UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

João Victor Montanha Costa de Oliveira

Utilização da temperatura local para otimização autônoma de anúncios de marketing digital

São Carlos

2020

João Victor Montanha Costa de Oliveira

Utilização da temperatura local para otimização autônoma de anúncios de marketing digital

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

O48u

Oliveira, João Victor Montanha Costa de
Utilização da temperatura local para otimização autônoma de anúncios de marketing digital / João Victor Montanha Costa de Oliveira; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. Marketing Digital. 2. Google Ads. 3. Dados de temperatura. 4. Google Ads Script. 5. Automação. 6. Geolocalização. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: João Victor Montanha Costa de Oliveira

Título: "Utilização da temperatura local para otimização autônoma de anúncios de marketing digital"

Trabalho de Conclusão de Curso defendido e aprovado em 26/06/2020,

com NOTA 10,0 (DEZ, ZERO), pela Comissão Julgadora:

Prof. Associado Evandro Luis Linhari Rodrigues - Orientador - SEL/EESC/USP (Docente Aposentado)

Mestre Rafael Guedes Lang - Doutorando - SEL/EESC/USP

Mestre Adam Henrique Moreira Pinto - Doutorando - ICMC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

AGRADECIMENTOS

Agradeço primeiramente aos meus pais e à minha irmã que estiveram presentes durante toda minha trajetória escolar, não medindo esforços para me apoiar sempre que precisei. Jamais teria tido a oportunidade de escrever uma monografia de conclusão de curso sem o incentivo e suporte deles.

Também gostaria de agradecer ao professor Floriano Castilho, responsável direto pela minha paixão pelas ciências exatas e pela engenharia devido às suas incríveis aulas que tive o prazer de participar entre os anos de 2006 e 2010, me ensinando a raciocinar de uma forma totalmente diferente, influenciando muito meu aprendizado até hoje.

Dentro da universidade, agradeço a todos meus amigos que estiveram presentes e fizeram deste período uma experiência única e excelente. Agradeço também ao Warthog Robotics, grupo de extensão ao qual fiz parte durante a maior parte da minha graduação e que me deu a oportunidade de aprender engenharia na prática e conhecer pessoas incríveis. Estendo também os agradecimentos aos professores que fizeram a diferença nas aulas, em especial ao Prof. Dr. Evandro Rodrigues, que além de aceitar ser o orientador deste trabalho, sempre incentivou em suas aulas a busca pelas novas tecnologias e a pensar como engenheiro na resolução de problemas.

Por fim, agradeço à Raccoon Marketing Digital, empresa a qual estagiei e hoje sou funcionário, por todo o conhecimento adquirido na área de marketing digital, permitindo que eu participasse de toda a idealização e realização da ferramenta descrita nesta monografia. Agradeço em especial à Alicia Dias, que esteve presente durante todo o desenvolvimento, me auxiliando sempre que necessário.

RESUMO

OLIVEIRA, J. **Utilização da temperatura local para otimização autônoma de anúncios de marketing digital**. 2020. 86p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Esta monografia tem como objetivo desenvolver uma ferramenta capaz de utilizar informações de temperatura de cada cidade para influenciar a performance de anúncios de marketing digital em uma localização através da modificação de ajustes de lance de local no Google Ads. Como diferencial em relação a outras ferramentas disponíveis, esta foi criada prezando pela fácil personalização sem necessidade de conhecimentos prévios de programação, além de utilizar o histórico de resultados das campanhas e cidades dos últimos sete dias como um modificador de lance. O *script* desenvolvido utiliza ferramentas gratuitas do Google, como o Google Planilhas e as APIs de desenvolvedores, obtendo os dados de temperatura diretamente do Centro de Previsão de Tempo e Estudos Climáticos (CPTEC) pertencente ao INPE (Instituto Nacional de Pesquisas Espaciais). Toda estrutura do código e da planilha de controle são explicadas neste documento, incluindo todas as possíveis customizações e próximos passos que podem ser efetuados com este trabalho.

Palavras-chave: Marketing digital. Google Ads. Google Ads Script. Dados de temperatura. Automação. Geolocalização.

ABSTRACT

OLIVEIRA, J. **Usage of local temperature to autonomous optimization of digital marketing ads**. 2020. 86p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

This monograph aims to develop a tool capable of using temperature information from each city to influence the performance of digital marketing ads in a location by modifying location bid adjustments in Google Ads. As a differential in relation to other available tools, this was created aiming at an easy customization without the need for previous programming knowledge, using campaigns and locations previous results from the last seven days as a bid modifier. The developed script uses free Google tools, such as Google Spreadsheets and its developers APIs, obtaining temperature data directly from brazilian Weather Forecast and Climate Studies Center (CPTEC) from INPE (National Institute for Space Research). The entire structure of the code and the control spreadsheet are explained in this document, including all possible customization and next steps that can be performed with this work.

Keywords: Digital Marketing. Google Ads. Google Ads Script. Temperature data. Automation. Geolocation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CPTEC	Centro de Previsão de Tempo e Estudos Climáticos
HTML5	Hypertext Markup Language, versão 5
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
INPE	Instituto Nacional de Pesquisas Espaciais
ROAS	Retorno Sobre o Investimento Publicitário
URL	Uniform Resource Locator
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Orientar os anúncios para aparecerem para as pessoas corretas e no momento correto é um dos principais objetivos em uma estratégia de marketing de performance, a qual se baseia em dados para maximizar o retorno em métricas relevantes em relação ao investimento realizado. Justamente por isso, este trabalho foi desenvolvido com o intuito em identificar os locais em que a sensação de temperatura do dia propicia um melhor cenário para conversão, ou seja, uma ação desejada (como uma venda, por exemplo) seja executada a partir de uma ação publicitária ou anúncio.

No setor de climatização e ventilação, por exemplo, há um aumento da necessidade da aquisição de ventiladores e aparelhos de ar-condicionado em locais com alta temperatura, como evidenciado pela notícia do Estadão (2019), em que houve crescimento de 70% de vendas de ventiladores durante o verão. Em contrapartida, aquecedores e aparelhos de ar-condicionado que possuem a função de aquecer tendem a serem mais requisitados em tempos frios. Direcionar esses produtos para os locais mais adequados é, então, fundamental para uma empresa que os tenha em seu catálogo.

Outro exemplo está nas baterias automotivas, que tendem a apresentar mais defeitos em temperaturas extremas, principalmente durante o frio, em que problemas com a bateria são evidenciados devido a incapacidade de produzir carga suficiente para o motor de partida nesta condição (Chikara et al., 2020).

Pode-se utilizar estrategicamente a informação de temperatura em muitos outros setores, como o alimentício, o da moda e o de lazer, que são influenciados direta e indiretamente por ela. Coleções de outlet, por exemplo, que contam com roupas de estações anteriores, podem ter sua divulgação impulsionada com melhor retorno em locais em que a temperatura está fora do padrão da estação atual.

Somando-se a todos estes pontos levantados a vasta diversidade climática do Brasil devido a sua extensão territorial, torna-se evidente os benefícios em utilizar a temperatura a favor do marketing digital para um negócio, principalmente se a cobertura do serviço oferecido for nacional. Ao mesmo tempo, seria ineficiente analisar a temperatura de cada cidade e ajustar manualmente as campanhas de anúncios, o que motiva a criação de um script capaz de realizar estas ações de maneira automatizada.

Buscando explorar oportunidades nesta área em crescente desenvolvimento, surgiu, em conjunto com a Raccoon Marketing Digital[1], a ideia de elaborar uma solução de engenharia para otimizar os resultados em marketing digital através do uso de informações de temperatura de cada cidade, que será endereçada neste trabalho.

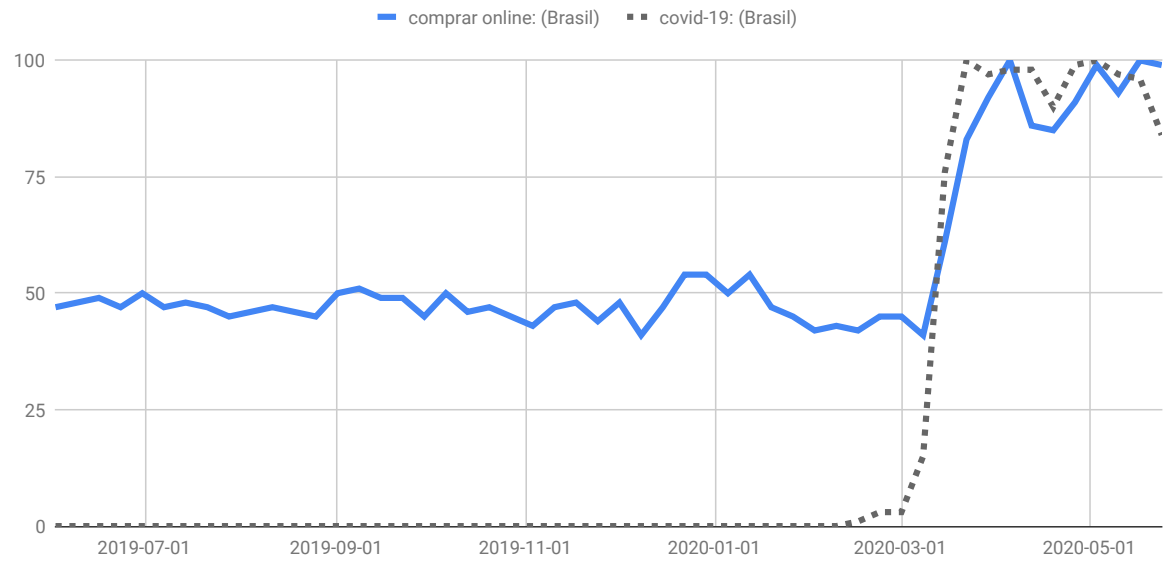
[1] Agência de Marketing Digital de São Carlos, SP <https://raccoon.ag/>.

1.1 Motivação

A ascensão ano a ano do setor de marketing digital, que vem ocupando o espaço do marketing tradicional com potencial de ser um dos setores mais expressivos do futuro (PAK et al., 2018), teve sua importância ampliada com a chegada do Covid-19, o novo corona vírus, e sua mudança no cotidiano das pessoas e das empresas em todo o mundo. Em um período em que o isolamento social é essencial e até mesmo imposto para a população, a internet passou a ser a sustentação de diversos negócios, forçando que empresas e estabelecimentos utilizem o meio digital (SEMRUSH, 2020).

O gráfico apresentado pela Figura 1 ilustra o crescimento do interesse pela pesquisa do termo "comprar online" no território brasileiro em comparação com o termo "covid-19" no Google, evidenciando ainda mais a importância das vendas onlines neste período delicado e, consequentemente, o aumento da necessidade de se utilizar estratégias de marketing digital para se obter sucesso neste meio.

Figura 1: Interesse de pesquisa relativo ao ponto mais alto durante o período de 2 de jun. de 2019 a 24 de mai. de 2020



Fonte: Elaborada pelo autor a partir de dados do Google Trends.

1.2 Objetivo

O objetivo deste trabalho consiste em disponibilizar uma solução capaz de auxiliar de maneira automatizada o gerenciamento de investimento de mídias pagas de maneira estratégica para que possa ser utilizada por todos os anunciantes, principalmente os de pequeno e médio porte que possuem maiores restrições orçamentárias e necessitam de maior precisão no direcionamento de seus anúncios.

1.3 Justificativa

Para atingir o objetivo desta monografia, deve-se elaborar um script capaz de definir automaticamente e em menos de trinta minutos, ajustes de lance em campanhas do Google Ads[2] para todas as cidades brasileiras desejadas, baseando-se em dados de temperatura. A solução deve ser facilmente personalizada para cada modelo de negócio, não exigindo conhecimentos prévios de programação por parte de seu utilizador. Além disso, deve levar em consideração o histórico recente de resultados naquele local, adicionando-se assim uma inteligência extra aos ajustes, que pode ser fundamental para melhores resultados.

Existem na literatura outras soluções que utilizam a temperatura como entrada para determinar-se ajustes de lances em Google Ads, como a apresentada pelo Google (2019). No entanto, essas soluções falham em dois aspectos importantes para o objetivo deste trabalho:

Fácil personalização

A configuração das condições desejadas exige elevado trabalho manual e desencoraja a utilização de várias campanhas e cidades, uma vez que é necessário adicionar cada local e cada campanha em uma linha, definindo sua regra.

Outra questão está na utilização do OpenWeatherMap[3], que pode apresentar problemas de desambiguação devido a existência de cidades homônimas, porém de estados diferentes, informação que não consta no serviço.

Uso do histórico de resultados

As alterações realizadas são baseadas puramente na temperatura, não levando em consideração os resultados recentes do local. Desta forma, é necessária intervenção manual para modificar os parâmetros das regras caso alguma cidade esteja se sobressaindo em relação ao objetivo da campanha, ferindo assim a ideia de ser um solução automática.

1.4 Organização do trabalho

Esta monografia inicia-se no Capítulo 2 com explicações e fundamentações acerca da área de marketing digital e do Google Ads, além de introduzir informações relevantes sobre os recursos utilizados no código elaborado. Na sequência, o Capítulo 3 apresenta o desenvolvimento do trabalho, informando todo o processo de elaboração e funcionamento do script desenvolvido, com os resultados obtidos pelos testes executados sendo apreciados pelo Capítulo 4. Por fim, a monografia é concluída no Capítulo 5, dissertando também sobre seus possíveis próximos passos.

[2] Plataforma de marketing digital do Google <https://ads.google.com/>.
[3] Serviço de processamento de dados de temperatura <https://openweathermap.org/>.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos e ferramentas utilizados para o desenvolvimento do trabalho.

2.1 Marketing digital

Com o marketing deixando de ser veiculado apenas em seus meios tradicionais para atingir os canais digitais como mídias sociais, blogs e páginas da web, que antes eram vistos apenas como uma forma de entretenimento e de conexão entre pessoas, surgiu o conceito de marketing digital, referindo-se assim à utilização de tecnologias digitais para promoção de produtos e serviços, o que tem crescido rapidamente nos últimos anos (BENGEL; SHAWKI; AGGARWAL, 2015).

