediment backlog* contém todos os itens que impedem o progresso do projeto e geralmente estão associados a riscos. Estes itens não possuem uma priorização, mas estão geralmente associados a algum item de *Backlog* do produto ou a tarefas do item (PEREIRA et al., 2007). Na Tabela 6 abaixo, são descritos os itens do *Impediment backlog* do projeto.

Tabela 6 – *Impediment Backlog* do projeto

Identificador	**Descrição**	**Item Associado *Product Backlog***
IB001	Estabelecer conexão com o banco de dados do discador preditivo.	PB005
IB002	Estabelecer conexão com a *API* do discador preditivo.	PB007

Fonte: O Autor, 2021

3.4 EXECUÇÃO

A execução do projeto envolve, portanto, as ações para colocar em prática o que foi planejado no plano de gerenciamento do projeto em seus múltiplos aspectos. E, de forma bem pragmática, são os resultados decorrentes da execução do projeto que, ao final das contas, indicam o sucesso do projeto (FERRARI, 2019).

Segundo Ferrari (2019), é na parte de execução onde se encontram a maioria dos riscos, pois é nessa etapa que os resultados são efetivamente produzidos, portanto é necessário todo o empenho da equipe para realizar o planejado, mantendo-se alertas à necessidade de mudanças.

Nesta seção são abordados aspectos das cerimônias do *Scrum*, que incorporam a execução do presente projeto. Inicia-se abordando a *Sprint* (3.4.1), com o ciclo de desenvolvimento do projeto, passando posteriormente pela *Sprint planning* (3.4.2), onde apresenta-se como é realizado o planejamento do projeto. Também são abordadas particularidades das cerimônias de *Daily scrum* (3.4.3), onde se discute diariamente o andamento do projeto e da *Sprint retrospective* (3.4.4), onde é realizado o fechamento de cada *Sprint*.

3.4.1 *Sprint*

O *Sprint* é o ciclo de desenvolvimento, onde o incremento do produto pronto é gerado pelo time de desenvolvimento a partir dos itens mais importantes do *Product backlog* (SABBAGH, 2014). Até o seu final, o projeto com *Scrum* funciona inteiro dentro de *Sprint*s, que acontecem um atrás do outro, sem paradas ou intervalos. Assim, não se para um *Sprint* após o seu final ou o interrompe por alguns dias para resolver questões relativas ao projeto. Tudo o que acontece em um projeto com *Scrum* é trazido para dentro do *Sprint* (SABBAGH, 2014).

No presente projeto, serão utilizadas *Sprint* de uma semana cada. As *Sprint* serão contínuas e sem pausas, até que o projeto seja finalizado por completo, concluindo todos os itens do *Product backlog* (3.3.1).

3.4.2 *Sprint Planning*

O planejamento de um ciclo de desenvolvimento, ou seja, de um *Sprint*, é realizado na reunião de *Sprint planning*. Ela se inicia logo no primeiro momento do primeiro dia do *Sprint* (SABBAGH, 2014). A reunião de *Sprint planning* conta com a participação obrigatória do *Product owner* e dos membros do time de desenvolvimento. Eles colaboram e negociam o que será desenvolvido e qual a meta desse *Sprint*. Os membros do time de desenvolvimento estabelecem então como o que foi negociado será desenvolvido (SABBAGH, 2014).

No presente projeto, a *Sprint planning* acontecerá no início de cada semana, logo no primeiro dia útil de trabalho, ou seja, todas as segunda-feiras. Na *Sprint planning* do projeto, será discutido quais tarefas do *Product backlog* (3.3.1) serão executadas na semana, de acordo com a prioridade de execução das mesmas. A quantidade de tarefas executadas durante a semana, depende de quanto tempo é exigido para cada execução e tempo disponível do time de desenvolvimento para o projeto na semana.

3.4.3 *Daily Scrum*

A *Daily scrum* é uma reunião curta realizada diariamente pelo time de desenvolvimento. Essa reunião é um período de quinze minutos e acontece preferencialmente no mesmo local e à mesma hora (SABBAGH, 2014). Existe um padrão utilizado por times de desenvolvimento para endereçar essas questões na reunião de *Daily scrum*. Cada membro se dirige a seus colegas e responde a três perguntas: O que eu fiz desde a última reunião de *Daily scrum*? O que eu pretendo fazer até a próxima reunião de *Daily scrum*? Quais obstáculos/impedimentos estiveram/estão em meu caminho, impedindo a realização do trabalho? (SABBAGH, 2014).

No presente projeto, as *Daily scrum* serão realizadas todos os dias úteis da semana na *Sprint* (3.4.1). A cerimônia será de quinze minutos, com o time de desenvolvimento responsável pelo projeto, abordando o andamento das tarefas em execução, priorizadas na *Sprint planning* (3.4.2).

3.4.4 *Sprint Retrospective*

Na reunião de *Sprint retrospective*, o time de *Scrum* inspeciona o *Sprint* que está se encerrando quanto a seus processos de trabalho, dinâmicas, comportamentos, práticas, ferramentas utilizadas e ambiente, e planeja as melhorias necessárias. Facilitados pelo *Scrum master*, time de desenvolvimento e *Product owner* identificam o que foi bem, e que por essa razão pode ser mantido no próximo *Sprint*, e o que se pode melhorar, buscando as causas raízes dos problemas enfrentados no período e traçando planos de ação com formas práticas para se realizarem as melhorias (SABBAGH, 2014).

No presente projeto, a *Sprint retrospective* acontecerá ao fim de cada *Sprint* (3.4.1), ou seja, ao fim de cada semana às sexta-feiras. Durante a cerimônia, serão abordados os itens priorizados na *Sprint planning* (3.4.2) que foram entregues, além de itens que possam estar em atrasos ou com algum impeditivo não previsto. Ao fim da cerimônia, se necessário, são planejados planos de ação para possíveis problemas, abordando lições aprendidas e melhorias para a próxima *Sprint*.

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentada uma visão geral das partes que compõe o presente projeto, seguindo a metodologia *Scrum* combinada à algumas técnicas de engenharia de *software*. Este capítulo será usado como base para o desenvolvimento do projeto.

Acredita-se que a metodologia *Scrum*, com suas particularidades de cerimônias e ciclo de desenvolvimento ágil, contribuirá de forma positiva para que o escopo seja entregue conforme as necessidades destacadas, entregando valor para o negócio em menor tempo e com menos burocracia.

No capítulo 4, na fase de desenvolvimento, o projeto entrará em um contexto mais específico e o modelo geral do projeto será refinado.

4 DESENVOLVIMENTO

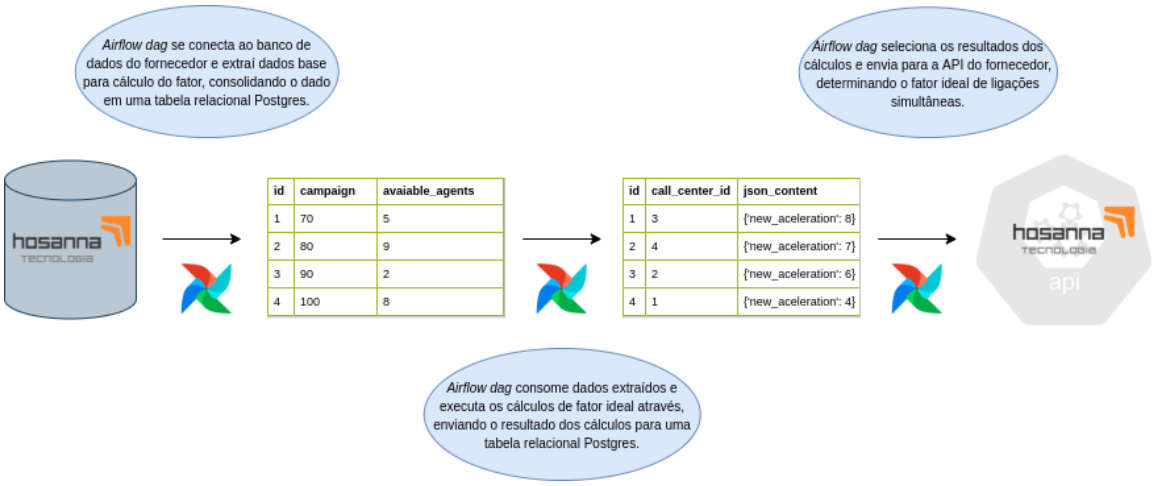
Este capítulo irá mostrar como o algoritmo de cálculo de ligações simultâneas foi desenvolvido do começo ao fim, baseado no projeto proposto (3). Serão abordados os passos que foram sendo definidos e construídos, além do código de cada parte que é desmistificado ao longo do capítulo.

Na seção de arquitetura (4.1), será abordada a arquitetura e aspectos gerais que permeiam o projeto. Na seção de extração dos dados (4.2), é definido como ocorre a extração dos dados necessários para os cálculos. Na seção de cálculo do fator (4.3), são abordadas as regras e cálculos definidos no algoritmo. Adiante, na seção de alteração do fator (4.4), é abordado o processo de envio do fator calculado para o discador. Por fim, na seção de acompanhamento (4.5), é definido como será embasado o acompanhamento do funcionamento do algoritmo.

4.1 ARQUITETURA

Antes de realizar o processo de desenvolvimento de fato, optou-se por realizar a construção de um diagrama, para representar de forma gráfica como o fluxo do algoritmo deveria se comportar, desde à extração dos dados, cálculo e definição do fator ideal. O diagrama é representado na Figura 6 abaixo.