2.1.1 Métricas relevantes

Devido à possibilidade de se obter dados fieis e em larga escala através de *tags*, que podem ser definidas como um pedaço de código que é adicionado em uma página web com o objetivo de fornecer informações sobre sua utilização por cada usuário (BENGEL; SHAWKI; AGGARWAL, 2015), é possível lidar com o marketing digital de forma analítica, utilizando as informações e métricas relevantes para encontrar um ponto de máximo retorno em relação a um investimento realizado, o chamado "marketing de performance". Deseja-se obter então mais resultados com o menor investimento possível.

Neste trabalho, duas métricas são utilizadas para medir, de forma simplificada, a performance de uma campanha.

Conversões

Uma conversão pode ser interpretada como uma meta atingida. Para um comércio eletrônico, por exemplo, uma conversão pode ser um cliente efetuar uma transação, porém há diversas outras metas que podem ser consideradas conversões em outros contextos, como o cadastro em um site, o download de um aplicativo ou até mesmo acessar uma página específica.

Como as conversões idealmente estão relacionadas ao objetivo de negócio, torna-se relevante atribuir qual a contribuição de uma ação de marketing digital para cada conversão. Devido a isso, surgiu o conceito de modelos de atribuição.

Diferentes modelos de atribuição estão disponíveis nas diversas plataformas de marketing digital, sendo os mais conhecidos o modelo "último clique", que atribui a conversão em sua totalidade ao último canal pelo qual o usuário passou antes da conversão,

e o modelo “baseado em dados”, que utiliza informações passadas para definir a contribuição de cada um dos canais que o usuário interagiu antes da conversão (GOOGLE, 2020). Enquanto no primeiro modelo citado as conversões são tratadas como números inteiros, no segundo é comum ocorrer o fracionamento de uma conversão em ações ou canais distintos.

Associada à métrica de conversão, está a métrica de **valor de conversão**. Desta forma, é possível diferenciar conversões mais relevantes para um negócio. No caso do comércio eletrônico, é comum que o valor de conversão seja equivalente à receita da transação.

Retorno Sobre o Investimento Publicitário

ROAS, ou Retorno Sobre o Investimento Publicitário, é uma métrica importante para o comércio, relacionando a receita total de uma campanha ou ação de marketing com seu investimento.

No Google Ads, plataforma utilizada neste trabalho, o ROAS deve ser calculado pela divisão do valor de conversão com o custo, ambos referentes a uma mesma dimensão em análise, que pode ser, dentre várias possibilidades, uma campanha, uma segmentação de público específica ou até mesmo um local, como é realizado pelo script apresentado nesta monografia.

O script em questão permite que outras métricas sejam utilizadas no lugar das previamente configuradas, exigindo no entanto adaptações do código para seu funcionamento correto. Para modelos de negócios em que não seja adequado analisar ROAS, pode-se, por exemplo, substituí-lo pelo Custo por Aquisição, ou CPA, que relaciona o investimento com a quantidade de conversões obtidas.

2.1.2 Google Ads

Google Ads é a solução de publicidade online do Google. Trata-se de uma plataforma na qual os anunciantes podem configurar e gerenciar anúncios pagos para atingir seus objetivos de negócio nas mais diversas redes do Google, como a Rede de Pesquisa (a busca do Google) e a Rede de Display (grupo com mais de dois milhões de websites, vídeos e aplicativos). De acordo com o critério do anunciante, define-se o perfil do público desejado para suas campanhas publicitárias, em quais momentos seus anúncios irá atingi-lo e as metas de negócio desejadas. (GOOGLE, 2020).

2.1.2.1 Estrutura da plataforma

Dentro do Google Ads, cada **conta de anúncio** possui **campanhas** que, por sua vez, possuem **grupos de anúncios**, os quais, como o nome sugere, possuem **anúncios**. Além disso, é possível possuir um agrupamento de contas de anúncios em uma **MCC**, sigla para Minha Central de Contas. A Figura 2 ilustra esta hierarquia com algumas

particularidades de cada elemento.

Figura 2: Estrutura de contas no Google Ads

Fonte: Google (2020).

Cada campanha é determinada por um nome, que deve ser único dentro da conta de anúncios, e um conjunto de configurações, com algumas delas sendo:

a) **orçamento**: define o investimento médio diário ou mensal;

b) **tipo de campanha**: determina como e onde os anúncios serão vistos. Alguns exemplos são:

   – Rede de Pesquisa (anúncios de texto);
   – Rede de Display (anúncios gráficos e de texto pela Web);
   – Shopping (anúncios de produtos);
   – Vídeo (anúncios em YouTube e sites parceiros);
   – App (anúncios voltados para a promoção de aplicativos);
   – Discovery (anúncios em produtos do Google, como o YouTube, o Gmail e o Discover).

c) **conversões**: determina qual ação de conversão será contabilizada por essa campanha;

d) **estratégia de lances**: define qual a estratégia será utilizada nos leilões de anúncios, podendo ser lances manuais ou lances automáticos;

e) **locais**: define em quais locais os anúncios serão veiculados, podendo ser limitados a países, estados, cidades ou até mesmo dentro de raios delimitadores.

Para o funcionamento do script descrito nesta monografia, é necessário que a campanha utilize a estratégia de lance manual e possua segmentadas as cidades as quais deseja-se monitorar a temperatura. Devido a isso, campanhas do tipo “App” ou “Discovery” não são contempladas por esta solução, uma vez que utilizam exclusivamente as estratégias de lances automáticos.

Dentro de cada campanha, devem ser configurados grupos de anúncios, os quais delimitam um conjunto de anúncios e suas regras específicas, variando para cada tipo de campanha. Por exemplo, em uma campanha de pesquisa, cada grupo de anúncio possui um conjunto de palavras-chave responsável por acionar seus anúncios - caso alguém pesquise no Google por um termo coberto pelas regras de palavra-chave, os anúncios daquele grupo podem entrar no leilão e aparecer para este usuário.

Da mesma forma que os grupos de anúncios, os anúncios também variam de acordo com tipo de campanha. Enquanto os anúncios de texto contém apenas elementos textuais, anúncios da Rede de Display podem possuir imagens, vídeos e até mesmo elementos em HTML5 (Hypertext Markup Language, versão 5). Como as configurações dos anúncios e de seus grupos não são afetadas pelo script, as explicações destes elementos não serão aprofundadas.

2.1.2.2 Ajustes de lance

O fato de um anúncio ser impresso ou não e sua posição na página são definidos através dos leilões de anúncio, explicados na subseção 2.1.2.3. O valor de cada lance em uma campanha que utiliza a estratégia de lances manuais é definida por um fator principal de cada grupo de anúncio (por exemplo, em anúncios de texto, seriam as palavras-chave; já em um anúncio de Shopping, seriam os produtos) acrescentado de seus ajustes adicionais.

Os ajustes de lances podem ser definidos em diversas elementos da campanha ou do grupo de anúncio, sempre indicados em percentuais de aumento ou redução em relação ao lance principal, sendo multiplicados para definir o lance final. Por exemplo, um lance de R$1,00 que possui um ajuste de $+10\%$ e outro de -20% resultaria em um lance final de R$0,88, devido ao ajuste final ser $1,1 * 0,8 = 0,88$. Este lance representa qual o valor máximo desejado a se pagar por uma ação, como clique ou visualização (GOOGLE, 2020).

Alguns exemplos de ajustes de lance são:

a) **dispositivo**: ajustes em relação ao tipo de dispositivo utilizado pelo usuário (desktop, mobile, tablet ou televisão);

b) **local**: ajustes em relação ao local geográfico do usuário;

c) **agendamento**: ajustes em relação ao horário e dia da semana;

d) **lista de público**: ajustes em relação às listas de público-alvo definidas para a campanha ou grupo de anúncio;

e) **informação demográfica**: ajustes em relação às informações demográficas do usuários, como gênero, idade e renda.

2.1.2.3 Funcionamento dos leilões

Denomina-se como leilão o processo do Google para decidir se um anúncio será exibido, quais anúncios aparecerão e a ordem de exibição, se aplicada. Este processo é acionado dependendo do tipo da campanha - em anúncios de pesquisa, por exemplo, ocorrerá quando uma pesquisa no Google possui correspondência com as palavras-chave escolhidas na campanha. Participam do leilão apenas anúncios qualificados para exibição, isto é, aprovados pelas políticas do Google e que possuem configurações de segmentação que abrangem a situação daquele leilão (por exemplo, um anúncio segmentado para aparecer apenas em São Paulo não participará de leilões no Rio de Janeiro).

O resultado do leilão, incluindo as posições em que os anúncios irão aparecer, são definidas por uma série de fatores:

a) **valor do lance**, ou seja, o valor máximo desejado a se pagar pela exibição ou interação com o anúncio, a depender do tipo de campanha;

b) **qualidade do anúncio e da página de destino**, analisando a relevância e utilidade do mesmo para a pessoa que o verá;

c) **classificação mínima do anúncio**, definida pelo Google Ads para garantir publicidade de alta qualidade;

d) **concorrência do leilão**;

e) **contexto**, levando em consideração vários fatores como pesquisas realizadas, localização geográfica, dispositivo utilizado, entre outros;

f) **estimativa do impacto de extensões e formatos de anúncio**.

Nesta monografia, deseja-se influenciar o fator “valor do lance”, alterando-se o limite do lance com base nos dados de temperatura de cada local.

2.2 Google Developers

O Google Developers[1] é um site do Google que expõe uma variedade de ferramentas de desenvolvimento de software, Interfaces de Programação de Aplicações (APIs) e recursos técnicos, contendo documentações e grupos de discussões acerca desse assunto. Algumas dessas ferramentas são usadas nesta monografia e estão explicadas nas subseções a seguir.

[1] Todos os produtos disponíveis no Google Developers podem ser encontrados em <https://developers.google.com/products>.

2.2.1 Google Apps Script

Com o Google Apps Script, é possível desenvolver complementos para os principais produtos do Google, como o Google Planilhas, Google Documentos, Gmail, Google Agenda e Hangouts, por exemplo. A seguir serão apresentados os pontos relevantes da documentação disponível em Google (2020) para esta monografia.

2.2.1.1 Serviço para Google Planilhas

Para manipulação de planilhas, serão utilizadas algumas classes do *Spreadsheet Service*, que permite a criação, acesso e modificação de arquivos do Google Planilhas.

Classe Spreadsheet App

Utilizada para acessar e criar planilhas. É utilizado neste trabalho apenas o método *openById*, o qual abre uma planilha a partir de seu código de identificação, retornando um objeto *SpreadSheet*;

Classe Spreadsheet

Responsável por acessar e modificar planilhas, sendo utilizado o método *getSheetByName*, o qual retorna um objeto *Sheet* contendo uma aba de uma planilha a partir de seu nome.

Classe Sheet

Responsável por acessar e modificar uma aba de uma planilha, sendo importantes os métodos *getDataRange*, o qual retorna um objeto *Range* que corresponde à uma seleção na qual os dados estão apresentados; e *getRange*, que também retorna um objeto *Range*, porém a partir de suas coordenadas.

Classe Range

Utilizada para acessar e modificar uma seleção de uma planilha. Nesta monografia, são utilizados os métodos *getValues*, para retornar uma matriz de duas dimensões contendo os valores da seleção em questão; *getLastRow*, para retornar um inteiro referente à posição da última linha de uma seleção; *setValue*, para escrever um valor em uma seleção de uma dimensão; e *setValues*, para escrever valores em uma seleção retangular.

2.2.1.2 Serviço para acesso a URLs

Através do *URL Fetch Service*, é possível que scripts façam requisições HTTP e HTTPS utilizando a infraestrutura de rede do Google.

Classe UrlFetchApp

É utilizado o método *fetch* desta classe para realizar uma requisição de busca, retornando um objeto *HTTPResponse*.

Classe HTTPResponse

Permite acesso a informações específicas de uma resposta HTTP. Nesta monografia é utilizado o método *getContentText*, responsável por retornar em uma *string* o conteúdo da resposta.

2.2.1.3 Serviço para acesso e manipulação de XML

Por meio do *XML Service*, pode-se analisar, criar e navegar em documentos XML utilizando um script.

Classe XMLService

Responsável por garantir as funcionalidades do serviço homônimo, tem o método *parse* utilizado nesta monografia, que cria um objeto *Document* que representa um documento XML.

Classe Document

Representa um documento XML, sendo utilizado neste trabalho o método *getRootElement*, o qual retorna um objeto *Element* contendo o nó principal do XML. Caso não possua, retorna vazio.

Classe Element

Representa um elemento nó de um XML. São utilizados os métodos *getChildren* e *getValue* desta classe. O primeiro, obtém todos os elementos-filhos imediatos ao nó em questão que possuem o mesmo nome informado como parâmetro, sendo retornados em um vetor de elementos. Já o segundo retorna uma *string* contendo o valor em texto de todos os elementos que são direta ou indiretamente filhos do nó em questão.

2.2.1.4 Serviços utilitários

Os scripts do Google podem utilizar da classe *Utilities* para facilitar tarefas variadas. Nesta monografia, é utilizado o método *formatString* para elaboração de *strings* utilizando as notações "%-" para substituição de termos.

2.2.2 Google Ads Script

Com a API do Google Ads Script, é possível automatizar processos do Google Ads, permitindo também a interação com dados externos, usando JavaScript de forma simplificada em um ambiente de desenvolvimento integrado no próprio navegador. Este trabalho foi desenvolvido utilizando-se a API Adwords v201809.

O objeto raiz da API é o *AdsApp*, do qual desprendem-se os outros objetos citados nesta subseção.

2.2.2.1 Relatórios do Google Ads

A partir do método *report* do objeto *AdsApp* é possível buscar um relatório do Google Ads a partir de uma *query*, retornando assim um *Report*.

Report

Representa um relatório do Google Ads. Neste trabalho, utiliza-se o método *rows* para se obter um iterador para as linhas do relatório, chamado de *ReportRowIterator*.

ReportRowIterator

Trata-se de um iterador para linhas de relatórios do Google Ads. Seus dois únicos métodos, *hasNext* e *next* são utilizados, sendo o primeiro para determinar se há ainda linhas para serem percorridas e o segundo para avançar para a próxima linha.

2.2.2.2 Critérios de direcionamento de anúncios

Utilizando o método *targeting* do objeto *AdsApp* é possível acessar os critérios de direcionamento das campanhas. Para este trabalho, deseja-se acessar os critérios de direcionamento de locais, o que pode ser feito especializando este seletor para locais utilizando o método *targetedLocations*, que retorna um seletor do tipo *TargetedLocationSelector*.

TargetedLocationSelector

Representa todos os locais alvo selecionados, de forma a permitir filtros e ordenações. Para esta monografia, deseja-se buscar os locais que possuem código de identificação igual aos de uma lista, o que deve ser feito através do método *withIds*, que restringe o seletor para utilizando as IDs de campanha e de local do Google Ads.

Por fim, com o método *get* é possível obter um iterador do tipo *TargetedLocationIterator* para os locais desejados.

TargetedLocationIterator

Semelhante ao *ReportRowIterator*, são utilizados nesta monografia os métodos *hasNext* e *Next*, que auxiliam a navegar pelos locais-alvo das campanhas do Google Ads selecionados, que são representados como objetos *TargetedLocation*.

TargetedLocation

Representa um local alvo do Google Ads. Nesta monografia, é utilizado o método *setBidModifier*, responsável por alterar o valor de ajuste de lance para o local em questão.

2.3 Previsão do tempo em XML - CPTEC/INPE

Para o desenvolvimento deste trabalho, foi utilizado o serviço de previsão do tempo via XML do Centro de Previsão de Tempo e Estudos Climáticos (CPTEC) do Instituto Nacional de Pesquisas Espaciais (INPE) com caráter de prova de conceito para anúncios

fictícios. No caso de uma aplicação comercial, deve-se utilizar uma fonte de dados adequada para essa finalidade.

Nesta seção será explicado como se obter de cada cidade desejada a temperatura máxima prevista para aquele dia, com base nas informações fornecidas em CPTEC/INPE (2014).

2.3.1 Busca de localidade

Para se obter a temperatura, necessita-se antes conhecer o código do município dentro da base do CPTEC/INPE. Isso pode ser feito através de uma requisição na URL base <http://servicos.cptec.inpe.br/XML/listaCidades> com o parâmetro *city* correspondendo ao nome da localidade desejada sem acentos. Desta forma, a requisição para "São Carlos" seria realizada através da URL <http://servicos.cptec.inpe.br/XML/listaCidades?city=saocarlos>. É possível substituir o espaço por "%20" também.

Como resposta, será recebido um arquivo XML cujo nó "cidades" terá os seguintes elementos:

a) **nome**, contendo o nome do município;

b) **uf**, Unidade de Federação, útil para desambiguação de municípios homônimos;

c) **id**, código do município dentro do CPTEC/INPE, representado por um número inteiro positivo.

O elemento "id" é o desejado. A Tabela 1 contém os códigos referente às capitais de cada capital brasileira, obtidos seguindo esse procedimento.

2.3.2 Requisição da previsão do tempo

É possível fazer uma requisição dos dados de previsão de tempo para quatro ou sete dias utilizando o serviço de previsão do CPTEC/INPE. Como para este trabalho necessita-se apenas da previsão para o dia atual, será utilizada a requisição para quatro dias apenas.

Para efetuar a busca pela temperatura, é necessário utilizar a URL de requisição <http://servicos.cptec.inpe.br/XML/cidade/codigo_da_localidade/previsao.xml>, onde "codigo_da_localidade" deve ser substituído pelo código do município obtido na subseção 2.3.1. Por exemplo, para a cidade de Rio Branco, no Acre, a requisição deve ser feita por meio da URL <http://servicos.cptec.inpe.br/XML/cidade/240/previsao.xml>.

Como resposta, é obtido um arquivo XML com os seguintes elementos:

a) **nome**, com o nome do município selecionado;

b) **uf**, com a sigla da Unidade da Federação a qual o município pertence;

c) **atualizacao**, com a data da última atualização dos dados de previsão;

d) **previsao**, elementos que possuem a informação de previsão.

Cada elemento “previsao” contém informações referentes a um dos dias, ordenados da data mais próxima até a mais distante. Dependendo da hora do dia, a primeira data passa a ser referente ao dia seguinte - por conta disso, as buscas realizadas nos testes desta monografia ocorrem sempre por volta das oito horas da manhã, horário em que a previsão do mesmo dia ainda está disponível. Cada umas previsões possuem os seguintes elementos:

a) **dia**, contendo a data referente à previsão;

b) **tempo**, contendo uma sigla referente à condição do tempo para a localidade;

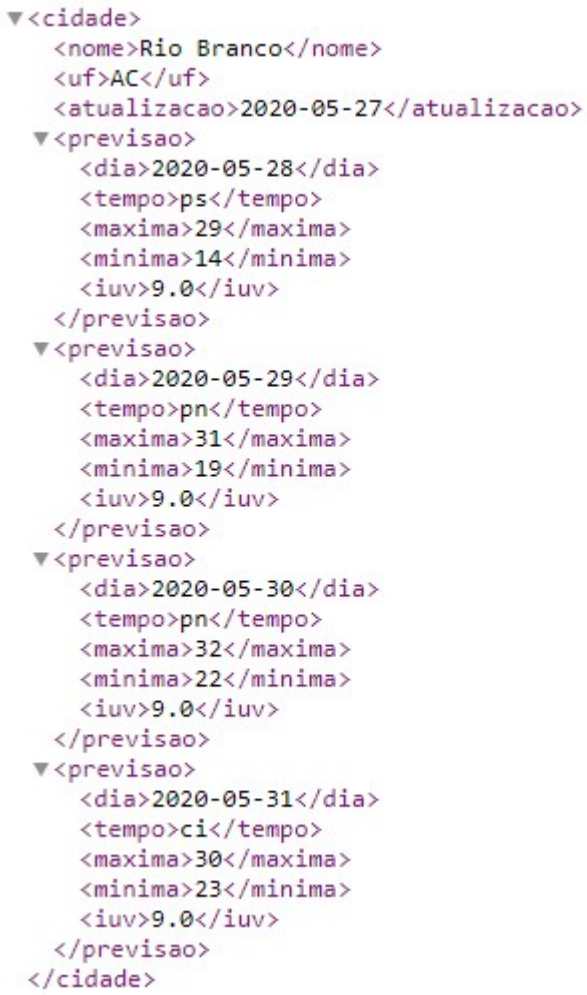
c) **maxima**, representando a temperatura máxima prevista para aquela data, em um valor inteiro;

d) **minima**, representando a temperatura mínima prevista para aquela data, em um valor inteiro;

e) **iuv**, representando o valor máximo diário de radiação ultravioleta, em um valor real (ponto flutuante).

A Figura 3 ilustra a árvore de elementos encontrada no XML do CPTEC/INPE para a cidade de Rio Branco, no Acre.

Figura 3: Árvore de elementos do XML do CPTEC/INPE para a capital Rio Branco



Fonte: Elaborada pelo autor a partir do XML com informações da previsão do tempo para municípios do CPTEC/INPE.

Tabela 1: Códigos CPTEC/INPE para cada capital brasileira

Capital	Código CPTEC/INPE
Aracaju	220
Belém	221
Belo Horizonte	222
Boa Vista	223
Brasília	224
Campo Grande	225
Cuiabá	226
Curitiba	227
Florianópolis	228
Fortaleza	229
Goiânia	230
João Pessoa	231
Macapá	232
Maceió	233
Manaus	234
Natal	235
Palmas	236
Porto Alegre	237
Porto Velho	238
Recife	239
Rio Branco	240
Rio de Janeiro	241
Salvador	242
São Luís	243
São Paulo	244
Teresina	245
Vitória	246

Fonte: Elaborada pelo autor com informações do XML do CPTEC/INPE.

3 DESENVOLVIMENTO

Para atingir o objetivo proposto nesta monografia, foi desenvolvido um script capaz de alterar ajustes de lance de local no Google Ads com base em dados de temperatura e nas configurações desejadas pelo utilizador, as quais devem ser inseridas em uma planilha de controle.

Este capítulo apresenta como foram confeccionados a planilha de controle e o código desenvolvido, explicando cada um de seus elementos e seu funcionamento.

3.1 Planilha de controle

A planilha de controle é o local destinado à personalização do script para atender os objetivos de marketing digital do usuário. Ela deve ser criada através do Planilhas Google[1] de forma que o endereço de e-mail utilizado na execução do script na plataforma do Google Ads possua acesso de edição à planilha em questão.

Esta planilha permite ao usuário definir quais as cidades e campanhas de publicidade do Google Ads devem ser analisadas e alteradas pelo código, além de definir quais alterações e suas condições para serem realizadas.

Para melhor organização das informações, são necessárias seis abas, que estão listadas e explicadas nas subseções a seguir. É necessário atenção apenas com os nomes das abas e na ordem das colunas de cada uma delas, uma vez que são informações utilizadas no código apresentado na seção 3.2.

3.1.1 Controle

Esta aba é a responsável por definir em cada uma de suas linhas as regras que serão analisadas pelo código. Cada regra é composta por três elementos principais: o quê será afetado; suas condições; e suas alterações caso as condições sejam satisfeitas.

Para definir o que será afetado pela regra, são utilizadas as colunas **categoria da campanha** e **regiões**. A primeira é responsável por indicar quais as campanhas de publicidade do Google Ads serão analisadas e possivelmente modificadas nesta regra, enquanto a última elenca quais cidades serão consideradas nesta regra.

Na definição das condições, também são consideradas duas colunas: **condição climática**, a qual define os limites de temperatura para a regra em questão; e **condição de ajuste modificado**, que define condições extras relacionadas ao desempenho histórico das campanhas para um ajuste modificado.

[1] Editor de planilhas do Google <https://www.google.com/intl/pt-BR/sheets/about/>.

Por fim, as alterações são definidas pelas últimas duas colunas, **ajuste padrão** e **ajuste modificado**, que devem ser preenchidas com valores entre 0.1 e 9.0 (utilizando o ponto como separador decimal), funcionando da seguinte forma:

a) caso o valor seja 1, o ajuste correspondente no Google Ads será nulo (0%);

b) caso o valor seja inferior a 1, o ajuste correspondente será negativo de módulo $1 - valor$;

c) caso o valor seja superior a 1, o ajuste correspondente será positivo de módulo $valor - 1$.

A Tabela 2 traz exemplos de ajustes realizados no Google Ads através dos valores definidos na planilha.

Tabela 2: Exemplos de ajustes de lance de acordo com os valores na planilha

Valor de ajuste na planilha	Ajuste correspondente no Google Ads
0.3	-70%
1.0	0%
2.0	+100%

Fonte: Elaborada pelo autor.

Caso uma cidade pertencente à região analisada esteja dentro da condição climática da regra, serão realizadas alterações ao ajuste de lance local destas cidades para as campanhas que compõem a categoria analisada da seguinte forma:

a) caso o resultado da campanha não satisfaça a condição de ajuste modificado, será aplicado o ajuste padrão;

b) caso a campanha satisfaça a condição de ajuste modificado, será aplicado o ajuste modificado.

A Tabela 3 traz como exemplo algumas linhas de uma planilha de controle preenchida para uma única região influenciando duas categorias de campanhas distintas.

3.1.2 Campanhas

Esta aba é responsável por definir as campanhas que compõem cada categoria de campanha utilizada na aba Controle (subseção 3.1.1). Apenas duas colunas são necessárias: **nome da campanha**, que deve conter o nome exato das campanhas como definido na plataforma do Google Ads; e **categoria da campanha**, a ser preenchida pelo usuário. A Tabela 4 traz como exemplo algumas linhas de uma planilha de campanhas preenchida.

Tabela 3: Exemplo de planilha de controle

Categoria da campanha	Regiões	Condição climática	Condição de ajuste mod.	Ajuste padrão	Ajuste modificado
Ar-Condicionado	Subtropical	Subtrop. Frio	ROAS > 20	0.8	1
Ar-Condicionado	Subtropical	Subtrop. Normal	ROAS > 20	1.1	1.2
Ar-Condicionado	Subtropical	Subtrop. Quente	ROAS > 20	1.5	1.7
Baterias	Subtropical	Subtrop. Frio	ROAS > 20	1.5	1.7
Baterias	Subtropical	Subtrop. Normal	ROAS > 20	1.1	1.2
Baterias	Subtropical	Subtrop. Quente	ROAS > 20	0.8	1

Fonte: Elaborada pelo autor.

Campanhas que compartilham o mesmo texto na coluna categoria da campanha são consideradas como da mesma categoria e, portanto, compartilham os mesmos valores de ajuste de lance padrão e modificado, além de serem avaliadas da mesma forma em termos de temperatura (a condição de ajuste modificado sempre considera as campanhas separadamente).

Apesar de ser possível adicionar uma campanha a mais de uma categoria, é aconselhado que cada campanha pertença a apenas uma única categoria, uma vez que os ajustes não são incrementais, apenas substituem o valor anterior.

Tabela 4: Exemplo de planilha de campanhas

Nome da campanha	Categoria da campanha
_GDN_ArInverter	Ar-Condicionado
_GDN_ArSplit	Ar-Condicionado
_S_ArCondicionado	Ar-Condicionado
_GDN_Baterias	Baterias
_GS_Baterias	Baterias

Fonte: Elaborada pelo autor.

3.1.3 Locais

Nesta aba devem estar listadas todas as cidades brasileiras que serão analisadas pelo script, relacionando-as às regiões correspondentes.

Para as informações de cada cidade, são necessários: o nome, sem qualquer acentuação; seu código do Google Ads, obtido na própria plataforma; e seu código do CPTEC/INPE, obtido conforme descrito na subseção 2.3.1.

Na última coluna, deve-se adicionar a qual região esta cidade pertence. As divisões por região podem ser definidas pelo usuário como ele preferir, porém é aconselhado utilizar o critério de clima para segmentação, uma vez que todas as cidades de uma região terão as mesmas faixas de temperatura máxima para definir se a previsão naquele dia está favorável ou não para os anúncios.

A Tabela 5 traz como exemplo algumas linhas de uma planilha de locais preenchida. Assim como na subseção 3.1.2, é possível que uma cidade esteja em mais de uma região, porém não aconselhável.