Figura 6 – Arquitetura fluxo algoritmo fator de ligações simultâneas

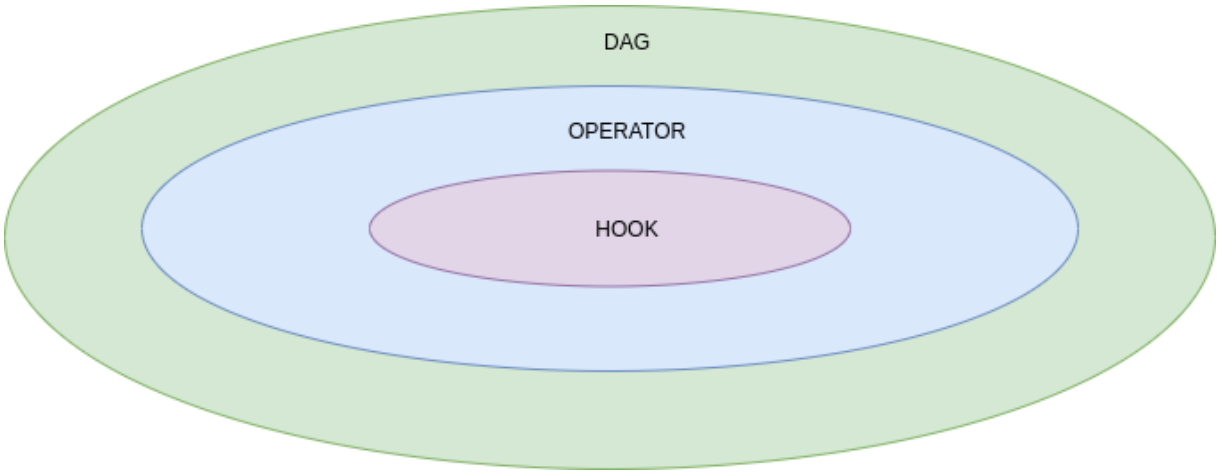


Fonte: O Autor, 2021

Praticamente todo o fluxo é executado pelo Airflow, ferramenta responsável por executar o processo de extração, transformação e carregamento de dados. Cada processo executado no Airflow, pode ser divido em três componentes: *dag* (4.1.1), *operator* (4.1.2)

e *hook* (4.1.3). A Figura 7 abaixo, ilustra como ocorre o encapsulamento de um processo no Airflow.

Figura 7 – Encapsulamento de um processo no Airflow



Fonte: O Autor, 2021

4.1.1 *Dag*

A *dag* é escrita na linguagem Python e é capaz de realizar a execução de processos de consultas e inserção de dados em bancos de dados, através da linguagem *SQL* e também o envio de dados para *APIs*, através da própria linguagem Python, criando processos adicionais que podem ser acionados pela *dag*. Uma *dag* por si só não executa todos os processos, ela apenas tem o objetivo e definir a frequência que aquele determinado processo é executado e assim executar os processos de fato, que são chamados de *operator*s. Desta forma, a *dag* aciona um *operator*, enviando parâmetros necessários para que o processo seja executado.

Para a construção do fluxo serão necessárias quatro *dag*s:

- A primeira *dag* será responsável pelo processo de extração e consolidação dos dados do banco de dados do fornecedor;
- A segunda *dag* será responsável pelo processo de cálculo do fator ideal e consolidação dos dados em uma tabela;
- A terceira *dag* realizará o envio do fator ideal para a *API* do fornecedor;
- A quarta *dag* fará a definição do fator de ligações simultâneas para o valor mínimo, todos os dias no início da operação, no início do dia e após o horário de almoço dos atendentes.

Para realizar a criação de uma *dag*, inicialmente é necessário importar a biblioteca DAG, conforme o bloco de código abaixo:

```
1 from airflow import DAG
```

Para criar a *dag*, também é necessário definir os argumentos padrões, ou *default_args*, que definem algumas informações básicas para a execução da *dag*. Vemos a explicação de cada parâmetro abaixo:

- ***owner*:** recebe um parâmetro no formato de *string* (texto) e indica o nome do responsável pela *dag*;
- ***depends_on_past*:** recebe um parâmetro no formato *booleano* (*true* ou *false*) e indica se a *dag* precisa criar tarefas de execuções no passado para fins de reprocessamento, baseado na data informada no parâmetro *start_date*;
- ***start_date*:** recebe um parâmetro no formato *date* e indica a data em que a *dag* deve começar a criar tarefas para a execução do fluxo, podendo ser um data já passada;
- ***email*:** recebe um parâmetro no formato de *string* (texto) e representa o e-mail do responsável pela *dag*;
- ***email_on_failure*:** recebe um parâmetro no formato *booleano* (*true* ou *false*) e indica se a *dag* deve enviar e-mail informado no parâmetro a *dag*, para o responsável em caso de falhas.

Os parâmetros preenchidos em uma *dag*, ficam escritos conforme o bloco de código exemplo abaixo:

```
1 default_args = {
2     "owner": "nome_teste",
3     "depends_on_past": False,
4     "start_date": datetime(2021, 1, 1),
5     "email": ["teste@teste.com"],
6     "email_on_failure": True,
7 }
```

Além disso, deve ser instanciada uma nova *dag*, informando alguns parâmetros obrigatórios. Abaixo vemos a explicação de cada parâmetro:

- ***catchup*:** recebe um parâmetro no formato *booleano* (*true* ou *false*) e indica se a *dag* deve executar todas as tarefas agendadas (*false*), ou somente a mais recente (*true*), em casos em que a *dag* é pausada e depois ligada novamente;
- ***default_args*:** recebe um parâmetro no formato *dict* (dicionário) e indica os argumentos padrões da *dag*, conforme mencionado anteriormente;

- ***schedule_interval*:** recebe um parâmetro no formato de *string* (texto), deve ser utilizado o padrão de agendamento de tarefas *cron* do Linux e indica a frequência de execuções das tarefas da *dag*;
- ***max_active_runs*:** recebe um parâmetro no formato *int* (inteiro) e indica qual o número máximo de execuções paralelas que a *dag* pode realizar.

Os parâmetros no código da *dag*, ficam conforme o bloco de código exemplo abaixo;

```
dag = DAG(
    "teste_dag",
    catchup=False,
    default_args=default_args,
    schedule_interval="*/5 * * * *",
    max_active_runs=1,
)
```

Opcionalmente, ainda pode ser definido o parâmetro *doc_md*, que tem como objetivo exibir o nome da *dag* no painel gráfico do Airflow. No código da *dag*, ele é declarado conforme o bloco de código abaixo:

```
dag.doc_md = __doc__
```

A partir da definição dos parâmetros citados, é possível criar uma nova *dag*, o que irá diferenciá-las a partir daí serão os *operator*s (4.1.2), onde cada *operator* pode executar uma ação diferente, podendo ser implementados nas *dag*s.

4.1.2 *Operator*

O operator é escrito na linguagem Python e é encapsulado pela *dag* e geralmente é genérico, podendo ser utilizado em mais de uma *dag*. O *operator* geralmente recebe dois parâmetros: o primeiro indicando de onde o dado vai ser extraído, podendo ser uma consulta *SQL* ou uma consulta em determinada *API* e o segundo parâmetro indica onde o dado será inserido, podendo ser uma inserção *SQL* ou uma inserção via *API*. Quando acionado por uma *dag* e recebendo os parâmetros necessários, o *operator* então executa o processo de extração e envio de dados. Além disso, existe um *operator* específico que pode simplesmente acionar outra *dag*, encadeando vários fluxos.

O fluxo dependerá de quatro *operator*s:

- O primeiro *operator* será encapsulado pela *dag* responsável por extrair o dado do banco de dados do fornecedor e consolidar o dado em uma tabela, sendo responsável por executar uma consulta *SQL* no banco de dados MySQL do fornecedor e inserir os dados no banco de dados PostgreSQL da companhia;

- O segundo *operator* será encapsulado pela *dag* responsável por calcular o fator ideal de ligações simultâneas e pela *dag* que fará a definição do fator para o valor mínimo no início da operação, sendo responsável por executar uma consulta *SQL* no banco de dados PostgreSQL da companhia e inserir em uma segunda tabela, também no banco de dados PostgreSQL da companhia;

- O terceiro *operator* será encapsulado pela *dag* responsável por enviar o fator ideal para a *API* do fornecedor, sendo responsável por executar uma consulta *SQL* no banco de dados PostgreSQL da companhia e envio dos dados para a *API* do fornecedor;

- O quarto *operator*, que será escapsulado pelas *dag*s para encadear o processo e definir a sequência de execução das *dag*s, ou seja, a primeira *dag* irá executar e utilizar o *operator* para acionar a próxima *dag* a ser executada, até a execução do fluxo completo.

Os *operator*s podem ser facilmente encontrados na comunidade, ou até podem ser criados novos *operator*s personalizados para utilização específica. Para a utilização, os *operator*s são importados nas *dag*s, conforme o código exemplo abaixo que importa o *operator* *DummyOperator*:

```
1 from airflow.operators.dummy_operator import DummyOperator
```

Para funcionamento, os *operator*s dependem dos *hook*s (4.1.3), que realizam a conexão com as fontes de dados e *APIs*.

4.1.3 *Hook*

O *hook* é escrito na linguagem Python e é encapsulado pelo *operator*, tendo um único objetivo: realizar a conexão e autenticação com as ferramentas necessárias, para seleção e envio de dados. Desta forma, o *hook* pode realizar a abertura de conexão com bancos de dados e comunicação com *APIs*, que serão utilizadas no *operator*.

No processo, serão utilizados três *hook*s:

- O primeiro *hook* será responsável por abrir conexão com o banco de dados MySQL do fornecedor;

- O segundo *hook* será responsável por abrir conexão com o banco de dados PostgreSQL da companhia;

- O terceiro *hook* será responsável por realizar a autenticação e comunicação com a *API* do fornecedor.

Os *hook*s, assim como os *operator*, são encontrados com certa facilidade na comunidade, mas também podem ser personalizados, de acordo com cada utilização. Os *hook*s são importados nos *operator*s, conforme o código exemplo abaixo que importa o *hook* *PostgresHook*:

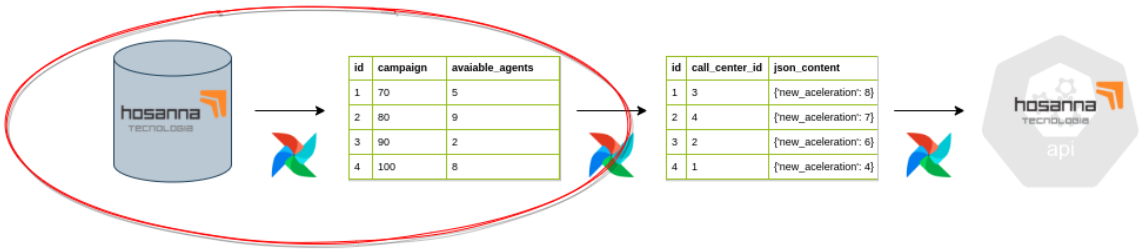
```
1 from airflow.hooks.postgres_hook import PostgresHook
```

A partir da importação, os *hook*s então recebem os parâmetros necessários para cada conexão e se encarregam de realizar o processo de conexão com as fontes de dados e *APIs*.