Tabela 5: Exemplo de planilha de locais

Cidade	Código Ads	Código CPTEC/INPE	Regiões
Brasilia	1001541	224	Tropical
Criciuma	1001704	1671	Subtropical
Guarulhos	1001736	2247	Tropical de altitude
Maceio	1001506	233	Tropical
Salvador	1001533	242	Tropical litorâneo

Fonte: Elaborada pelo autor.

3.1.4 Condições

Nesta aba, são listadas todas as condições climáticas que serão utilizadas pelas regras na aba de controle descrita em subseção 3.1.1. Cada uma deve possuir um nome, o tipo da condição (‘Entre’, ‘Acima’ ou ‘Abaixo’) e os valores em graus Celsius. Caso a condição seja ‘Acima’ ou ‘Abaixo’, apenas a primeira coluna de valor deve ser preenchida. Caso seja ‘Entre’, são necessários os dois valores, com o menor deles na primeira coluna. Todas as condições incluem também seus valores limitadores, isto é, ‘Acima’ é o mesmo que ‘maior ou igual’.

As condições devem ser pensadas de acordo com as regiões e objetivos das campanhas, uma vez que a sensação de calor ou frio pode variar de acordo com a temperatura usual do local. Como a análise é feita de acordo com a temperatura máxima prevista para aquele dia, é importante que os limites sejam definidos considerando também a temperatura máxima usual.

A Tabela 6 ilustra um exemplo de uma tabela de condições preenchida. Apesar do nome das condições conter nomes de regiões, é possível utilizar uma mesma condição para diferentes regiões, se desejado. Devido aos valores de temperatura fornecidos pelo CPTEC/INPE serem sempre inteiros, é aconselhável que as entradas da planilha também sejam.

Tabela 6: Exemplo de planilha de condições

Condição	Tipo	Valor 1 (°C)	Valor 2 (°C)
Tropical quente	Acima	30	
Tropical normal	Entre	26	29
Tropical frio	Abaixo	25	
Subtropical quente	Acima	27	
Subtropical normal	Entre	22	26
Subtropical frio	Abaixo	21	

Fonte: Elaborada pelo autor.

3.1.5 Lances modificados

Semelhante à aba de condições descrita na subseção 3.1.4, nesta devem estar listadas as condições de lances modificados, ou seja, condições relacionadas aos resultados da campanha em si no Google Ads. Esta versão do script tem suporte para análise de dados de retorno sobre investimento publicitário (ROAS) e de quantia de conversões, sempre para os sete dias anteriores. Com alterações no código é possível utilizar outras diversas métricas da plataforma e até mesmo modificar o período de análise, sendo necessário apenas atenção ao limite de tempo de execução de scripts no Google Ads (trinta minutos).

Cada condição deve possuir um nome, métrica a ser analisada ('ROAS' ou 'Conversão', neste caso), tipo da condição ('Entre', 'Acima' ou 'Abaixo') e seus valores em números inteiros. A Tabela 7 ilustra o preenchimento desta aba com exemplos.

Tabela 7: Exemplo de planilha de condições modificadas

Condição	Métrica	Tipo	Valor 1	Valor 2
ROAS > 20	ROAS	Acima	20	
ROAS bom	ROAS	Entre	10	20
Converteu	Conversão	Acima	1	

Fonte: Elaborada pelo autor.

3.1.6 Histórico

Esta aba, diferente das demais, não deve ser preenchida pelo usuário. Trata-se de um histórico de temperatura que pode ser utilizado para análises de desempenho do script e até mesmo para refinar os limites de temperatura das condições de cada região. Apenas três colunas são utilizadas: data, nome da cidade e temperatura máxima do dia. A Tabela 8 exemplifica um exemplo de registro do histórico.

Tabela 8: Exemplo de planilha de histórico

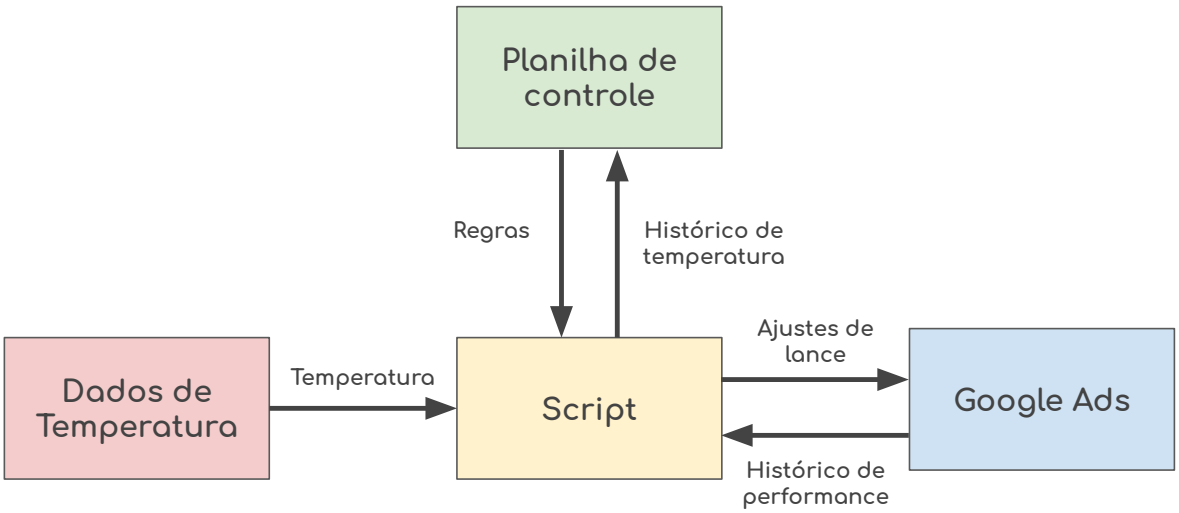
Data	Cidade	Temperatura Máxima
2020-3-7	Macae	28
2020-3-7	Niteroi	26
2020-3-7	Nova Friburgo	24
2020-3-7	Nova Iguacu	26

Fonte: Elaborada pelo autor. Dados de temperatura obtidos pelo CPTEC/INPE através da execução do código deste trabalho na primeira semana de março de 2020.

3.2 Código do script

Para cumprir com o objetivo da monografia, desenvolveu-se um script capaz de receber informações de uma fonte de dados de temperatura, no caso o CPTEC/INPE, e do próprio Google Ads; e interpretar estes dados seguindo as regras definidas pela planilha de controle desenvolvida na seção 3.1. Como resultado, o código retorna alterações em ajustes de lance nas campanhas desejadas do Google Ads e também registra as temperaturas lidas na aba "Histórico" da planilha de controle. A Figura 4 ilustra a troca de informações que ocorre durante seu funcionamento.

Figura 4: Esquema de troca de informações do código em funcionamento



Fonte: Elaborada pelo autor.

Nesta seção será explicada cada etapa do código desenvolvido, explicitando também possíveis personalizações do mesmo para outras implementações. O código-fonte encontra-se no Apêndice A.

3.2.1 Variáveis e parâmetros globais

Para tornar o código de fácil leitura e personalização, foram criadas variáveis globais que definem informações importantes para seu funcionamento. A única exceção está no objeto global "WEATHER_CACHE", que inicia-se vazio e armazena os valores de temperatura coletados pela função *getWeather* (subseção 3.2.2.2) para cada cidade já analisada, poupando assim requisições repetidas desnecessárias aos dados do CPTEC/INPE.

A única lista global é a "ACC_ID", responsável por armazenar o número de identificação das contas do Google Ads a serem alteradas pelo script. Devido a isso, o código deve ser executado sempre em uma MCC (Minha Central de Contas). É necessário que o usuário tenha permissão de edição de lances (acesso padrão) na conta utilizada para hospedar o código.

As demais variáveis são referentes à planilha de controle elaborada na seção 3.1 e estão explicadas a seguir:

a) "SPREADSHEET_ID" recebe uma *string* com o código da planilha do Google Planilhas a ser utilizada, com as seguintes variáveis relacionadas a ela:
   - "CTRL_CAMPAIGN_CATEGORY_COLUMN" recebe o índice da coluna na qual as categorias das campanhas estão listadas;
   - "CTRL_REGION_NAME_COLUMN" recebe o índice da coluna na qual as regiões estão listadas;
   - "CTRL_CONDITION_COLUMN" recebe o índice da coluna na qual as condições climáticas estão listadas;
   - "CTRL_BID_CONDITION_COLUMN" recebe o índice da coluna na qual as condições de lances modificados estão listadas;
   - "CTRL_REGULAR_BID_COLUMN" recebe o índice da coluna na qual estão listados os lances a serem aplicados caso apenas as condições climáticas sejam satisfeitas;
   - "CTRL_MODIFIED_BID_COLUMN" recebe o índice da coluna na qual estão listados os lances a serem aplicados caso as condições climáticas e as condições de ajustes modificados sejam satisfeitas simultaneamente.

b) "CONTROL_SHEET" recebe uma *string* com o nome da aba de controle;

c) "WEATHER_CONDITIONS_SHEET" recebe uma *string* com o nome da aba de condições climáticas, com as seguintes variáveis relacionadas a ela:
   - "WEA_CONDITION_NAME_COLUMN" recebe o índice da coluna na qual o nome das condições climáticas estão listados;
   - "WEA_CONDITION_COLUMN" recebe o índice da coluna na qual o tipo de condição está descrito;

- "WEA_TEMP1_COLUMN" recebe o índice da coluna na qual as primeiras temperaturas de cada condição estão listadas;
- "WEA_TEMP2_COLUMN" recebe o índice da coluna na qual as temperaturas secundárias de cada condição, se existentes, estão listadas.

d) "BIDS_CONDITIONS_SHEET" recebe uma *string* com o nome da aba de condições para lances modificados, com as seguintes variáveis relacionadas a ela:

- "BID_CONDITION_NAME_COLUMN" recebe o índice da coluna na qual o nome das condições de lance modificados estão listados;
- "BID_TYPE_COLUMN" recebe o índice da coluna na qual a métrica a ser analisada por cada condição é listada;
- "BID_CONDITION_COLUMN" recebe o índice da coluna na qual o tipo de condição está descrito;
- "BID_VALUE1_COLUMN" recebe o índice da coluna na qual os primeiros valores de cada condição estão listadas;
- "BID_VALUE2_COLUMN" recebe o índice da coluna na qual os valores secundários de cada condição, se existentes, estão listados.

e) "LOCATION_SHEET" recebe uma *string* com o nome da aba contendo as informações de locais a serem utilizados, com as seguintes variáveis relacionadas a ela:

- "CITY_NAME_COLUMN" recebe o índice da coluna na qual as cidades estão listadas;
- "ADS_CODE_COLUMN" recebe o índice da coluna na qual os códigos de identificação do Google Ads para cada cidade estão listados;
- "CPTEC_CODE_COLUMN" recebe o índice da coluna na qual os códigos de identificação do CPTEC/INPE para cada cidade estão listados;
- "REGION_NAME_COLUMN" recebe o índice da coluna na qual são determinadas as regiões de cada cidade.

f) "CAMPAIGN_SHEET" recebe uma *string* com o nome da aba contendo as informações das campanhas de anúncio a serem utilizadas, com as seguintes variáveis relacionadas a ela:

- "CAMPAIGN_NAME_COLUMN" recebe o índice da coluna na qual os nomes das campanhas estão localizados;
- "CAMPAIGN_CATEGORY_COLUMN" recebe o índice da coluna na qual as categorias das campanhas estão localizadas.

g) "HISTORY_SHEET" recebe uma *string* com o nome da aba no qual o histórico de temperatura obtido pelos dados do CPTEC/INPE será armazenado.

Desta forma, é possível alterar a ordem das colunas e o nome das abas da planilha de controle sem grandes problemas, bastando modificar as variáveis correspondentes no script.

3.2.2 Leitura e organização dos dados

A primeira etapa do script realiza a leitura dos dados inseridos pelo usuário na planilha de controle elaborada na seção 3.1, organizando-os para serem utilizados nas etapas posteriores.

3.2.2.1 Criação do mapa de regras

Para elaborar o mapa de regras que será utilizado na determinação dos ajustes de lance, foi criada a função *getRulesMap*. Cada regra elaborada possui as seguintes informações:

a) "campaignCategory", contendo o nome da categoria da campanha a qual essa regra pertence;

b) "campaigns", contendo uma lista com todas as campanhas associadas à regra;

c) "region", contendo o nome da região a qual essa regra pertence;

d) "cities", contendo uma lista com todas as cidades pertencentes à região de análise que estão dentro da condição climática associada à regra;

e) "climate", contendo a condição climática a qual essa regra pertence;

f) "climateConditions", contendo as informações da condição climática associada à regra;

g) "convertionType", contendo o nome da condição de ajuste modificado que deve ser analisado;

h) "convertionCondition", contendo as informações da condição de ajuste modificado associada à regra;

i) "standardBid", contendo o ajuste de lance padrão da regra;

j) "modifiedBid", contendo o ajuste de lance modificado da regra.

Para que todos esses dados sejam obtidos, são utilizadas as funções auxiliares a seguir.

Abertura e leitura de planilha

A função auxiliar *openAndReadSheet* é responsável por abrir e ler uma aba de uma planilha do Google Planilhas. Para isso, são utilizados os métodos *openById* e *getSheetByName* apresentados na subseção 2.2.1.1, determinando assim qual aba deve ser

lida, e os métodos *getDataRange* e *getValues* para se obter uma matriz contendo todos os dados daquela aba, que são retornados pela função.

Mapeamento de campanhas e regiões

Para realizar o mapeamento de campanhas e regiões climáticas, são utilizadas as funções *getCampaignMap* e *getRegionMap*, respectivamente. Ambas funções são semelhantes e servem para organizar os dados de categorias de campanhas e de regiões em objetos nos quais as chaves são os nomes de cada item (categoria ou região) e seus valores são uma lista de objetos contendo as informações detalhadas de cada elemento pertencente a essa categoria ou região.

Obtenção das condições

As funções *getClimateConditions* e *getModifiedConditions* são semelhantes entre si e servem para organizar as condições de temperatura e de ajustes modificados baseados no histórico de resultados, respectivamente, obtidas pela leitura da planilha de controle. Ambas retornam um objeto contendo essas informações.