4.2 EXTRAÇÃO DOS DADOS

O processo de extração de dados é o primeiro passo para a execução do fluxo, conforme o diagrama da Figura 8 abaixo e fluxo apresentado na seção de arquitetura (4.1). O processo consiste em selecionar alguns dados da operação do discador, no momento em que o fluxo começa a ser executado, inserindo estes dados em uma tabela do banco de dados da companhia, para que estes dados sirvam como base para posterior cálculo do novo fator ideal de ligações simultâneas, representando a realidade atual da operação.

Figura 8 – Processo de extração dos dados



Fonte: O Autor, 2021

Para a extração dos dados, o fornecedor não fornece *API*, mas sim uma réplica MySQL do banco de dados de produção da aplicação. Para extrair os dados, foi necessário acionamento ao fornecedor para entendimento das tabelas que armazenavam as informações, pois a estrutura não estava documentada, dificultando o entendimento.

Com devido acesso ao banco de dados da aplicação de discador, inicialmente foi levantado quais informações seriam necessárias para a extração, sendo:

- **ID da campanha**: representa o ID identificador da campanha cadastrada dentro da ferramenta de discador, necessária para envio do novo fator ideal à *API* do discador posteriormente;

- **Nome da campanha**: representa o nome da campanha cadastrada dentro da ferramenta de discador, não necessária para execução do fluxo, mas serve como dado informativo para acompanhamento, facilitando o entendimento posteriormente;
- **Fator de ligações simultâneas atual**: representa o fator de ligações simultâneas que a campanha está utilizando naquele momento, necessário para cálculo do novo fator ideal;
- **Total de atendentes *online***: representa o número total de atendentes que estão alocados na campanha naquele momento e estão *online* na ferramenta, necessário para cálculo do novo fator ideal;
- **Total de atendentes disponíveis**: representa o número total de atendentes que estão disponíveis e aptos para recebimento de ligações na campanha naquele momento, ou seja, não estão em atendimento, necessário para cálculo do novo fator ideal.

De acordo com a definição de quais dados eram necessários para a construção do fluxo e com devido entendimento das tabelas do fornecedor, foi possível montar uma consulta na liguagem *SQL*, que selecionava os dados no banco de dados MySQL do fornecedor. A consulta *SQL* ficou formatada conforme o código abaixo:

```
1  select
2      tabela_campanha.id as id_campanha,
3      tabela_campanha.nome as nome_campanha,
4      tabela_campanha.fator_atual as fator_atual,
5      count(tabela_atendimentos.id) as total_atendentes,
6      sum(if(disponivel is true, 1, 0)) as total_atendentes_disponiveis
7  from tabela_campanha
8  join tabela_atendimentos
9      on tabela_atendimentos.campanha_id = tabela_campanha.id
10 where tabela_campanha.ativa is true
11 group by 1, 2, 3
```

A partir da montagem da consulta *SQL*, que selecionava os dados necessários para o cálculo, foi possível iniciar a construção da primeira *dag* Airflow que iniciava o fluxo. Essa *dag* é responsável por executar a consulta *SQL*, obtendo os dados necessários no banco de dados MySQL do fornecedor e inserindo em um banco de dados PostgreSQL da companhia, consolidando os dados necessários em um lugar único, para que sejam posteriormente cruzados com outros dados operacionais localizados no mesmo banco de dados da companhia e realizado o devido cálculo de fator ideal.

A *dag* utiliza o *operator MysqlToPostgresOperator*, que realiza a seleção de dados em um banco de dados MySQL de acordo com uma consulta *SQL* recebida e insere em um banco de dados PostgreSQL de acordo com uma inserção *SQL* recebida. O *operator* pode ser utilizado realizando a importação no código Python da *dag*, conforme abaixo:

```
1 from lib.operators.mysql_to_postgres_operator import
    MysqlToPostgresOperator
```

O *operator MysqlToPostgresOperator* por sua vez, utiliza durante a execução dois *hook*s: um *hook* que abre conexão com o banco de dados MySQL do fornecedor e outro *hook* que abre conexão com o banco de dados PostgreSQL da companhia.

Para realizar a extração dos dados no banco de dados MySQL e inserção dos dados no banco PostgreSQL, o *operator MysqlToPostgresOperator* necessita receber os seguintes parâmetros:

- ***source_conn_id*:** recebe um parâmetro no formato *string* (texto) e indica qual o nome da conexão cadastrada no Airflow, com os devidos parâmetros para a conexão, que será utilizada para consultar os dados;
- ***source_sql*:** recebe um parâmetro no formato *string* (texto) e indica qual consulta *SQL* deve ser utilizada para selecionar os dados necessários;
- ***destination_conn_id*:** recebe um parâmetro no formato *string* (texto) e indica qual o nome da conexão cadastrada no Airflow, com os devidos parâmetros para a conexão, que será utilizada para inserir os dados selecionados;
- ***destination_sql*:** recebe um parâmetro no formato *string* (texto) e indica qual inserção *SQL* deve ser utilizada para inserir os dados necessários.

Além do *operator MysqlToPostgresOperator*, a *dag* utiliza o operator *TriggerDagRunOperator*, que tem como objetivo acionar a próxima *dag* do fluxo, para que o fluxo seja executado em sequência. Este *operator* recebe somente o parâmetro *trigger_dag_id*, que indica o nome da *dag* que deve ser acionada em seguida. O *operator* pode ser utilizado realizando a importação no código Python da *dag*, conforme abaixo:

```
1 from airflow.operators.dagrun_operator import TriggerDagRunOperator
```

No código da *dag* também foi necessário a utilização da biblioteca *datetime*, para formatar a data a ser passada no parâmetro *start_date* da *dag*, que defini a data em que o processo inicia-se, conforme mencionado anteriormente, sendo utilizado importado conforme abaixo:

```
1 from datetime import datetime
```

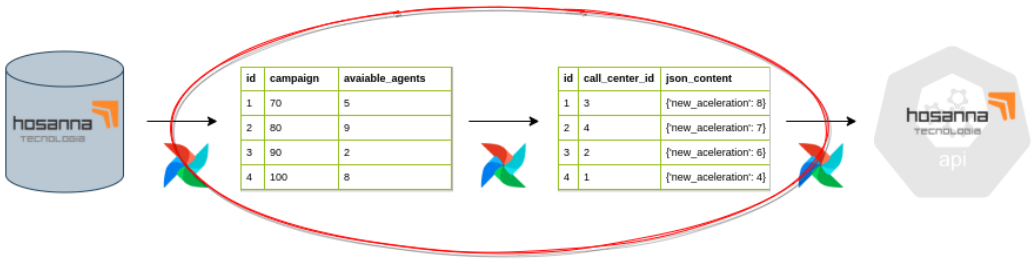
Com o entendimento necessário dos *operator*s, bibliotecas e parâmetros a serem utilizados, foi construída a primeira *dag*, responsável por selecionar os dados no banco de dados MySQL do fornecedor e inserí-los no banco de dados PostgreSQL da companhia. O código final, ficou definido conforme o ANEXO A.

Após a execução da *dag* citada acima, é realizado o acionamento da próxima da *dag* do fluxo, então inicia-se o processo de cálculo do fator ideal de ligações simultâneas (4.3).

4.3 CÁLCULO DO FATOR

O processo de cálculo do fator é o segundo passo do fluxo, conforme o diagrama da Figura 9 abaixo e fluxo apresentado na seção de arquitetura (4.1). O processo consiste em selecionar os dados do discador, obtidos no processo de extração dos dados (4.2) e com base nos dados, calcular o fator ideal de ligações simultâneas para o discador, inserindo esses dados em uma segunda tabela do banco de dados da companhia, já com os devidos cálculos realizados, formatado conforme o padrão exigido pela *API* do fornecedor e pronto para ser enviado ao discador.

Figura 9 – Processo de cálculo do fator



Fonte: O Autor, 2021

Para definição de como deveria funcionar o cálculo, foi inicialmente realizada uma análise com base em alguns números acompanhados internamente pelos gestores dos times de atendimento. O processo de análise tinha como objetivo entender qual era a média ideal dos fatores e quais valores eram considerados ideais para cada métrica. A análise se baseou em três fatores:

- **Percentual de contato não falado:** varia de 0% a 100% e indica qual o percentual dos *lead*s importados no discador ainda não atenderam as ligações disparadas pelo discador, o dado é obtido no banco de dados da companhia e é atualizado a cada dez minutos;
- **Percentual de agentes disponíveis:** varia de 0% a 100% e indica qual o percentual dos atendentes da operação está disponível para recebimento de ligações,

considerado período improdutivo pelo negócio, o dado é obtido no banco de dados da companhia através do processo de extração dos dados (4.2), onde é realizado um cálculo entre o total de atendentes disponíveis e o total de atendentes *online*, chegando no percentual necessário e é atualizado a cada dez minutos;

- **Percentual de *drop*:** varia de 0% a 100% e indica qual o percentual das ligações foram atendidas pelos *lead*s importados no discador e foram posteriormente abandonadas, por falta de atendentes disponíveis para atendimento, considerado péssima experiência ao usuário pelo negócio, o dado é obtido no banco de dados da companhia e é atualizado a cada dez minutos.

Com base nas métricas acompanhadas pelos gestores dos times de atendimento, chegou-se em uma conclusão com relação aos fatores ideais que o algoritmo deveria respeitar acima de tudo, permitindo chegar em uma definição de regras para incremento e decremento do fator de ligações simultâneas do discador, considerados os seguintes gatilhos:

- **Gatilho 1 - Percentual de contato não falado:** o gatilho de incremento se dá quando o taxa atinge um valor superior a 60% e o decremento se dá quando o valor atinge valor inferior a 40%;
- **Gatilho 2 - Percentual de agentes disponíveis:** o gatilho de incremento se dá quando o taxa atinge um valor superior a 60% e o decremento se dá quando o valor atinge valor inferior a 40%;
- **Gatilho 3 - Percentual de *drop*:** somente para gatilho de decremento, quando percentual atinge 5% ou mais, independente dos valores dos percentuais anteriores, prezando pela experiência dos clientes.

O discador permite a realização de ligações simultâneas para 1 até 10 *lead*s para cada agente disponível. Desta forma, o algoritmo deveria realizar os cálculos necessários e com base nos gatilhos pré definidos, aumentar ou diminuir o fator em 1 unidade de ligações simultâneas, até que o fator se estabeleça ideal e respeitando os gatilhos definidos.