Obtenção das temperaturas de cada cidade

Para realizar a obtenção das temperaturas lidas do CPTEC/INPE é utilizada a função *getCitiesRegionTemperature*, a qual recebe uma lista de objetos contendo as informações das cidades que deseja-se obter as temperaturas máximas previstas para aquele dia. Estas informações são organizadas para chamar a função *getWeather* (subseção 3.2.2.2) para cada uma das cidades desejadas.

Ao final, é retornada uma cópia da lista de objetos recebida inicialmente, porém contendo os dados de temperatura máxima.

Análise da condição de temperatura

A análise para determinar se uma cidade encontra-se ou não em uma determinada condição de temperatura é realizada pela função *checkCityWeather*. A partir da lista de objetos obtidas pela função *getCitiesRegionTemperature* e das informações de condição climática, é retornada uma lista de cidades que satisfazem a condição climática analisada em questão.

3.2.2.2 Leitura da previsão do tempo do CPTEC/INPE

Para realizar a leitura dos dados fornecidos pelo XML do CPTEC/INPE apresentado na seção 2.3, foi criada a função *getWeather*, que recebe como argumento o nome da cidade e o código CPTEC/INPE referente a ela.

Inicialmente, é verificado se a cidade está no objeto global “WEATHER_CACHE”, responsável por armazenar os dados de cidades já lidas pelo script, conforme consta na subseção 3.2.1. Em caso verdadeiro, o valor é consultado diretamente deste objeto e

retornado, evitando requisições repetidas ao CPTEC/INPE.

Se a cidade ainda não tiver sido lida pelo código, é iniciada a preparação para obter-se a temperatura através do XML do CPTEC/INPE. Para isso, é preparada uma *string* contendo a URL do XML correspondente à cidade através da substituição do termo "%s" na URL <http://servicos.cptec.inpe.br/XML/cidade/%s/previsao.xml> pelo código CPTEC/INPE, conforme as instruções presentes na subseção 2.3.2.

Para ser possível navegar pelo XML do CPTEC/INPE utilizando os métodos da classe XmlService descritos em subseção 2.2.1.3, necessita-se que ele possua um elemento raiz, o que não é obtido diretamente pelo XML do CPTEC. Devido a isso, a função lê inicialmente o XML em tempo utilizando o método *fetch* da classe UrlFetchApp e, após isso, é adicionado manualmente a indicação do elemento raiz. Por fim, utilizando o método *parse* é possível criar um novo XML que pode ser percorrido como desejado.

Na sequência, obtém-se o primeiro elemento "maxima" do primeiro elemento "previsao", que por sua vez está contido no primeiro elemento "cidade" do elemento raiz gerado. Este é o valor de temperatura máxima previsto para aquele dia. É importante ressaltar que este script foi desenvolvido para analisar os dados do XML às oito horas da manhã de cada dia, horário em que as informações ainda estão disponíveis para aquela data. É possível que em horários posteriores apenas a previsão do dia seguinte esteja disponível.

A temperatura máxima é então retornada pela função e também armazenada no objeto "WEATHER_CACHE" para a cidade correspondente.

3.2.3 Definição de ajustes de lance baseada em temperatura

Com as regras e os dados de temperatura obtidos corretamente, inicia-se o procedimento para determinar e alterar o ajuste de lance de cada cidade para cada campanha. A principal função desta etapa é a *setStandardBids*, a qual deve receber as regras mapeadas pela subseção 3.2.2.1. Esta função realiza a leitura de cada regra uma a uma e prepara a modificação dos ajustes de lance relacionado a ela, utilizando as funções auxiliares descritas a seguir.

Obtenção da lista de códigos do Google Ads das cidades alvo

É necessário obter uma lista contendo todos os códigos do Google Ads referentes às cidades que terão seus ajustes de lance alterados. Para isso, foi criada a função *getAdsCode* que recebe a lista de cidades vinda do mapeamento de regras e extrai apenas a informação do código do Google Ads de cada uma delas, retornando-as em uma lista.

Geração da lista de códigos de campanha do Google Ads

Para se fazer as alterações nas campanhas corretas, é necessário traduzir o nome delas para seus códigos de identificação, conhecidos como *CampaignId*. Para isso, deve-se

gerar um relatório do Google Ads a partir dos métodos descritos em subseção 2.2.2.1 selecionando *CampaignId* de do relatório de performance à nível de campanha "CAMPAIGN_PERFORMANCE_REPORT" no qual o nome da campanha corresponde a algum da lista passada como parâmetro na chamada da função *getCampaignID*, responsável for realizar esta interação.

Ao final, a função trata a informação obtida disponibilizando-a em uma lista de códigos de identificação de campanha que é retornada por ela.

Associação das listas de códigos

A função *getTargetedLocationIds* tem como objetivo associar os códigos de identificação de cada campanha para cada código de localização do Google Ads em uma lista de listas, cada uma contendo apenas dois elementos (ID de campanha e ID de local). Esta estrutura é utilizada para facilitar a alteração de ajuste de local realizada na sequência.

Alteração do ajuste de lance

Após chamar as funções listadas anteriormente, a função *setStandardBids* inicia a preparação para atualizar os ajustes de lance criando um iterador para locais que estão dentro da lista produzida pela função *getTargetedLocationIds* utilizando os métodos do AdsApp para *targeting*, conforme explicado na subseção 2.2.2.2.

Percorrendo-se o iterador, é elaborada uma lista contendo todos os locais que deseja-se alterar o lance, chamando assim a função *setBid*, que irá percorrer toda a lista aplicando o valor de lance desejado através do método *setBidModifier*. Desta forma, o lance baseado na temperatura é inserido para cada local e campanha correspondente, restando agora analisar o ajuste de lance modificado.

3.2.4 Definição de ajustes de lance baseado em histórico de performance

Determinado o ajuste com base na temperatura, é iniciado o processo de verificação dos ajustes modificados. Diferente do anterior, só são alterados os locais e campanhas que atendem os requisitos impostos pelo ajuste modificado. Este tipo de ajuste foi idealizado como um bônus para situações em que a performance possui bom histórico, porém o usuário pode configurar as regras como achar melhor.

A função principal deste passo é a *setModifiedBids*, que recebe também as regras mapeadas. Inicialmente, ela chama a função *getCodeMap* com o objetivo de mapear todos os códigos de identificação da planilha novamente, porém utilizando o nome da cidade como chave ao invés do nome da região, como feito no mapeamento realizado pela subseção 3.2.2.1.

Feito isso, entra-se em uma sequência de repetições para cada uma das regras a serem analisadas, lendo inicialmente suas informações para então avaliar se foram satisfeitas ou não.

Avaliação das regras

Para avaliar as regras, primeiro deve-se extrair um relatório do Google Ads conforme apresentado em subseção 2.2.2.1 contendo as informações desejadas. Como as regras criadas são referentes à quantidade de conversão e ao ROAS, precisa-se obter as métricas conversão, valor de conversão e custo e utilizar as relações descritas em subseção 2.1.1. Desta forma, o relatório gerado deve possuir as seguintes características:

a) **selecionar** o código da campanha (*CampaignId*), conversões (*Conversions*), valor de conversão (*ConversionValue*), custo (*Cost*) e código da cidade (*CityCriteriaId*);

b) **retirado do** "GEO_PERFORMANCE_REPORT", o relatório de performance por local;

c) **no qual** o código da campanha e o código da cidade pertencem à regra em análise;

d) **com duração** referente aos últimos sete dias.

Com estes dados obtidos, necessita-se corrigir o valor de conversão devido ao seu formato incluir vírgulas como separadores de milhar. Para isso, utiliza-se o método *replace* para substituir globalmente as vírgulas por vazio. Para calcular o ROAS, apenas converte-se as *strings* de valor de conversão e custo para ponto flutuante e, então, dividi-se o primeiro pelo segundo.

Após isto, é chamada a função *checkModifiedCondition* que irá encaminhar corretamente a condição para análise da função *checkValue* com os parâmetros adaptados para a métrica escolhida na confecção da planilha de controle. Esta última função analisa se o valor lido no relatório está dentro dos limites definidos e retorna verdadeiro se estiver. Neste caso, o par campanha/local em questão é adicionado em uma lista destinada a receber o ajuste modificado relacionado à regra.

Por fim, segue-se a lógica de alteração do ajuste de lance descrita na subseção 3.2.3, de forma a sobrescrevê-lo.

3.2.5 Registro no histórico de temperatura

Para manter um registro das temperaturas lidas, o que possibilita realizar análises comparativas de performance, é necessário abrir novamente a planilha em questão e acessar a aba Histórico através do *SpreadsheetApp*, seguindo o que foi explicado em subseção 2.2.1.1.

Primeiramente, armazena-se na variável *nextRow* qual a posição atual da última linha com dado acrescida de um, indicando-se assim qual a linha que deve começar a receber os dados, e, na sequência, é montada uma matriz *vetor* contendo, em cada linha, uma cidade e sua respectiva temperatura lidas diretamente da variável WEATHER_CACHE.

Cria-se então um objeto *Date* que é desmembrado para se obter a data atual no formato 'YYYY-MM-DD', em que Y representa o ano, M o mês e D o dia. Este valor é inserido na primeira coluna a partir da linha indicada por *nextRow*, repetindo-se em quantia igual a de linhas presente em *vetor*.

Por fim, escreve-se nas duas seguintes colunas, também a partir da linha indicada em *nextRow*, toda a informação coletada por *vetor*, encerrando assim o registro do histórico e a execução do script.

4 RESULTADOS

Com a planilha de controle e o código elaborado, foram realizados testes finais para avaliar o funcionamento do projeto. Para isso, foram criadas três campanhas fictícias em uma conta teste do Google Ads, listadas na Tabela 9, sendo todas de lance manual e contendo como segmentação de local as capitais brasileiras.

Tabela 9: Campanhas criadas para o teste

Nome da campanha	Tipo da campanha
_S_ArCondicionado_BidbyWeather	Campanha de pesquisa
_S_OutletVerao_BidbyWeather	Campanha de pesquisa
_DN_ColecaoInverno_BidbyWeather	Campanha da Rede de Display

Fonte: Elaborada pelo autor.

Por não se tratarem de campanhas com investimento real, não há histórico de resultados a ser utilizado. Devido a isso, os testes foram divididos em duas etapas: a primeira considerando apenas os dados de temperatura com histórico nulo e a segunda considerando um histórico fictício lido de uma planilha.

4.1 Dados de temperatura com histórico nulo

Neste primeiro teste, a temperatura é lida diretamente dos dados do CPTEC/INPE para ser utilizada na definição do ajuste de temperatura, enquanto os dados de histórico são lidos do Google Ads, sendo assim todos nulos. Devido a isso, as condições de ajuste modificado foram criadas contemplando valores nulos em determinadas situações a fim de verificar o funcionamento correto do código, no entanto o relatório utilizado para receber os dados de histórico da campanha por local apenas apresenta os locais que tiveram alguma impressão no período.

4.1.1 Utilização da planilha de controle para determinar as regras

Utilizando a planilha criada durante a seção 3.1, as campanhas foram divididas em duas categorias distintas, denominadas “Calor” e “Frio” e as capitais em duas regiões, “Teste A” e “Teste B”, da seguinte forma:

a) **categoria Calor** para as campanhas _S_ArCondicionado_BidbyWeather e _S_OutletVerao_BidbyWeather;

b) **categoria Frio** para a campanha _DN_ColecaoInverno_BidbyWeather;

c) **região Teste A** para as capitais Aracaju, Belém, Belo Horizonte, Boa Vista, Brasília, Goiânia, João Pessoa, Macapá, Maceió, Manaus, Rio Branco, Rio de Janeiro, Salvador, São Luís e São Paulo;

d) **região Teste B** para as capitais Campo Grande, Cuiabá, Curitiba, Florianópolis, Fortaleza, Natal, Palmas, Porto Alegre, Porto Velho, Recife, Teresina e Vitória.

Foram criadas três condições climáticas, "Quente", "Normal" e "Frio", conforme indicado pela Tabela 10, e quatro condições de lances modificados, "ROAS > 10", "ROAS < 5", "Converteu" e "Não converteu", como mostrado pela Tabela 11.

Tabela 10: Condições de temperatura criadas para o teste

Condição	Tipo	Valor 1 (°C)	Valor 2 (°C)
Quente	Acima	29	
Normal	Entre	26	28
Frio	Abaixo	25	

Fonte: Elaborada pelo autor.

Tabela 11: Condições de lances modificados criadas para o teste

Condição	Métrica	Tipo	Valor 1	Valor 2
ROAS > 10	ROAS	Acima	10	
ROAS < 5	ROAS	Abaixo	5	
Converteu	Conversão	Acima	1	
Não converteu	Conversão	Abaixo	0	

Fonte: Elaborada pelo autor.

Com essas informações inseridas nas planilhas, elaborou-se a tabela principal de controle com ajustes diferentes para uma fácil identificação das regras acionadas. A Tabela 12 traz todas as regras elaboradas.

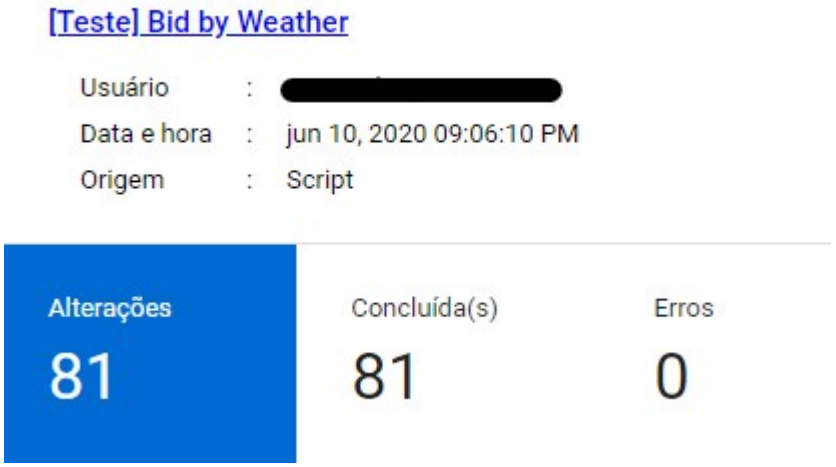
Com a planilha devidamente preenchida, alteraram-se os códigos da conta do Google Ads e da planilha de controle para seus valores corretos e executou-se o script. O tempo de execução foi de 1 minuto e 29 segundos, com 81 alterações, como evidenciado pela Figura 5.

Tabela 12: Planilha de controle elaborada para o primeiro teste

Categoria da campanha	Regiões	Condição climática	Condição de ajuste mod.	Ajuste padrão	Ajuste modificado
Calor	Teste A	Quente	Converteu	1.7	3.4
Calor	Teste A	Normal	Não Converteu	1.2	2.4
Calor	Teste A	Frio	ROAS $>$ 10	0.8	1.4
Calor	Teste B	Quente	ROAS $<$ 5	1.8	3.6
Calor	Teste B	Normal	Converteu	1.3	2.6
Calor	Teste B	Frio	Não Converteu	0.7	0.9
Frio	Teste A	Quente	ROAS $>$ 10	0.5	0.7
Frio	Teste A	Normal	ROAS $<$ 5	1.0	2.0
Frio	Teste A	Frio	Converteu	1.6	3.2
Frio	Teste B	Quente	Não Converteu	0.6	0.9
Frio	Teste B	Normal	Converteu	1.1	2.2
Frio	Teste B	Frio	ROAS $<$ 5	1.5	3.0

Fonte: Elaborada pelo autor.

Figura 5: Execução do script na plataforma do Google Ads para dados de temperatura



Fonte: Elaborada pelo autor a partir da plataforma do Google Ads.

4.1.2 Verificação da leitura de temperatura

Através dos dados preenchidos na aba Histórico pelo script, foi possível confirmar se o valor lido pelo código estava de acordo com o obtido pelo XML do CPTEC/INPE. Na Tabela 13 encontram-se todas as leituras realizadas no dia 10 de junho de 2020 referente ao dia seguinte, devido ao XML já ter removido a previsão para o mesmo dia no horário

em questão, enquanto a Figura 6 apresenta a leitura do XML para três dos municípios como forma de conferência dos resultados.

Tabela 13: Resultado obtido pela aba Histórico (dia 11 de junho de 2020)

Cidade	Temperatura (°C)	Cidade	Temperatura (°C)
Aracaju	30	Manaus	33
Belem	33	Natal	29
Belo Horizonte	27	Palmas	34
Boa Vista	31	Porto Alegre	22
Brasilia	28	Porto Velho	33
Campo Grande	31	Recife	29
Cuiaba	35	Rio Branco	32
Curitiba	26	Rio de Janeiro	32
Florianopolis	23	Salvador	28
Fortaleza	30	Sao Luis	30
Goiania	32	Sao Paulo	28
Joao Pessoa	29	Teresina	33
Macapa	32	Vitoria	32
Maceio	29		

Fonte: Elaborada pelo autor a partir das informações da previsão do tempo para municípios do CPTEC/INPE.

4.1.3 Verificação das alterações realizadas

Como não há dados referentes a nenhum dos locais para nenhuma das campanhas, o relatório de histórico gerado pelo código não apresentará nenhuma informação para eles, não permitindo assim que haja ajustes modificados. Desta forma, é possível predizer as alterações utilizando apenas as condições climáticas e os ajustes escolhidos para cada combinação de categoria de campanha e região com elas.

Dividindo-se cada cidade em sua correspondente condição climática (“Quente”, “Normal” ou “Frio”), obtém-se a Tabela 14, na qual a separação entre regiões está apresentada em parenteses.

Assim, é possível determinar previamente quais serão os 81 ajustes de lances aplicados para cada campanha:

a) **para as cidades na condição Frio**, o ajuste deve ser +50% nas campanhas “Frio” e -30% nas campanhas “Calor”;

Figura 6: Árvore de elementos do XML do CPTEC/INPE para Aracaju, Cuiabá e Porto Alegre

Fonte: Elaborada pelo autor a partir do XML com informações da previsão do tempo para municípios do CPTEC/INPE.

b) **para Curitiba**, o ajuste deve ser +10% nas campanhas “Frio” e +30% nas campanhas “Calor”;

c) **para as demais cidades na condição Normal**, o ajuste deve ser 0% nas campanhas “Frio” e +20% nas campanhas “Calor”;

d) **para as cidades da região Teste B na condição Quente**, o ajuste deve ser -40% nas campanhas “Frio” e +80% nas campanhas “Calor”;

e) **para as demais cidades na condição Quente**, o ajuste deve ser -50% nas campanhas “Frio” e +70% nas campanhas “Calor”.

Como evidenciado no histórico de execução do script apresentado nas Figura 7 e Figura 8, as alterações foram realizadas conforme desejado. Na Figura 9, é possível constatar que de fato as alterações foram registradas na interface de ajuste de local do Google Ads.

4.2 Dados de temperatura com histórico simulado

Para testar os ajustes modificados, foi utilizada uma aba auxiliar na planilha de controle. Nela, foram inseridos valores aleatórios de custo, valor de conversão e conversão para cada local em questão. Para este teste, foi utilizada apenas a categoria de campanha “Frio”.

Figura 7: Execução do script para campanhas da categoria “Frio”

Campanha	Grupo de anúncios	Alterar descrição	Status
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +70% Aracaju, Sergipe, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +70% Boa Vista, Roraima, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +70% Manaus, Amazonas, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +70% São Luís, Maranhão, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +70% Belém, Pará, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +20% Salvador, Bahia, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +20% Brasília, Distrito Federal, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +20% Belo Horizonte, Minas Gerais, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +20% São Paulo, São Paulo, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +80% Fortaleza, Ceará, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +80% Vitória, Espírito Santo, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +80% Teresina, Piauí, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +80% Campo Grande, Mato Grosso do Sul, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +80% Palmas, Paraná, Brasil	✓ Concluída(s)
_S_ArCondicionado_BidbyWeather		Ajuste de lance por local definido como +80% Porto Velho, Rondônia, Brasil	✓ Concluída(s)

Fonte: Elaborada pelo autor a partir da plataforma do Google Ads.

Figura 8: Execução do script para campanhas da categoria “Calor”

Campanha	Grupo de anúncios	Alterar descrição	Status
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% João Pessoa, Paraíba, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% Rio de Janeiro, Rio de Janeiro, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% Goiânia, Goiás, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% Aracaju, Sergipe, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% Boa Vista, Roraima, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% Manaus, Amazonas, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% São Luís, Maranhão, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -50% Belém, Pará, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como +0% Salvador, Bahia, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como +0% Brasília, Distrito Federal, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como +0% Belo Horizonte, Minas Gerais, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como +0% São Paulo, São Paulo, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -40% Fortaleza, Ceará, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -40% Vitória, Espírito Santo, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local definido como -40% Teresina, Piauí, Brasil	✓ Concluída(s)

Fonte: Elaborada pelo autor a partir da plataforma do Google Ads.

Tabela 14: Condição climática de cada cidade para o dia 11/06/2020

Frio	Normal	Quente
Porto Alegre (B)	Curitiba (B)	Joao Pessoa (A)
Florianopolis (B)	Belo Horizonte (A)	Maceio (A)
	Brasilia (A)	Natal (B)
	Salvador (A)	Recife (B)
	Sao Paulo (A)	Aracaju (A)
		Fortaleza (B)
		Sao Luis (A)
		Boa Vista (A)
		Campo Grande (B)
		Goiania (A)
		Macapa (A)
		Rio Branco (A)
		Rio de Janeiro (A)
		Vitoria (B)
		Belem (A)
		Manaus (A)
		Porto Velho (B)
		Teresina (B)
		Palmas (B)
		Cuiaba (B)

Fonte: Elaborada pelo autor a partir das informações da previsão do tempo para municípios do CPTEC/INPE.

4.2.1 Dados utilizados para simulação

Para simulação, foram escolhidos aleatoriamente os valores presentes na Tabela 15. Os valores não são coerentes entre si - há valor de conversão em cidades sem conversão - no entanto tomou-se essa liberdade para evidenciar o funcionamento do código para ambas condições.

4.2.2 Verificação das alterações desejadas

Com o código adaptado para receber o histórico via planilha, o script foi novamente executado, agora para as informações referentes ao dia 12 de junho. Com isso, os dados de temperatura foram alterados, conforme indicado na Tabela 16. O tempo de execução do script para apenas a categoria de campanha em questão foi de 29 segundos.

Nenhuma cidade neste dia esteve na condição “Frio”, restringindo as condições de ajuste modificado para as seguintes, conforme apresentado anteriormente pela Tabela 12:

a) caso a cidade esteja na **condição Quente** e na **região Teste A**, ela receberá o ajuste modificado apenas **se seu ROAS for superior a 10**;

Figura 9: Alterações aplicadas em uma campanha da categoria “Calor”

Região de segmentação	Ajuste de lance	Cliques	Impr.	CTR	CPC médio	Custo	Conversões
Aracaju, Sergipe, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
Belém, Pará, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
Belo Horizonte, Minas Gerais, Brasil	+20%	0	0	–	–	R$ 0,00	0,00
Boa Vista, Roraima, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
Brasília, Distrito Federal, Brasil	+20%	0	0	–	–	R$ 0,00	0,00
Campo Grande, Mato Grosso do Sul, Brasil	+80%	0	0	–	–	R$ 0,00	0,00
Cuiabá, Mato Grosso, Brasil	+80%	0	0	–	–	R$ 0,00	0,00
Curitiba, Paraná, Brasil	+30%	0	0	–	–	R$ 0,00	0,00
Florianópolis, Santa Catarina, Brasil	-30%	0	0	–	–	R$ 0,00	0,00
Fortaleza, Ceará, Brasil	+80%	0	0	–	–	R$ 0,00	0,00
Goiânia, Goiás, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
João Pessoa, Paraíba, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
Macapá, Amapá, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
Maceió, Alagoas, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
Manaus, Amazonas, Brasil	+70%	0	0	–	–	R$ 0,00	0,00
Natal, Rio Grande do Norte, Brasil	+80%	0	0	–	–	R$ 0,00	0,00

Fonte: Elaborada pelo autor a partir da plataforma do Google Ads.

b) caso a cidade esteja na **condição Normal** e na **região Teste A**, ela receberá o ajuste modificado apenas **se seu ROAS for inferior a 5**;

c) caso a cidade esteja na **condição Quente** e na **região Teste B**, ela receberá o ajuste modificado apenas **se não tiver conversões registradas**;

d) caso a cidade esteja na **condição Normal** e na **região Teste B**, ela receberá o ajuste modificado apenas **se tiver conversões registradas**.

Ao todo, ocorreram apenas 15 das possíveis 54 modificações (27 modificações via temperatura mais 27 via histórico) durante esta execução, como evidenciado pela Figura 10, o que indica que algumas cidades não tiveram seus valores alterados por não terem mudado o ajuste em relação ao anterior.

A Figura 11 traz alguns exemplos de alterações, mostrando inclusive a cidade de Porto Alegre sendo modificada duas vezes - primeiro devido à temperatura, e depois novamente devido ao seu histórico, aplicando assim o ajuste modificado de +120%. Identifica-se

Tabela 15: Dados utilizados para simulação de histórico

Cidade	Custo	Valor de Conversão	Conversões
Aracaju	77,87	1.342,01	0
Belem	85,24	1.453,78	0
Belo Horizonte	19,52	220,80	5
Boa Vista	49,09	906,63	2
Brasilia	79,59	1.520,99	0
Campo Grande	94,32	1.056,68	4
Cuiaba	93,47	120,88	3
Curitiba	42,54	349,53	4
Florianopolis	38,89	421,12	0
Fortaleza	21,34	157,28	0
Goiania	29,18	559,00	0
Joao Pessoa	13,64	93,78	2
Macapa	91,73	1.786,18	4
Maceio	51,48	457,70	4
Manaus	15,32	146,00	3
Natal	33,12	25,10	2
Palmas	43,52	832,76	1
Porto Alegre	54,75	514,03	4
Porto Velho	74,09	1.181,69	5
Recife	81,72	478,88	0
Rio Branco	79,20	29,71	0
Rio de Janeiro	48,06	250,75	1
Salvador	30,58	308,95	0
Sao Luis	3,39	42,06	0
Sao Paulo	32,84	126,44	2
Teresina	11,07	88,10	0
Vitoria	83,93	1.549,47	2

Fonte: Elaborada pelo autor.

também oito cidades recebendo ajuste de -30% devido à regra “ROAS > 10”, a cidade de São Paulo recebendo +100% devido a possuir um ROAS menor que 5 e Teresina e Recife recebendo -10% devido a não possuírem conversões.

Em relação ao tempo de execução do código, a execução em um teste realizado em

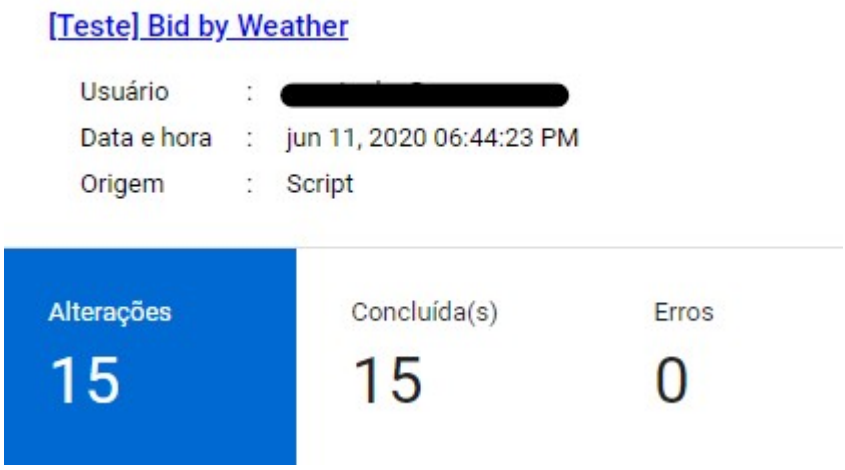
Tabela 16: Resultado obtido pela aba Histórico (dia 12 de junho de 2020)

Cidade	Temperatura (°C)	Cidade	Temperatura (°C)
Aracaju	30	Manaus	32
Belem	32	Natal	31
Belo Horizonte	29	Palmas	35
Boa Vista	31	Porto Alegre	27
Brasilia	27	Porto Velho	31
Campo Grande	30	Recife	30
Cuiaba	35	Rio Branco	31
Curitiba	27	Rio de Janeiro	32
Florianopolis	25	Salvador	28
Fortaleza	28	Sao Luis	31
Goiania	31	Sao Paulo	28
Joao Pessoa	30	Teresina	33
Macapa	31	Vitoria	32
Maceio	30		

Fonte: Elaborada pelo autor a partir das informações da previsão do tempo para municípios do CPTEC/INPE.

uma conta de anúncios com informações lidas diretamente do Google Ads, utilizando 72 regras para 28 campanhas e 127 cidades, demorou apenas 12 minutos, ou seja, menos da metade do tempo limite.