Com base nas regras definidas, foi possível iniciar a construção da *dag* responsável pela realização do cálculo. Para a construção, foi utilizado o *operator PostgresUpsertOperator*, que realiza a seleção dos dados em uma tabela do banco de dados PostgreSQL da companhia e insere em uma segunda tabela do banco de dados PostgreSQL da companhia, já com os valores do cálculo realizados. O *operator* pode ser utilizado realizando a importação no código Python da *dag*, conforme abaixo:

```
1 from lib.operators.postgres_upsert_operator import
    PostgresUpsertOperator
```

O *operator PostgresUpsertOperator*, utiliza em sua execução, um *hook* que é responsável por abrir a conexão com o banco de dados PostreSQL da companhia.

Para realizar a seleção dos dados e inserção do fator ideal calculado, o *operator PostgresUpsertOperator* necessita receber os seguintes parâmetros:

- ***source_conn_id*:** recebe um parâmetro no formato *string* (texto) e indica qual o nome da conexão cadastrada no Airflow, com os devidos parâmetros para a conexão, que será utilizada para consultar os dados;
- ***source_sql*:** recebe um parâmetro no formato *string* (texto) e indica qual consulta *SQL* deve ser utilizada para selecionar os dados necessários;
- ***destination_conn_id*:** recebe um parâmetro no formato *string* (texto) e indica qual o nome da conexão cadastrada no Airflow, com os devidos parâmetros para a conexão, que será utilizada para inserir os dados selecionados;
- ***destination_sql*:** recebe um parâmetro no formato *string* (texto) e indica qual inserção *SQL* deve ser utilizada para inserir os dados necessários.

Foi definido que o cálculo do fator ideal, poderia ser realizado pela própria consulta *SQL*, que é enviada no parâmetro *source_sql* do *operator PostgresUpsertOperator*, ou seja, no momento da seleção dos dados dos gatilhos para definição do fator, é realizado os cálculos que definem se o algoritmo deve aumentar ou diminuir o fator. A consulta *SQL*, com os cálculos implementados, ficou conforme o bloco de código abaixo:

```
new_acceleration as(
    select
        call_center_log.id,
        call_center_log.campaign_id,
        call_center_log.total_agents,
        call_center_log.avaiable_agents,
        case
            when dialer_contacts.dropped_percent_contacts >= 5
                then call_center_log.current_acceleration_factor - 1
            when round(
                    (
                        call_center_log.avaiable_agents_percentage +
                        dialer_contacts.not_speak_percent_contacts
                    )/2, 1) > 6
                then call_center_log.current_acceleration_factor + 1
            when round(
```

```
                    (
                        call_center_log.avaiable_agents_percentage +
                        dialer_contacts.not_speak_percent_contacts
                    )/2, 1) < 4
                then call_center_log.current_acceleration_factor - 1
            else call_center_log.current_acceleration_factor
        end as acceleration
    from call_center_log
    join dialer_contacts
        on dialer_contacts.campaign_id = call_center_log.campaign_id
)
select
    id as call_center_log_id,
    campaign_id,
    json_build_object (
        'campaignid', campaign_id,
        'acceleration', acceleration,
        'channels', ceil(acceleration * total_agents)
    )::text as json_content
from new_acceleration
where avaiable_agents > 0
and acceleration between 1 and 10
```

Assim como no processo de extração dos dados (4.2), foi necessária a utilização do *operator TriggerDagRunOperator* para disparo da próxima *dag* do fluxo e da biblioteca *datetime* para formatar a data a ser enviada no parâmetro *start_date* da *dag*:

```
from airflow.operators.dagrun_operator import TriggerDagRunOperator
from datetime import datetime
```

Com as definições necessárias, foi possível enfim, construir a segunda *dag* do fluxo, responsável por selecionar os dados necessários no banco de dados PostgreSQL da companhia, realizar o cálculo do fator ideal e inserir na tabela *dialer.acceleration_sync* do banco de dados PostgreSQL da companhia, para posterior envio ao discador. Vale ressaltar que essa *dag* recebe o parâmetro *schedule_interval=None*, ou seja, nunca é disparada por si só, pois é acionada pela *dag* que realiza o processo de extração dos dados (4.2). Além disso, foi implementada uma lógica no momento da inserção, para que os registros fiquem com a coluna *sync_status* com o valor 'WAITING', ou seja, aguardando envio para a *API* do fornecedor, isso permite visualizar se o processo está ocorrendo conforme esperado no banco de dados, pois quando o dado é enviado para a *API* do fornecedor de fato, o *sync_status* é alterado de *'WAITING'* para *'DONE'*. O código final, ficou definido

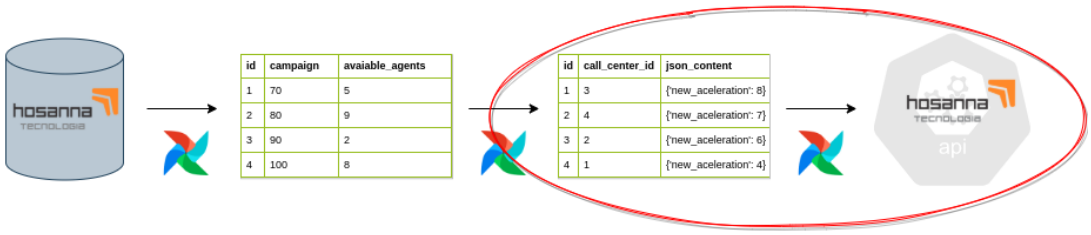
conforme o ANEXO B.

Após a execução da *dag* citada acima, então é realizado o acionamento da próxima *dag* do fluxo, é iniciado o processo de alteração do fator ideal de ligações simultâneas (4.4).

4.4 ALTERAÇÃO DO FATOR

O processo de alteração do fator é o terceiro e último passo do fluxo, conforme o diagrama da Figura 10 abaixo e fluxo apresentado na seção de arquitetura (4.1). O processo tem como objetivo selecionar os dados gerados pelo cálculo do fator ideal de ligações simultâneas (4.3), que já estão formatados conforme o padrão exigido pela *API* do fornecedor e com novo valor de fator ideal de ligações simultâneas definido e, enviar para *API* do fornecedor, alterando o fator no discador e refletindo na operação de atendimento.

Figura 10 – Processo de alteração do fator



Fonte: O Autor, 2021

Para realizar o processo de enviar o novo fator de ligações simultâneas para o discador, foi necessário utilizar o *operator PostgresToHttp*, que realiza a seleção dos dados na tabela *dialer.acceleration_sync* do banco de dados PostgreSQL da companhia e envia este dado obtido para *API* do discador, no formato *json* exigido pela *API*. O *operator PostgresToHttp* pode ser utilizado realizando a importação no código Python da *dag*, conforme abaixo:

```
1 from lib.operators.postgres_to_http import PostgresToHttp
```

O *operator PostgresToHttp*, utiliza em sua execução dois *hooks*: um *hook* responsável por realizar a conexão com o banco de dados PostgreSQL da companhia e outro *hook* responsável por realizar a conexão com a *API http* do fornecedor.

Para que o *operator PostgresToHttp* funcione conforme esperado, é necessário enviar alguns parâmetros em sua utilização:

- ***source_conn_id*:** recebe um parâmetro no formato *string* (texto) e indica qual o

nome da conexão cadastrada no Airflow, com os devidos parâmetros para a conexão, que será utilizada para consultar os dados;

- ***source_sql*:** recebe um parâmetro no formato *string* (texto) e indica qual consulta *SQL* deve ser utilizada para selecionar os dados necessários;
- ***http_conn_id*:** recebe um parâmetro no formato *string* (texto) e indica qual o nome da conexão de *API* cadastrada no Airflow, com os devidos parâmetros como *URL* e *token* de acesso para a *API*, que será utilizada para enviar o dado;
- ***endpoint*:** recebe um parâmetro no formato *string* (texto) e indica qual o caminho da *URL* (*path*) da *API* os dados devem ser enviados;
- ***headers*:** recebe um parâmetro no formato *string* (texto) e indica quais informações são necessárias para utilização no cabeçalho da *API*.

Tendo conhecimento dos parâmetros necessários, foi preciso primeiramente criar a consulta *SQL* que realiza a consulta na tabela *dialer.acceleration_sync* do banco de dados PostgreSQL da companhia, para capturar o novo fator de ligações simultâneas a ser enviado para a *API* do discador. Então foi construída a seguinte consulta *SQL*, a ser enviada no parâmetro *source_sql* do *operator PostgresToHttp*, para realizar a captura do dado, conforme o bloco de código abaixo:

```
select
    json_content as json_data
from dialer.acceleration_sync
where sync_status = 'WAITING'
and created_at >= current_date
group by 1, 2, 3
```

Assim como nos processos anteriores de extração dos dados (4.2) e cálculo do fator (4.3), foi necessária a utilização da biblioteca *datetime* para formatar a data a ser enviada no parâmetro *start_date* da *dag*:

```
from datetime import datetime
```

Ainda foi preciso, criar um *update SQL* na *dag*, para realizar a alteração do *sync_status* do registro inserido como *'WAITING'* no processo de cálculo do fator (4.3) para *'DONE'*, indicando que o registro já foi enviado para a *API* do discador. O *SQL* ficou conforme o bloco de código abaixo:

```
    update dialer.acceleration_sync
        set sync_status = 'DONE',
        sync_at = now()
```

```
4     where sync_status = 'WAITING'
5     and created_at >= current_date
```

Para realizar a execução do *update SQL* acima, foi preciso utilizar o *operator PostgresOperator*, que tinha como único objetivo realizar o *update* na tabela. O *operator* foi importado no código Python da *dag*, conforme o bloco de código abaixo:

```
1 from airflow.operators.postgres_operator import PostgresOperator
```

O *operator PostgresOperator* utiliza somente um *hook*, para se conectar ao banco de dados PostgreSQL da companhia.

Para que o *operator PostgresOperator* funcione conforme esperado, é necessário enviar dois parâmetros em sua utilização:

- ***postgres_conn_id*:** recebe um parâmetro no formato *string* (texto) e indica qual o nome da conexão cadastrada no Airflow, com os devidos parâmetros para a conexão, que será utilizada para execução do *script SQL*;
- ***SQL*:** recebe um parâmetro no formato *string* (texto) e indica qual *script SQL* deve ser executado no banco de dados.

Com todas as definições realizadas, foi possível então, construir a terceira e última *dag* do fluxo, responsável por selecionar o fator calculado no banco de dados PostgreSQL da companhia e realizar o envio para a *API* do fornecedor. Vale ressaltar que essa *dag* recebe o parâmetro *schedule_interval=None*, ou seja, nunca é disparada por si só, pois é acionada pela *dag* que realiza o processo de cálculo do fator (4.3). O código final da *dag*, ficou definido conforme o ANEXO C.