Figura 10: Execução do script na plataforma do Google Ads com simulação de histórico



Fonte: Elaborada pelo autor a partir da plataforma do Google Ads.

Figura 11: Alterações do script utilizando o histórico de resultados

Campanha	Grupo de anúncios	Alterar descrição	Status
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de +50% para +10% Porto Alegre, Rio Grande do Sul, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -50% para -30% Macapá, Amapá, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -50% para -30% Belém, Pará, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -50% para -30% Aracaju, Sergipe, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -50% para -30% Goiânia, Goiás, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -50% para -30% Boa Vista, Roraima, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -50% para -30% São Luís, Maranhão, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -50% para -30% Belo Horizonte, Minas Gerais, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de +0% para +100% São Paulo, São Paulo, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -40% para -10% Teresina, Piauí, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de -40% para -10% Recife, Pernambuco, Brasil	✓ Concluída(s)
_DN_ColecaoInverno_BidbyWeather		Ajuste de lance por local alterado de +10% para +120% Porto Alegre, Rio Grande do Sul, Brasil	✓ Concluída(s)

Fonte: Elaborada pelo autor a partir da plataforma do Google Ads.

5 CONCLUSÃO

Este trabalho iniciou-se com o propósito de cumprir o objetivo de utilizar a temperatura para influenciar o marketing digital, porém com fácil personalização e uso de outras informações para complementá-lo, o que foi concluído com sucesso como evidenciado pelos resultados apresentados.

Enquanto outras versões exigem que cada combinação de local e campanha possua uma regra manualmente configurada em uma planilha, a desenvolvida neste trabalho foi capaz de realizar 81 modificações utilizando apenas 12 regras, podendo realizar muito mais dependendo da complexidade desejada. Os agrupamentos que permitem essa simplificação foram criados para serem ao mesmo tempo intuitivos e relevantes para quem estiver configurando, sem necessitar de qualquer conhecimento de programação.

Ao mesmo tempo, foi acoplado ao código a possibilidade de utilizar o histórico de resultados de cada local para cada campanha, permitindo assim utilizar os dados de temperatura sem perder as otimizações baseadas em performance, que também são importantes.

O código produzido também permite sua expansão para outras funcionalidades que serão implementadas em passos futuros - tanto pode-se expandir a atuação utilizando a informação do tempo de cada local, quanto pode-se expandir para outras informações relacionadas à localização. Além disso, existe a oportunidade de incrementar seu funcionamento utilizando a lógica difusa, também conhecida como lógica *fuzzy*.

Para a análise do tempo, pode-se utilizar as outras informações disponíveis no XML do CPTEC/INPE, como a previsão de chuva e de incidência de radiação ultravioleta, dependendo da importância destas informações para o negócio. Ainda com temperatura, é praticável adicionar funções extras como ativar e pausar anúncios e até mesmo realizar o controle de orçamento a nível de campanha.

Já em relação a novos dados de localização, é possível acoplar qualquer tipo de dado relacionado às cidades desde que haja uma fonte de dados que possa ser acessada. Por exemplo, possuir uma fonte que indique a porcentagem de isolamento social em um momento de pandemia pode ser útil para guiar as campanhas de marketing digital de acordo com o objetivo: caso a campanha tenha interesse em visitas a uma loja física, seria interessante aumentar o lance em cidades com baixa taxa, enquanto que lojas eletrônicas tendem a ter melhor performance em locais com maior taxa de isolamento social, situação em que as compras são feitas prioritariamente pela internet.

No campo social, pode-se utilizar esta ferramenta para impulsionar anúncios de conscientização da população para o combate à pandemia em locais com baixa adesão

ao isolamento social. Outra aplicação possível está na gestão inteligente de anúncios para coleta de sangue próximos a hemocentros com deficiência em determinados tipos sanguíneos.

Ao contrário de otimizações baseadas exclusivamente em resultados, o uso de informações de geolocalização permite adicionar inteligência ao marketing digital com ajustes de forma preditiva, o que pode ser vantajoso contra a concorrência e permite aproveitar desde o início do dia as oportunidades relacionadas a esta informação.

REFERÊNCIAS

AJUDA do Google Ads. Google, 2020. Apresenta a documentação referente ao Google Ads, explicando definições e funcionamentos da plataforma. Disponível em: <https://support.google.com/google-ads/>. Acesso em: 03 jun. 2020.

BENGEL, A.; SHAWKI, A.; AGGARWAL, D. Simplifying web analytics for digital marketing. In: IEEE. **2015 IEEE International Conference on Big Data (Big Data)**. [S.l.], 2015. p. 1917–1918.

Chikara, A. et al. Study and implementation of charging battery of the car using pumping system and indicating the battery percentage present in fuel car. In: **2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)**. [S.l.: s.n.], 2020. p. 711–714.

COMO marketing digital pode ajudar a sua empresa nos tempos de coronavírus? Blog SEMrush, 2020. Disponível em: <https://pt.semrush.com/blog/marketing-digital-coronavirus/>. Acesso em: 02 jun. 2020.

GOOGLE Apps Script. Google Developers, 2020. Apresenta documentação de APIs e ferramentas de desenvolvimento para os aplicativos do Google, que podem ser utilizados no desenvolvimento de *scripts* em suas plataformas. Disponível em: <https://developers.google.com/apps-script>. Acesso em: 27 mai. 2020.

PAK, B. K. et al. Development of autonomous intelligent system for google ads. In: IEEE. **2018 Thirteenth International Conference on Digital Information Management (ICDIM)**. [S.l.], 2018. p. 102–107.

PREVISAO de Tempo em XML - CPTEC/INPE. Centro de Previsão de Tempo e Estudos Climáticos do Instituto Nacional de Pesquisas Espaciais, 2014. Apresenta documentação para utilização da previsão de tempo em XML do CPTEC. Disponível em: <http://servicos.cptec.inpe.br/XML/>. Acesso em: 06 jun. 2020.

VERAO faz crescer mais de 70% as vendas de ventilador e provoca faltas pontuais. O Estado de São Paulo, 2019. Disponível em: <https://economia.estadao.com.br/noticias/geral,verao-intenso-impulsiona-vendas-de-ventilador-e-ar-condicionado,70002683261>. Acesso em: 02 jun. 2020.

WEATHER-BASED Campaign Management. Google Developers, 2019. Apresenta um gerenciamento de campanhas baseado em temperatura, utilizando dados do OpenWeatherMap. Disponível em: <https://developers.google.com/google-ads/scripts/docs/solutions/weather-based-campaign-management>. Acesso em: 02 jun. 2020.

Apêndices

APÊNDICE A – CÓDIGO FONTE

```
/*
 Copyright (c) 2020 RACCOON MARKETING DIGITAL

 Permission is hereby granted, free of charge, to any person
 obtaining a copy of this software and associated documentation files
 (the "Software"), to deal in the Software without restriction,
 including without limitation the rights to use, copy, modify, merge,
 publish,  distribute, sublicense, and/or sell copies of the Software,
 and to permit persons to whom the Software is furnished to do so,
 subject  to the following conditions:

 The above copyright notice and this permission notice shall be
 included in all copies or substantial portions of the Software.

 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 SOFTWARE.
*/

/**
 * Este código tem como finalidade realizar alterações em ajustes
 * de lance de local no Google Ads através da leitura da temperatura
 * máxima prevista para o dia, utilizando também os resultados ante-
 * riores como entrada.
 **/
function main() {
  MccApp.accounts().withIds(ACC_ID).executeInParallel('process')
}

function process(){
var rulesMap = getRulesMap();
```

```
    setStandardBids(rulesMap);
    setModifiedBids(rulesMap);
    setHistory();
}

/**
 * Definições globais. Referir-se ao Capítulo 3 da monografia para
 * informações detalhadas de cada uma delas.
 **/
//The script is expected to work with following API version
API_VERSION = {
apiVersion: 'v201809'
}

ACC_ID = ["826-177-5681"];
SPREADSHEET_ID = "1MpF33hgF5qh484UOKVcz06IdjyabrK5w-JuStMnjhzM";
CONTROL_SHEET = "Controle";
WEATHER_CONDITIONS_SHEET = "Condições";
BIDS_CONDITIONS_SHEET = "Lances modificados";
LOCATION_SHEET = "Locais";
CAMPAIGN_SHEET = "Campanhas";
HISTORY_SHEET = "Histórico";

WEATHER_CACHE = {};

// CAMPAIGN_SHEET
CAMPAIGN_NAME_COLUMN = 0;
CAMPAIGN_CATEGORY_COLUMN = 1;

// LOCATION_SHEET
CITY_NAME_COLUMN = 0;
ADS_CODE_COLUMN = 1;
CPTEC_CODE_COLUMN = 2;
REGION_NAME_COLUMN = 3;

// WEATHER_CONDITIONS_SHEET
WEA_CONDITION_NAME_COLUMN = 0;
WEA_CONDITION_COLUMN = 1;
WEA_TEMP1_COLUMN = 2;
```

```
WEA_TEMP2_COLUMN = 3;

// BIDS_CONDITIONS_SHEET
BID_CONDITION_NAME_COLUMN = 0;
BID_TYPE_COLUMN = 1;
BID_CONDITION_COLUMN = 2;
BID_VALUE1_COLUMN = 3;
BID_VALUE2_COLUMN = 4;

// CONTROL_SHEET
CTRL_CAMPAIGN_CATEGORY_COLUMN = 0;
CTRL_REGION_NAME_COLUMN = 1;
CTRL_CONDITION_COLUMN = 2;
CTRL_BID_CONDITION_COLUMN = 3;
CTRL_REGULAR_BID_COLUMN = 4;
CTRL_MODIFIED_BID_COLUMN = 5;



/**
 * Mapeia todas as regras escritas na planilha de controle.
 *
 * Retorna as informações detalhadas de cada regra.
 **/
function getRulesMap(){
  var rulesMap = {};

  var controlSheetData = openAndReadSheet(CONTROL_SHEET);
  var weatherConditionsSheetData =
                    openAndReadSheet(WEATHER_CONDITIONS_SHEET);
  var locationSheetData = openAndReadSheet(LOCATION_SHEET);
  var campaingSheetData = openAndReadSheet(CAMPAIGN_SHEET);
  var bidsSheetData = openAndReadSheet(BIDS_CONDITIONS_SHEET);

  var campaignMap = getCampaignMap(campaingSheetData);
  var regionMap = getRegionMap(locationSheetData)
  var climateConditionsMap =
                  getClimateConditions(weatherConditionsSheetData);
  var convertionMap = getModifiedConditions(bidsSheetData);
```

```
  for(var i = 1; i < controlSheetData.length; i++){
    var campaignCategory =
             controlSheetData[i][CTRL_CAMPAIGN_CATEGORY_COLUMN];
    var region = controlSheetData[i][CTRL_REGION_NAME_COLUMN];
    var climate = controlSheetData[i][CTRL_CONDITION_COLUMN];
    var convertionCondition =
                controlSheetData[i][CTRL_BID_CONDITION_COLUMN];

    var campaigns = campaignMap[campaignCategory];
    var climateConditions = climateConditionsMap[climate];

    var cities = getCitiesRegionTemperature(regionMap[region]);
    var checkedCities = checkCityWeather(cities, climateConditions);

    rulesMap[i] = {
      'campaignCategory': campaignCategory,
      'campaigns': campaigns,
      'region': region,
      'cities': checkedCities,
      'climate': climate,
      'climateConditions': climateConditions,
      'convertionType': convertionCondition,
      'convertionCondition': convertionMap[convertionCondition],
      'standardBid':  controlSheetData[i][CTRL_REGULAR_BID_COLUMN],
      'modifiedBid':  controlSheetData[i][CTRL_MODIFIED_BID_COLUMN]
    }
  }
  return rulesMap;
}

/**
 * Abre uma planilha e lê todos os dados de uma aba.
 *
 * Parâmetros: sheetname -> nome da aba a ser lida.
 *
 * Retorna todos os dados presentes na aba.
 **/
function openAndReadSheet(sheetName){
  var sheet = SpreadsheetApp
```

```
              .openById(SPREADSHEET_ID)
       .getSheetByName(sheetName);
  var data = sheet.getDataRange().getValues();
  return data
}

/**
 * Mapeia todas as campanhas a serem modificadas pelo script.
 *
 * Parâmetros: data -> dados de uma seleção da aba de campanhas.
 *
 * Retorna as informações de cada campanha.
 **/
function getCampaignMap(data){
  var campaignMap = {};

  for(var i = 1; i < data.length; i++){
    var campaignName = data[i][CAMPAIGN_NAME_COLUMN];
    var campaignCategory = data[i][CAMPAIGN_CATEGORY_COLUMN];

    if(!campaignMap[campaignCategory]){
      campaignMap[campaignCategory] = [];
    }
    if(campaignMap[campaignCategory]){
      campaignMap[campaignCategory].push(campaignName);
    }
  }
  return campaignMap;
}

/**
 * Mapeia todas as regiões (agrupamentos de cidades).
 *
 * Parâmetros: data -> dados de uma seleção da aba de região.
 *
 * Retorna as informações de cada região.
 **/
function getRegionMap(data){
  var regionsMap = {};
```

```
  for(var i = 1; i < data.length; i++){
    var RegionName = data[i][REGION_NAME_COLUMN];

    var region = {
      'city': data[i][CITY_NAME_COLUMN],
      'cptecCode': data[i][CPTEC_CODE_COLUMN],
      'adsCode': data[i][ADS_CODE_COLUMN],
    }

    if(!regionsMap[RegionName]){
      regionsMap[RegionName] = [];
    }
    if(regionsMap[RegionName]){
      regionsMap[RegionName].push(region);
    }
  }
  return regionsMap;
}

/**
 * Mapeia todas as condições de temperatura.
 *
 * Parâmetros: data -> dados de uma seleção da aba de condições.
 *
 * Retorna as informações de cada condição de temperatura.
 **/
function getClimateConditions(data){
  var weatherConditionsMap = {};

  for(var i = 1; i < data.length; i++){
    var weatherConditionName = data[i][WEA_CONDITION_NAME_COLUMN];

    weatherConditionsMap[weatherConditionName] = {
      'condition': data[i][WEA_CONDITION_COLUMN],
      't1': data[i][WEA_TEMP1_COLUMN],
      't2': data[i][WEA_TEMP2_COLUMN]
    }
  }
```

```
  return weatherConditionsMap;
}

/**
 * Mapeia todas as condições de ajuste modificado.
 *
 * Parâmetros: data -> dados de uma seleção da aba de condições.
 *
 * Retorna as informações de cada condição de ajuste modificado.
 **/
function getModifiedConditions(data){
  var modifiedMap = {};

  for(var i = 1; i < data.length; i++){
    var modifiedName = data[i][BID_CONDITION_NAME_COLUMN];

    modifiedMap[modifiedName] = {
      'type': data[i][BID_TYPE_COLUMN],
      'condition': data[i][BID_CONDITION_COLUMN],
      'value1': data[i][BID_VALUE1_COLUMN],
      'value2': data[i][BID_VALUE2_COLUMN]
    }
  }
  return modifiedMap;
}

/**
 * Acrescenta a temperatura máxima do dia a um objeto de informação
 * de cidades.
 *
 * Parâmetros: cities -> lista de objetos com informações de cada
 * cidade.
 *
 * Retorna uma cópia do parâmetro de entrada, com acréscimo da infor-
 * mação de temperatura.
 **/
function getCitiesRegionTemperature(cities){
  for(var i = 0; i < cities.length; i++){
    var city = cities[i]['city'];
```

```
    var cptecCode = cities[i]['cptecCode'];
    var maxTemp = getWeather(city, cptecCode);
    cities[i]['maxTemp'] = maxTemp;
  }
  return cities
}

/**
 * Checa se a temperatura das cidades estão dentro dos limites da
 * regra analisada.
 *
 * Parâmetros: cities -> lista de objetos com info. de cidades.
 *             climateConditions -> informações da condição de tem-
 *             peratura.
 *
 * Retorna uma lista de cidades que satisfazem as condições.
 **/
function checkCityWeather(cities, climateConditions){
  var cityList = [];

  for(var i = 0; i < cities.length; i++){
    if(climateConditions['condition'] == 'Acima'){
      if(cities[i]['maxTemp'] >= climateConditions['t1']){
        cityList.push(cities[i]);
      }
    }

    else if(climateConditions['condition'] == 'Abaixo'){
      if(cities[i]['maxTemp'] <= climateConditions['t1']){
        cityList.push(cities[i]);
      }
    }

    else if(climateConditions['condition'] == 'Entre'){
      if(cities[i]['maxTemp'] <= climateConditions['t2'] &&
     cities[i]['maxTemp'] >= climateConditions['t1']){
        cityList.push(cities[i]);
      }
    }
```

```
  }
  return cityList;
}

/**
 * Obtém a temperatura máxima de uma cidade.
 *   - Caso a cidade não esteja em WEATHER_CACHE, a leitura é feita
 *   pelo XML do CPTEC/INPE e, depois armazenada em WEATHER_CACHE.
 *   - Caso a cidade já esteja em WEATHER_CACHE, a leitura da tempe-
 *   ratura é feita através do WEATHER_CACHE.
 *
 * Parâmetros: city -> nome da cidade;
 *             cptecCode -> códico CPTEC/INPE da cidade.
 *
 * Retorna a temperatura máxima para aquela cidade.
 **/
function getWeather(city, cptecCode){
  if(!(city in WEATHER_CACHE)){
    var cptecUrl = Utilities.formatString(
"http://servicos.cptec.inpe.br/XML/cidade/%s/previsao.xml",
cptecCode);
    var response = UrlFetchApp
               .fetch(cptecUrl, {"muteHttpExceptions":true})
   .getContentText();
    var newCont = response.replace(
"<?xml version='1.0' encoding='ISO-8859-1'?>",
"<?xml version='1.0' encoding='ISO-8859-1'?><root>");
newCont += "</root>";
    var xml = XmlService.parse(newCont);
    var maxTemp = xml
              .getRootElement()
              .getChildren("cidade")[0]
              .getChildren("previsao")[0]
  .getChildren("maxima")[0]
  .getValue();
    WEATHER_CACHE[city] = maxTemp;
  }
  else{
    maxTemp = WEATHER_CACHE[city];
```

```
  }
  return maxTemp;
}

/**
 * Define os lances baseados em dados de temperatura.
 *
 * Parâmetros: rulesMap -> regras mapeadas.
 **/
function setStandardBids(rulesMap){
  for(var rule in rulesMap){
    var targetList = [];

// Leitura da regra atual para análise
    var campaigns = rulesMap[rule]['campaigns'];
    var cities = rulesMap[rule]['cities'];
    var bid = parseFloat(rulesMap[rule]['standardBid']);
    var campaignCategory = rulesMap[rule]['campaignCategory'];
    var region = rulesMap[rule]['region'];
    var climate = rulesMap[rule]['climate'];
    var convertionCondition = rulesMap[rule]['convertionCondition'];

var locationIds = getAdsCode(cities);
    var campaignsIds = getCampaignID(campaigns);
    var targetedLocationIds = getTargetedLocationIds(campaignsIds,
                                                     locationIds);
    if(bid && cities){
      var adsIterator = AdsApp.targeting()
      .targetedLocations()
      .withIds(targetedLocationIds)
      .get();

      while(adsIterator.hasNext()){
        var target = adsIterator.next();
        targetList.push(target);
      }
      setBid(targetList, bid);
    }
  }
```

```
}

/**
 * Obtém uma lista das IDs de uma lista de objetos de cidades.
 *
 * Parâmetros: cities -> lista de objetos com dados de cidades.
 *
 * Retorna uma lista de códigos do Google Ads de local.
 **/
function getAdsCode(cities){
  var adsCodeList = [];
  for(var i = 0; i < cities.length; i++){
    adsCodeList.push(cities[i]['adsCode']);
  }
  return adsCodeList;
}

/**
 * Obtém lista de IDs de uma lista de campanhas.
 *
 * Parâmetros: campaigns -> lista de nomes de campanhas.
 *
 * Retorna uma lista de IDs de campanhas.
 **/
function getCampaignID(campaigns){
  var campaignsId = [];
  var report = AdsApp.report(
    'SELECT CampaignId ' +
    'FROM   CAMPAIGN_PERFORMANCE_REPORT ' +
    'WHERE ' +
    'CampaignName IN ' + '[' + campaigns + ']'; API_VERSION);

  var rows = report.rows();

  while (rows.hasNext()) {
    var row = rows.next();
    var campaignId = row['CampaignId'];
    campaignsId.push(campaignId);
  }
```

```
  return campaignsId;
}

/**
 * Associa todas as localizações para cada uma das campanhas.
 *
 * Parâmetros: campaignsIds -> lista de IDs de campanhas;
 *             locationIds -> lista de IDs de local.
 *
 * Retorna uma lista de listas com essas IDs pareadas.
 **/
function getTargetedLocationIds(campaignsIds, locationIds){
  var targetedLocationIds = []
  for(var i = 0; i < campaignsIds.length; i++){
    for(var j = 0; j < locationIds.length; j++){
      targetedLocationIds.push([campaignsIds[i], locationIds[j]]);
    }
  }
  return targetedLocationIds;
}

/**
 * Altera o lance em uma lista de pares locais/campanha.
 *
 * Parâmetros: targetList -> lista com IDs pareadas;
 *             bid -> ajuste a ser aplicado.
 **/
function setBid(targetList, bid){
  for(var i = 0; i < targetList.length; i++){
    targetList[i].setBidModifier(bid);
  }
}

/**
 * Define os lances baseados em dados de histórico.
 *
 * Parâmetros: rulesMap -> regras mapeadas.
 **/
function setModifiedBids(rulesMap){
```

```
  var cityCodes = getCodeMap();
  for(var rule in rulesMap){
    var targetList = [];

// Leitura da regra atual
    var convertionCondition = rulesMap[rule]['convertionCondition'];
    var campaigns = rulesMap[rule]['campaigns'];
    var cities = rulesMap[rule]['cities'];
    var bid = parseFloat(rulesMap[rule]['modifiedBid']);
    var campaignCategory = rulesMap[rule]['campaignCategory'];
    var region = rulesMap[rule]['region'];
    var climate = rulesMap[rule]['climate'];

    if (cities.length > 0){
      var checkedCities = runReport(campaigns, cities,
                                convertionCondition, cityCodes);

      if(bid && cities){
        for (var campaign in checkedCities){
          var adsIterator = AdsApp.targeting()
          .targetedLocations()
          .withIds(checkedCities[campaign])
          .get();

          while(adsIterator.hasNext()){
            var target = adsIterator.next();
            targetList.push(target);
          }
          setBid(targetList, bid);
        }
      }
    }
  }
}

/**
 * Mapeia todos os códigos do Google Ads da planilha para facilitar
 * a pesquisa através do nome da cidade.
 *
```

```
 * Retorna todos os códigos mapeados com a chave sendo o nome da
 * cidade.
 **/
function getCodeMap(){
  var codeMap = {};
  var data = openAndReadSheet(LOCATION_SHEET);
  for(var i = 1; i < data.length; i++){
    var city = data[i][CITY_NAME_COLUMN];
    var adsCode = data[i][ADS_CODE_COLUMN];
    var code = {
      'adsCode': adsCode
    }

    if(!codeMap[city]){
      codeMap[city] = code;
    }
  }
  return codeMap;
}

/**
 * Checa quais cidades devem receber os ajustes modificados.
 * Parâmetros: campaigns -> lista de nomes de campanhas;
 *             cities -> lista de nomes de cidades;
 *             convertionCondition -> informações da condição de
 *           ajustes modificados;
 *       cityCodes -> lista de códigos do Google Ads.
 * Retorna uma lista de pares cidade/local para receberem ajustes mo-
 * dificados
 **/
function runReport(campaigns, cities, convertionCondition, cityCodes){
  var setToBid = {};
  var adsCodeList = getAdsCode(cities);
  var campaignsIds = getCampaignID(campaigns);

  if(adsCodeList && campaignsIds){
// Monta o relatório desejado
    var report = AdsApp.report(
      'SELECT CampaignId, Conversions, ConversionValue, '+
```

```
          'Cost, CityCriteriaId ' +
      'FROM   GEO_PERFORMANCE_REPORT ' +
      'WHERE ' +
      'CampaignId IN ' + '[' + campaignsIds + '] ' +
      'AND CityCriteriaId IN ' + '[' + adsCodeList + '] ' +
      'DURING LAST_7_DAYS'; API_VERSION);

    var rows = report.rows();

    while (rows.hasNext()) {
      var row = rows.next();
  // Lê informações de uma linha do relatório
      var campaignId = row['CampaignId'];
      var conversions = row['Conversions'];
      var conversionValue = row['ConversionValue'].replace(/,/g , '');
      var cost = row['Cost'];
      var cityCriteriaId = row['CityCriteriaId'];
  var roas = 0;
  if(parseFloat(cost) == 0){

if(parseFloat(conversionValue) > 0){
  roas = 99999;
}
  }
  else {
roas = parseFloat(conversionValue) / parseFloat(cost);
  }

      if(checkModifiedCondition(convertionCondition,
                           conversions, roas)){
        if(!setToBid[campaignId]){
          setToBid[campaignId] = [];
        }

        if(setToBid[campaignId]){
  setToBid[campaignId]
  .push([parseInt(campaignId),
        cityCodes[cityCriteriaId]['adsCode']]);
        }
```

```
      }
    }
  }
  return setToBid;
}

/**
 * Checa se uma determinada condição de ajuste modificado está
 * sendo cumprida.
 *
 * Parâmetros: convertionCondition -> informações da condição;
 *             conversions -> quantia de conversões;
 *             roas -> valor calculado do ROAS.
 *
 * Retorna verdadeiro ou falso.
 **/
function checkModifiedCondition(convertionCondition,
                                conversions, roas){
  if(convertionCondition['type'] == 'Conversão'){
    return checkValue(convertionCondition, conversions);
  }
  else{
    return checkValue(convertionCondition, roas);
  }
}

/**
 * Checa se um valor está dentro dos limites de uma condição.
 *
 * Parâmetros: convertionCondition -> informações da condição;
 *             value -> valor a ser comparado.
 **/
function checkValue(convertionCondition, value){
  if(convertionCondition['condition'] == 'Acima'){
    if(value >= convertionCondition['value1']){
      return true;
    }
  }
```

```
  else if (convertionCondition['condition'] == 'Abaixo'){
    if(value <= convertionCondition['value1']){
      return true;
    }
  }
  else if(convertionCondition['condition'] == 'Entre'){
    if(value >= convertionCondition['value1']
   && value <= convertionCondition['value2']){
      return true;
    }
  }
  return false;
}

/**
 * Registra o histórico de temperatura na planilha de controle.
 **/
function setHistory() {
  var spreadsheet = SpreadsheetApp.openById(SPREADSHEET_ID)
  var sheet = spreadsheet.getSheetByName(HISTORY_SHEET)
  var nextRow = sheet.getDataRange().getLastRow() + 1

  var vetor = []

  for (var key in WEATHER_CACHE) {
    vetor.push([key, WEATHER_CACHE[key]])
  }

  var range = sheet.getRange('A' + nextRow + ':A' +
                             (nextRow + vetor.length - 1))
  writeDate(range)
  sheet.getRange(nextRow, 2, vetor.length, 2).setValues(vetor)
}

/**
 * Escreve a data em uma seleção da planilha.
 **/
function writeDate(range) {
  var today = new Date()
```

```
  var day = today.getDate()
  var month = today.getMonth() + 1
  var year = today.getFullYear()
  var date = year + '-' + month + '-' + day
  range.setValue(date)
}
```