Para cumprir o requisito funcional RF006, definido no tópico de requisitos funcionais do projeto (3.1.1), onde é definido que o algoritmo deve redefinir o fator de aceleração para o valor mínimo todos os dias no início das operações de discador, foi necessária a construção de uma *dag* específica para essa redefinição. A redefinição é necessária, pois geralmente, no fim do período de operação o fator está em um valor alto, pois aos fins das manhãs e tardes é onde as ligações são atendidas com maior frequência e o fator tem maior êxito. No início do dia, e após o horário de almoço dos atendentes, para que as primeiras discagens não ocorram com o valor alto, é redefinido o valor para o valor mínimo (1), assim garante-se que o percentual de abandono não fique alto nesses períodos, respeitando a experiência do *lead*.

Para que o processo de redefinição fosse realizado, foi utilizado o *operator PostgresUpsertOperator*, assim como no processo de cálculo do fator (4.3), gerando um registro na tabela *dialer.acceleration_sync* que altera o fator para 1. A *dag* é disparada de forma

isolada ao fluxo, todos os dias úteis no início da manhã e da tarde. O código final da *dag* ficou definido conforme o ANEXO D.

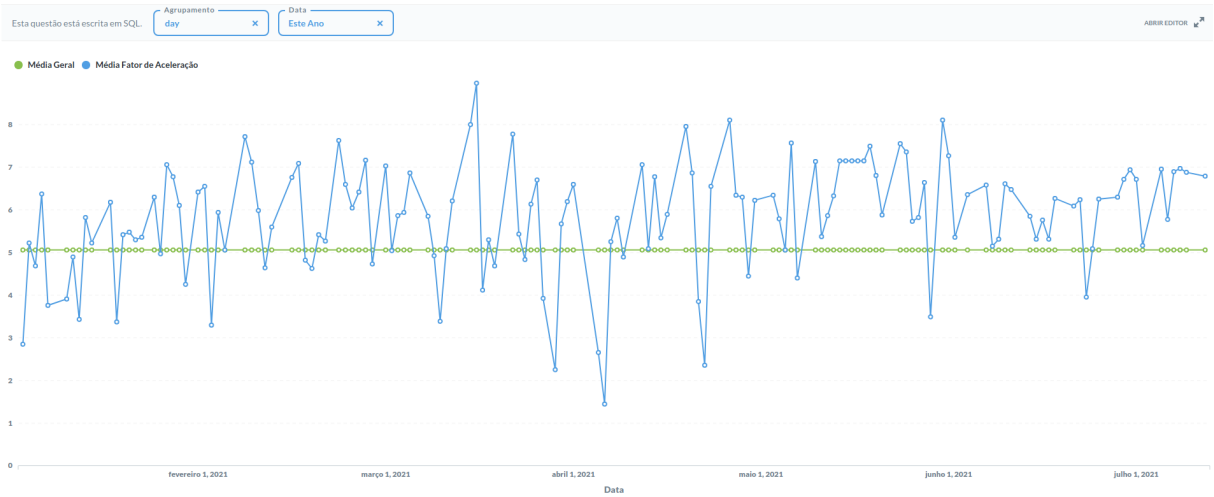
O fluxo então é concluído e permanece sendo disparado por completo a cada dez minutos, realizando uma nova alteração no fator de ligações simultâneas, sempre respeitando os gatilhos definidos e com base em dados atualizados, sem a necessidade de ação humana, permanecendo automatizado por completo.

4.5 ACOMPANHAMENTO

Após a finalização do projeto, os dados que são salvos no banco de dados da companhia, ainda puderam ser utilizados para montar uma visão de acompanhamento do algoritmo. A visão foi construída a partir da linguagem de consulta *SQL*, que realiza a consulta ao dado no banco de dados PostgreSQL da companhia, através da ferramenta de visualização Metabase, utilizada na companhia. Com isso, foi possível criar uma visão de acompanhamento das alterações de fator realizadas pelo algoritmo.

O gráfico construído, tem como objetivo, mostrar a média móvel do fator de ligações simultâneas por hora, dia, mês, trimestre ou ano e a média geral do fator, ambos atualizados em tempo real, conforme o funcionamento do algoritmo. Também é possível filtrar períodos específicos, alterando os filtros do gráfico. O gráfico ficou conforme a Figura 11 abaixo:

Figura 11 – Gráfico média fator de ligações simultâneas



Fonte: O Autor, 2021

Para a construção do gráfico, foi construída uma consulta *SQL*, que foi inserida na ferramenta de visualização Metabase da companhia e, posteriormente realizada a plotagem do gráfico. A consulta *SQL*, ficou conforme o bloco de código abaixo:

```
1 select
```

```
2     date_trunc({{agrupamento}}, call_center_log.created_at) as "Data",
3     avg(call_center_log.current_acceleration_factor) as "Media Fator
          de Aceleracao",
4     (select avg(call_center_log.current_acceleration_factor)from
          dialer.call_center_log join dialer.acceleration_sync on
          acceleration_sync.campaign_id = call_center_log.campaign_id) as
           "Media Geral"
5 from dialer.call_center_log
6 join dialer.acceleration_sync
7     on acceleration_sync.campaign_id = call_center_log.campaign_id
8 where {{data}}
9 group by 1
10 order by 1
```

Por fim, a visão então foi compartilhada com os interessados do negócio para acompanhamento do funcionamento do algoritmo, comprovando o funcionamento do projeto proposto no presente trabalho.

5 RESULTADOS

Este capítulo tem como objetivo demonstrar os resultados obtidos durante o projeto e principalmente após a finalização do seu desenvolvimento (4). Na seção de conhecimentos adquiridos (5.1), serão abordados os conhecimentos que foram absorvidos durante a execução do projeto. Na seção de produto desenvolvido (5.2), será exibido o produto final que foi construído ao longo deste projeto. Por fim, na seção (5.3), será demonstrado um comparativo dos indicadores acompanhados, pré e pós implementação do algoritmo.

5.1 CONHECIMENTOS ADQUIRIDOS

Ao longo deste Trabalho de Conclusão, foi possível compreender e implementar as técnicas aprendidas no decorrer do curso, além da absorção de técnicas adicionais específicas da área de dados, que é um tema em alta nas empresas atualmente. Com a execução deste projeto, foi possível participar desde o início da construção de um produto de fato, desde o levantamento de requisitos com os usuários, discussão de escopo, priorização, aplicação de metodologias ágeis, até a execução do plano em si e entrega do produto desenvolvido. O projeto também permitiu demonstrar à empresa, o quanto a área de dados é importante para aplicação de inteligência no ambiente empresarial. A comunicação direta com os interessados, definindo todas as etapas juntos, foi importantíssima para o alinhamento do que estava sendo construído e qual seria o produto final apresentado.

O uso de várias tecnologias que já estavam implementadas na companhia, como bibliotecas e ferramentas, facilitou o processo de construção, pois o time já estava capacitado para apoio imediato em caso de problemas e, todo o padrão de desenvolvimento já estava definido, seguindo o que era estabelecido pela companhia e viabilizando o projeto. A utilização de ferramentas internas que já eram utilizadas para outros fins, como o Apache Airflow, Github, PostgreSQL e Amazon Web Services, viabilizaram o projeto, pois não tinha impacto significativo nos custos da companhia, visto que não era necessário a contratação de nenhuma outra ferramenta adicional.

O fato desse tipo de conteúdo, relacionado à aplicação de tecnologia em *call cente*rs brasileiros, ter pouca visibilidade no país, tornou o projeto um dos pioneiros na área, mostrando como os dados podem ser utilizados para geração de inteligência para o setor de comunicação. O projeto também proporcinou a busca do Autor por conhecimentos e soluções de problemas durante o processo de desenvolvimento, permitindo a absorção de conteúdo na prática e, constribuindo para o crescimento pessoal e profissional.

5.2 PRODUTO DESENVOLVIDO

O produto desenvolvido neste Trabalho de Conclusão de Curso, tinha como objetivo a construção de um algoritmo que automatizasse com base em dados, o processo de atualização do fator de ligações simultâneas do discador da empresa Conta Azul, de forma que o processo fosse autônomo e o mais próximo possível do ideal.

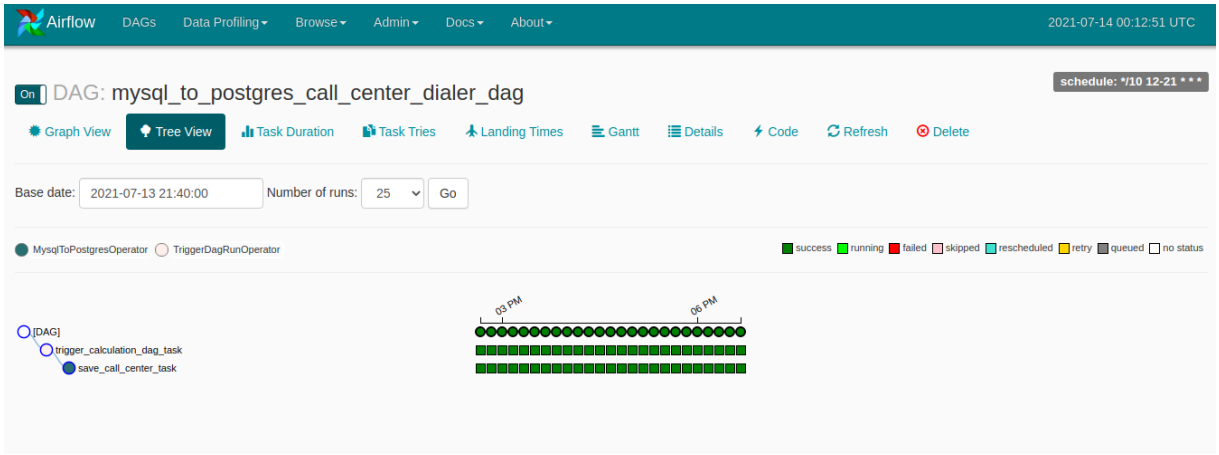
A primeira versão do produto construído, cumpriu todos os requisitos definidos e cumpriu o objetivo geral definido de forma concreta, automatizando o processo do início ao fim, sem a necessidade de ação humana durante o fluxo, que é executado com autonomia.

Foram construídas quatro *dag*s, abordadas no capítulo de desenvolvimento (4), que são responsáveis pelos processos de extração dos dados necessários para o cálculo, cálculo do fator ideal de ligações simultâneas para aquele determinado momento do dia, atualização do fator ideal do discador da companhia e redefinição do fator no início das operações de discador.

Na ferramenta Airflow, utilizada para o agendamento e controle das execuções do fluxo, é possível monitorar as execuções das *dag*s construídas no fluxo de forma visual, conforme as imagens a seguir.

- ***mysql_to_postgres_call_center_dialer_dag*:** *dag* responsável pelo processo de extração dos dados (4.2).

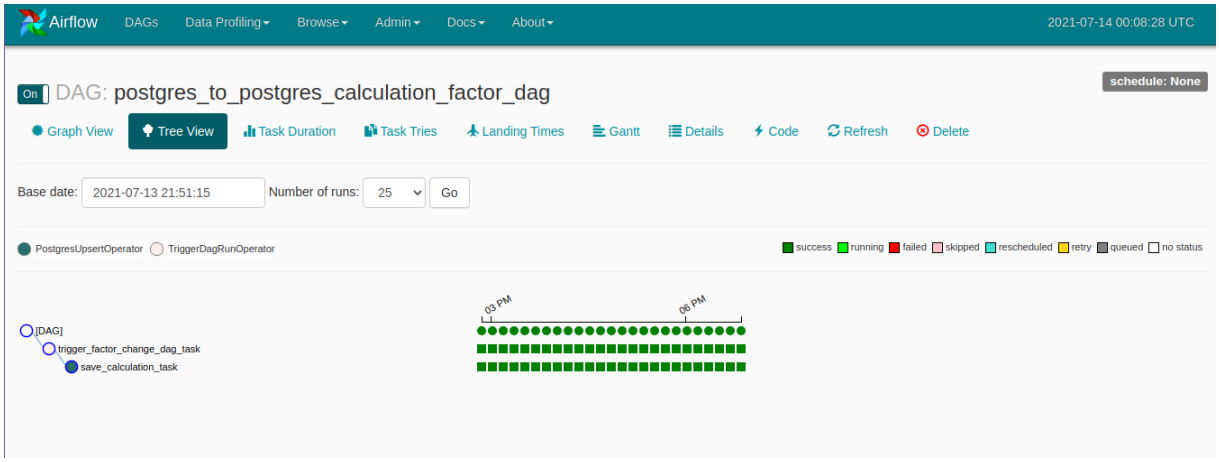
Figura 12 – *Dag* do processo de extração dos dados



Fonte: O Autor, 2021

- ***postgres_to_postgres_calculation_factor_dag*:** *dag* responsável pelo processo de cálculo do fator (4.3).

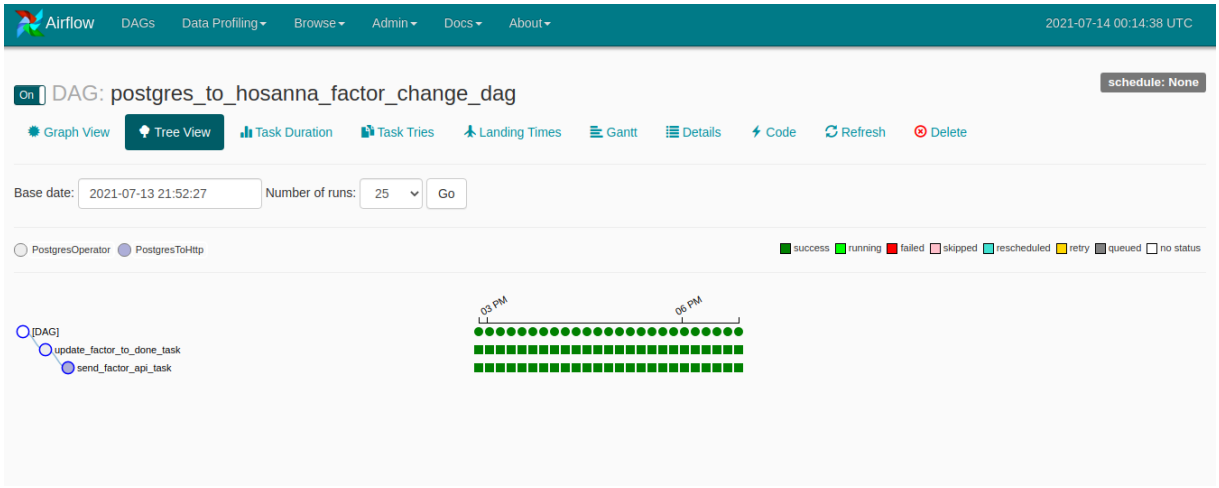
Figura 13 – *Dag* do processo de cálculo do fator



Fonte: O Autor, 2021

- ***postgres_to_hosanna_factor_change_dag*:** *dag* responsável pelo processo de alteração do fator (4.4).

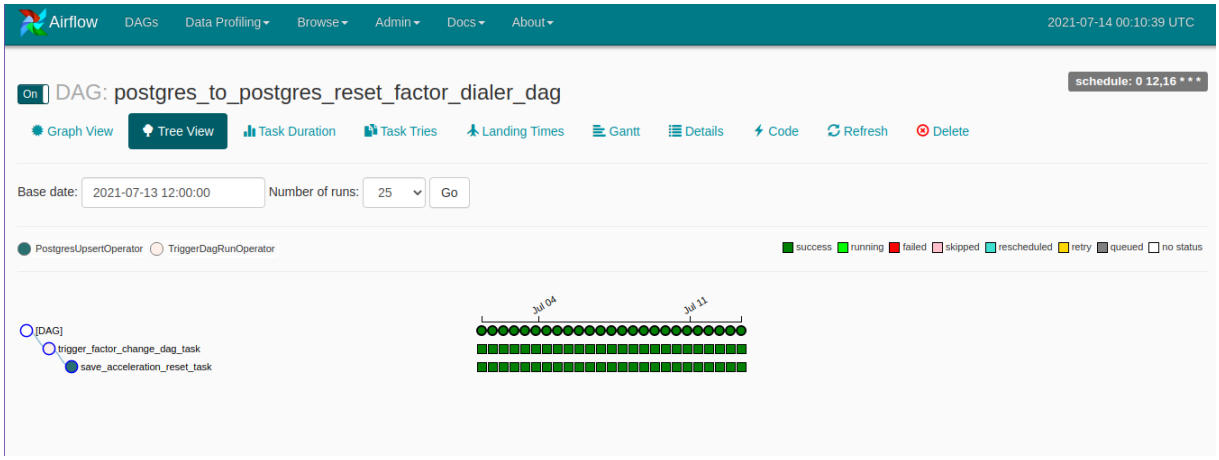
Figura 14 – *Dag* do processo de alteração do fator



Fonte: O Autor, 2021

- ***postgres_to_postgres_reset_factor_dialer_dag*:** *dag* responsável pelo processo de redefinição do fator (4.4).

Figura 15 – *Dag* do processo de redefinição do fator



Fonte: O Autor, 2021

Ao fim do projeto, foi realizado um repasse técnico do funcionamento do fluxo para toda a área de dados da companhia, abrangendo os times de ciência de dados, análise de dados, engenharia de dados e, um repasse de negócio para todo o time de operações que é impactado pela solução disponibilizada no discador. Além disso, o processo foi documentado a fim de disseminar o conteúdo para novos integrantes da companhia.

5.3 RESULTADOS OBTIDOS

Após a implementação do algoritmo e acompanhamento durante alguns meses, foi possível realizar um comparativo dos indicadores acompanhados internamente pelo negócio, sendo:

- **Percentual falado:** varia de 0% a 100%, sendo o inverso do percentual de contato não falado, que é utilizado como parte do gatilho implementado no algortimo e, indica qual o percentual dos *lead*s importados no discador atenderam as ligações disparadas pelo discador, ou seja, quanto mais perto de 100% é considerado melhor;
- **Percentual de *drop*:** varia de 0% a 100% e indica qual o percentual das ligações foram atendidas pelos *lead*s importados no discador e foram posteriormente abandonadas, por falta de atendentes disponíveis para atendimento, considerado péssima experiência ao usuário pelo negócio, ou seja, quanto mais perto de 0% é considerado melhor;
- **Percentual de agentes disponíveis:** varia de 0% a 100% e indica qual o percentual dos atendentes da operação está disponível para recebimento de ligações, considerado período improdutivo pelo negócio, ou seja, quanto mais perto de 0% é considerado melhor.

A Tabela 7 abaixo, mostra os resultados obtidos antes e depois da implentação do algoritmo, com base na média mensal dos indicadores.

Tabela 7 – Resultados obtidos pré e pós implementação

Mês	**Percentual Falado (média)**	**Percentual de *Drop* (média)**	**Percentual de Agentes Disponíveis (média)**
1° (pré)	51%	4%	N/A
2° (pré)	55%	4%	N/A
3° (pré)	54%	4%	N/A
4° (pré)	58%	1%	N/A
5° (pré)	54%	3%	N/A
1° (pós)	53%	5%	57%
2° (pós)	49%	11%	63%
3° (pós)	38%	15%	58%
4° (pós)	43%	10%	53%
5° (pós)	48%	8%	55%
6° (pós)	46%	8%	58%
7° (pós)	47%	6%	59%
8° (pós)	39%	4%	60%
9° (pós)	43%	5%	56%
10° (pós)	41%	3%	62%

Fonte: O Autor, 2021

Na Tabela 7, podemos notar que nos primeiros meses após a implementação do algoritmo, houve uma leve queda nos indicadores, isso se deve ao fato de que nos primeiros meses ainda houveram algumas interveções manuais que ocorriam durante o período de atendimento por parte do time operacional, interferindo nos resultados obtidos. Infelizmente, não foi possível comparar o indicador de percentual de agentes disponível, pois o dado não era salvo antes da implementação do algoritmo e não possuía histórico para a realização da análise.

Após a aderência completa ao algoritmo, que ocorreu a partir do 4° mês após a implementação, podemos notar na Tabela 7 que os indicadores tem uma melhora e voltam ao patamar que estavam antes da implementação. Vale ressaltar também, que o algoritmo busca preservar a experiência do cliente ao máximo, tentando manter o percentual de *drop* próximo a 5% e, por esse motivo diminuiu levemente o percentual falado, pois desacelerou o discador nos momentos de alto percentual de *drop*.

Com base nos resultados obtidos, é possível concluir, que o algoritmo conseguiu manter os indicadores acompanhados no mesmo patamar que estavam antes da implementação, conseguindo acelerar ao máximo o processo de discagem, sem deixar de preservar

a experiência do cliente. Provavelmente, seria possível ter resultados de produtividade operacional (percentual de agentes disponíveis) ainda melhores, caso o discador fosse acelerado ao máximo, porém não iria de acordo com um dos princípios da companhia, que visa entregar uma boa experiência ao cliente em todos os pontos de contato.

6 CONCLUSÃO

Com a finalização da construção deste Trabalho de Conclusão de Curso, foi possível validar que o próposito definido no presente projeto alcançou os objetivos gerais e específicos estabelecidos, respondendo as perguntas levantadas e resolvendo o problema mapeado.

Automatizar o processo de atualização do fator de aceleração do discador da companhia era o objetivo principal do presente projeto, além de garantir que o fluxo teria uma base sólida em dados, que trouxesse eficiência para o processo de discagens e tornando o fluxo os mais próximo possível do ideal.

Para que o objetivo fosse cumprido, foi necessário primeiramente, pesquisar trabalhos correlacionados com o tema para entender se era possível reaproveitar o conhecimento de pesquisas já realizadas no presente trabalho. Além disso, foi necessário o entendimento a fundo das regras de negócio utilizadas para as alterações do fator, que ocorriam de forma manual, afim de identificar como processo era realizado.

Realizando uma análise, juntamente com as regras identificadas o processo, foi notado então que a alteração, mesmo que manual, era guiada por indicadores acompanhados pelos times operacionais e, a partir desses indicadores foi possível extrair regras aptas à aplicação em um algoritmo, gerando um método de cálculo para a implementação.

O algoritmo foi desenvolvido, sendo capaz de extrair os dados de discagens e aplicá-los de acordo com as regras de cálculo estabelecidas, calculando os fatores ideais para o discador. Reutilizando recursos internos, foi possível conciliar o projeto com a arquitetura que já era utilizada para outros fins da companhia, quando se tratam de fluxos de dados e, desta forma, não houve aumento significativo nos custos com infraestrutura.

O algoritmo foi implementado no processo de vendas e ao fim da implementação, foi possível concluir que o algoritmo desenvolvido substituiu a ação humana com sucesso, evitando possíveis faltas de atualização que ocorriam por erro humano. Além disso, de acordo com os resultados obtidos (5.3), o algoritmo conseguiu manter os indicadores acompanhados, liberando um recurso humano para outras tarefas internas.

Não foram necessários ajustes nas regras do algoritmo, pois o algoritmo respeita indicadores que já eram acompanhados pela própria operação e se comportou conforme esperado. Desta forma, foram apenas realizados pequenos ajustes de padrão de codificação interno, sem grandes modificações a serem mapeadas para trabalhos futuros acerca deste tema.

O projeto foi entregue e comunicado aos times envolvidos, além dos repasses que

foram realizados para demonstrar o funcionamento do algoritmo e regras estabelecidas. O recurso humano que era responsável pela atualização do fator de aceleração do discador, foi redirecionado para outras atividades internas, afim de contribuir de forma mais ativa para o desenvolvimento da companhia. O projeto serviu de exemplo para a inserção da tecnologia nos times operacionais e reforço da importância de decisões baseadas em dados.

Referências

AMAZON. *Computação em nuvem com a AWS*. Filbert Pub., 2003. Disponível em: <https://aws.amazon.com/pt/what-is-aws>. Citado na página 30.

ANDRADE, M. M. d. *Introdução à metodologia do trabalho científico*. [S.l.: s.n.], 2010. 158–158 p. Citado na página 17.

BORGES, L. E. *Python para desenvolvedores: aborda Python 3.3*. [S.l.]: Novatec Editora, 2014. Citado na página 29.

BRENNER, P. *Primeiros passos com o Apache Airflow: ETL fácil, robusto e de baixo custo*. Data Hackers, 2019. Disponível em: <https://medium.com/data-hackers-/primeiros-passos-com-o-apache-airflow-etl-f%C3%A1cil-robusto-e-de-baixo-custo-f80db989edae>. Citado na página 28.

CARVALHO, S.; CAMPOS, W. Estatística básica simplificada. *Rio de Janeiro: Campus*, 2008. Citado na página 23.

CASTRO, C. d. M. A prática da pesquisa. In: *A prática da pesquisa*. [S.l.: s.n.], 1978. p. 156–156. Citado na página 16.

CHRISTINE, T. *Git e GitHub: por onde começar?* reprogramabr, 2019. Disponível em: <https://medium.com/reprogramabr/git-e-github-por-onde-come%C3%A7ar-ca88a783c223>. Citado na página 30.

COLZANI, V. F. *Guia para Redação do Trabalho Científico. 2 a edição*. [S.l.: s.n.], 2002. Citado na página 17.

DILINI, W. N.; KARUNARATNE, A.; RANASINGHE, D. Methods for controlling the call placemant ratio for outbound dialing of a call center. 2011. Citado 2 vezes nas páginas 27 e 28.

ESPINHA, R. G. *Escopo do Projeto: o que é e como fazer em 6 passos*. Artia, 2021. Disponível em: <https://artia.com/blog/escopo-do-projeto-como-fazer-em-6-passos>. Citado na página 36.

FERRARI, O. *10 coisas para fazer na execução do projeto*. 2019. Disponível em: <https://vanzolini.org.br/weblog/2018/05/24/10-coisas-para-fazer-na-execucao-do-projeto>. Citado na página 38.

FILHO, G. *Automatizando seu FLUXO de Trabalho com Airflow*. 2018. Disponível em: <https://medium.com/@gilsondev/automatizando-seu-fluxo-de-trabalho-com-airflow-4dbc1c932dcb>. Citado na página 28.

FILHO, P. J. de F. et al. Using simulation to predict market behavior for outbound call centers. In: IEEE. *2007 Winter Simulation Conference*. [S.l.], 2007. p. 2247–2251. Citado 2 vezes nas páginas 25 e 26.

FOURATI, S.; TABBANE, S. Optimization of a predictive dialing algorithm. In: IEEE. *2010 Sixth Advanced International Conference on Telecommunications*. [S.l.], 2010. p. 212–218. Citado 2 vezes nas páginas 26 e 27.

JÚNIOR, O. V. et al. O processo de trabalho e criação de valor nos callcenters brasileiros. 2015. Citado 4 vezes nas páginas 13, 15, 18 e 19.

KOIVISTO, T. Efficient data analysis pipelines. 2019. Citado 2 vezes nas páginas 22 e 23.

MACEDO, N. D. de. *Iniciação à pesquisa bibliográfica*. [S.l.]: Edições Loyola, 1995. Citado na página 17.

MEDEIROS, E. M. et al. Fluxograma. In: *III Mostra de Pesquisa, Ensino e Extensão do IFRS-Campus Viamão*. [S.l.: s.n.], 2018. Citado na página 35.

MUNOZ, D.; BRUTUS, M. C. Understanding the trade-offs in a call center. In: CITESEER. *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*. [S.l.], 2013. p. 3992–3993. Citado 2 vezes nas páginas 20 e 21.

NSTECH. *Git e Github-Por que e como usar? -Parte 1*. NSTech, 2019. Disponível em: <https://medium.com/nstech/git-e-github-por-que-e-como-usar-parte-1-fcb98261fde0>. Citado na página 30.

OLIVEIRA, W. *O que é scrum? Conceito, definições e etapas*. 2020. Disponível em: <https://evolvemvp.com/o-que-e-scrum-conceito-definicoes-e-etapas/>. Citado na página 32.

PEREIRA, P.; TORREÃO, P.; MARÇAL, A. S. Entendendo scrum para gerenciar projetos de forma ágil. *Mundo PM*, v. 1, p. 3–11, 2007. Citado 3 vezes nas páginas 32, 36 e 37.

PŁAZA, M.; PAWLIK, Ł. Influence of the contact center systems development on key performance indicators. *IEEE Access*, IEEE, v. 9, p. 44580–44591, 2021. Citado 2 vezes nas páginas 21 e 22.

PORTO, F.; ZIVIANI, A. Ciência de dados. *III Seminário de Grandes Desafios da Computação no Brasil, Rio de Janeiro, RJ*, 2014. Citado 2 vezes nas páginas 24 e 25.

POSTGRESQL. *About*. The PostgreSQL Global Development Group, 1996. Disponível em: <https://www.postgresql.org/about/>. Citado na página 29.

REIS, M. F. d. C. T. Metodologia da pesquisa. 2009. Citado na página 16.

SABBAGH, R. *Scrum: Gestão ágil para projetos de sucesso*. [S.l.]: Editora Casa do Código, 2014. Citado 2 vezes nas páginas 38 e 39.

SOMMERVILLE, I. *Engenharia de software*. [S.l.]: Pearson, 2012. Citado 2 vezes nas páginas 33 e 34.

YONAMINE, J. S.; SALIBY, E.; ZACHARIAS, M. L. B. Simulação de um call center para avaliação do impacto da adoção de um discador preditivo. 2007. Citado 4 vezes nas páginas 13, 15, 19 e 20.

Glossário

backlog itens que aguardam implementação em um produto. 36, 37

blog site atualizado com frequência. 28

booleano váriavel que possui somente dois valores 0 ou 1, falso ou verdadeiro. 43

bug defeito ou falha. 30

business value valor gerado para o negócio. 36

bytecode código independente gerado por um compilador. 29

call center central de atendimento. 13, 18–22, 25–27, 58

chatbot sistema capaz de simular uma conversação com um ser humano através de mensagens escritas. 21

cloud computing computação em nuvem especializada em armazenamento de dados e capacidade de computação sem necessidade de gerenciamento direto de um utilizador. 30

contact center centro de atendimento. 21

cron agendador de tarefas baseado em sistemas Unix. 44

dag fluxo de processamento de dados. 12, 41–45, 47–50, 52–56, 59, 60, 72–81

daily scrum reunião diária para alinhamento de execução de itens na metodologia *Scrum*. 38, 39

data driven decisões orientadas à dados. 15

date variável que possui formato de data. 43

dict estrutura de dados em dicionário, com uma sequência mutável de objetos mapeados. 43

drop contatos que abandonam ligações por não terem um agente disponível na equipe de atendimento para atendê-los. 50, 61, 62

false palavra em inglês que traduzida significa falso. 43

framework abstração que faz a união de códigos comuns entre diferentes projetos de programação e provê uma funcionalidade genérica. 29

hook conecta bases de dados e *APIs* para realização de uma ação em um *operator* do Airflow. 42, 45, 46, 48, 51, 53, 55

impediment backlog lista de impeditivos que podem ocorrer na implementação de um produto. 36, 37

int váriavel que possui formato de número inteiro. 44

lead pessoa com interesse em tornar-se possível cliente de uma empresa. 49, 50, 55, 61

mainframe grande computador com processamento de volume alto de informações. 30

online objeto ou pessoa que está conectado, acessa algo ou está disponível para algo. 47, 50

open source sistema com código fonte livre. 28

operator determina qual ação é realizada em uma *dag* do Airflow. 41, 42, 44–46, 48–55

path caminho de um diretório. 54

pipeline segmentação de instruções programadas. 22, 23, 28–30

product backlog lista ordenada de itens que precisam ser implementados em um produto. 36, 38, 39

product owner pessoa responsável por definir e acompanhar itens a serem implementados em um produto. 38, 39

script tarefas programadas via código para execução em sequência. 55

scrum master pessoa responsável por aplicar a metodologia *Scrum* em uma empresa. 39

scrum metodologia para gerenciamento de projetos. 8, 17, 32, 33, 38–40

software programa ou sistema que realiza uma sequência de instruções. 13, 14, 32, 40

sprint planning período de planejamento de itens priorizados para um ciclo de desenvolvimento na metodologia *Scrum*. 38, 39

sprint retrospective reunião de encerramento de um ciclo de desenvolvimento na metodologia *Scrum*. 38, 39

sprint período determinado para um ciclo de desenvolvimento na metodologia *Scrum*. 38, 39

string tipo de variável de programação no formato de texto. 43, 44, 48, 51, 53–55

telemarketing vendas e serviços por telefone. 13

token chave ou senha de acesso para algum sistema. 54

true palavra em inglês que traduzida significa verdadeiro. 43

update realização de atualização em algo. 54, 55

videobot sistema capaz de simular uma conversação com um ser humano através de mensagens de vídeo. 21

voicebot sistema capaz de simular uma conversação com um ser humano através de mensagens de voz. 21

web rede mundial de computadores. 18, 30

Anexos

ANEXO A – *Dag* extração de dados

```python
from airflow import DAG
from datetime import datetime
from airflow.operators.dagrun_operator import TriggerDagRunOperator
from lib.operators.mysql_to_postgres_operator import
    MysqlToPostgresOperator

default_args = {
    "owner": "DATA_TEAM",
    "depends_on_past": False,
    "start_date": datetime(2021, 1, 1),
    "email": ["data_team@contaazul.com"],
    "email_on_failure": True,
}

dag = DAG(
    "mysql_to_postgres_call_center_dialer_dag",
    catchup=False,
    default_args=default_args,
    schedule_interval="*/10 12-21 * * *",
    max_active_runs=1,
)
dag.doc_md = __doc__

source_sql = """
    select
        tabela_campanha.id as id_campanha,
        tabela_campanha.nome as nome_campanha,
        tabela_campanha.fator_atual as fator_atual,
        count(tabela_atendimentos.id) as total_atendentes,
        sum(if(disponivel is true, 1, 0)) as
            total_atendentes_disponiveis
    from tabela_campanha
    join tabela_atendimentos
```

```
32          on tabela_atendimentos.campanha_id = tabela_campanha.id
33      where tabela_campanha.ativa is true
34      group by 1, 2, 3
35  """
36
37  destination_sql = """
38      insert into dialer.call_center_log (
39          campaign_id,
40          campaign_name,
41          current_acceleration_factor,
42          total_agents,
43          avaiable_agents
44       )
45      values (
46          %(campaign_id)s,
47          %(campaign_name)s,
48          %(current_acceleration_factor)s,
49          %(total_agents)s,
50          %(avaiable_agents)s
51      )
52  """
53
54  save_call_center_task = MysqlToPostgresOperator(
55      task_id="save_call_center_task",
56      source_conn_id="mysql_hosanna",
57      source_sql=source_sql,
58      destination_conn_id="postgres_company",
59      destination_sql=destination_sql,
60      dag=dag,
61  )
62
63  trigger_calculation_dag_task = TriggerDagRunOperator(
64      task_id="trigger_calculation_dag_task",
65      trigger_dag_id="postgres_to_postgres_calculation_factor_dag",
66      dag=dag,
67  )
68
69  save_call_center_task >> trigger_calculation_dag_task
```

ANEXO B – *Dag* cálculo do fator

```python
from airflow import DAG
from datetime import datetime
from airflow.operators.dagrun_operator import TriggerDagRunOperator
from lib.operators.postgres_upsert_operator import
    PostgresUpsertOperator

default_args = {
    "owner": "DATA_TEAM",
    "depends_on_past": False,
    "start_date": datetime(2021, 1, 1),
    "email": ["data_team@contaazul.com"],
    "email_on_failure": True,
}

dag = DAG(
    "postgres_to_postgres_calculation_factor_dag",
    catchup=False,
    default_args=default_args,
    schedule_interval=None,
    max_active_runs=1,
)
dag.doc_md = __doc__

source_sql = """
    new_acceleration as(
        select
            call_center_log.id,
            call_center_log.campaign_id,
            call_center_log.total_agents,
            call_center_log.avaiable_agents,
            case
                when dialer_contacts.dropped_percent_contacts >= 5
                    then call_center_log.current_acceleration_factor - 1
```

```
33              when round(
34                      (
35                          call_center_log.avaiable_agents_percentage +
36                          dialer_contacts.not_speak_percent_contacts
37                      )/2, 1) > 6
38                  then call_center_log.current_acceleration_factor + 1
39              when round(
40                      (
41                          call_center_log.avaiable_agents_percentage +
42                          dialer_contacts.not_speak_percent_contacts
43                      )/2, 1) < 4
44                  then call_center_log.current_acceleration_factor - 1
45              else call_center_log.current_acceleration_factor
46          end as acceleration
47      from call_center_log
48      join dialer_contacts
49          on dialer_contacts.campaign_id = call_center_log.
                campaign_id
50  )
51  select
52      id as call_center_log_id,
53      campaign_id,
54      json_build_object (
55          'campaignid', campaign_id,
56          'acceleration', acceleration,
57          'channels', ceil(acceleration * total_agents)
58      )::text as json_content
59  from new_acceleration
60  where avaiable_agents > 0
61  and acceleration between 1 and 10
62 """
63
64 destination_sql = """
65  insert into dialer.acceleration_sync (
66      call_center_log_id,
67      campaign_id,
68      json_content,
69      sync_status
70  )
```

```
    values (
        %(call_center_log_id)s,
        %(campaign_id)s,
        %(json_content)s,
        'WAITING'
     )
"""

save_calculation_task = PostgresUpsertOperator(
    task_id="save_calculation_task",
    source_conn_id="postgres_company",
    source_sql=source_sql,
    destination_conn_id="postgres_company",
    destination_sql=destination_sql,
    dag=dag,
)

trigger_factor_change_dag_task = TriggerDagRunOperator(
    task_id="trigger_factor_change_dag_task",
    trigger_dag_id="postgres_to_hosanna_factor_change_dag",
    dag=dag,
)

save_calculation_task >> trigger_factor_change_dag_task
```

ANEXO C – *Dag* alteração do fator

```python
from airflow import DAG
from datetime import datetime
from lib.operators.postgres_to_http import PostgresToHttp
from airflow.operators.postgres_operator import PostgresOperator


default_args = {
    "owner": "DATA_TEAM",
    "depends_on_past": False,
    "start_date": datetime(2021, 1, 1),
    "email": ["data_team@contaazul.com"],
    "email_on_failure": True,
}

dag = DAG(
    "postgres_to_hosanna_factor_change_dag",
    catchup=False,
    default_args=default_args,
    schedule_interval=None,
    max_active_runs=1,
)
dag.doc_md = __doc__

select_factor_to_api_sql = """
    select
        json_content as json_data
    from dialer.acceleration_sync
    where sync_status = 'WAITING'
    and created_at >= current_date
"""

update_factor_sql = """
    update dialer.acceleration_sync
```

```
        set sync_status = 'DONE',
        sync_at = now()
    where sync_status = 'WAITING'
    and created_at >= current_date
"""

headers = {"Content-Type": "Application/json", "Accept": "application/
    json"}

send_factor_api_task = PostgresToHttp(
    task_id="send_factor_api_task",
    source_conn_id="postgres_company",
    http_conn_id="api_hosanna",
    endpoint="api/campaign/",
    headers=headers,
    source_sql=select_factor_to_api_sql,
    dag=dag,
)

update_factor_to_done_task = PostgresOperator(
    task_id="update_factor_to_done_task",
    sql=update_factor_sql,
    postgres_conn_id="postgres_company",
    dag=dag,
)

send_factor_api_task >> update_factor_to_done_task
```

ANEXO D – *Dag* redefinição do fator

```python
from airflow import DAG
from datetime import datetime
from airflow.operators.dagrun_operator import TriggerDagRunOperator
from lib.operators.postgres_upsert_operator import
    PostgresUpsertOperator

default_args = {
    "owner": "DATA_TEAM",
    "depends_on_past": False,
    "start_date": datetime(2021, 1, 1),
    "email": ["data_team@contaazul.com"],
    "email_on_failure": True,
}

dag = DAG(
    "postgres_to_postgres_reset_factor_dialer_dag",
    catchup=False,
    default_args=default_args,
    schedule_interval="0 12,16 * * *",
    max_active_runs=1,
)
dag.doc_md = __doc__

source_factor_reset_sql = """
    with campaigns_to_reset as (
        select
            max(last_log.id) as id,
            last_log.campaign_id
        from call_center_log last_log
        where last_log.created_at between current_date -1 and
            current_date
        group by 2
    )
```

```
    select
        id as call_center_log_id,
        campaign_id,
        json_build_object (
            'campaignid', campaign_id,
            'acceleration', 1.00
        )::text as json_content
    from campaigns_to_reset
"""

destination_factor_reset_sql = """
    insert into dialer.acceleration_sync (
        call_center_log_id,
        campaign_id,
        json_content,
        sync_status
    )
    values (
        %(call_center_log_id)s,
        %(campaign_id)s,
        %(json_content)s,
        'WAITING'
     )
"""

save_factor_reset_task = PostgresUpsertOperator(
    task_id="save_factor_reset_task",
    source_conn_id="postgres_company",
    source_sql=source_factor_reset_sql,
    destination_conn_id="postgres_company",
    destination_sql=destination_factor_reset_sql,
    dag=dag,
)

trigger_factor_change_dag_task = TriggerDagRunOperator(
    task_id="trigger_factor_change_dag_task",
    trigger_dag_id="postgres_to_hosanna_factor_change_dag",
    dag=dag,
)
```

```
71
72 save_acceleration_reset_task >> trigger_factor_change_dag_task
```