Arthur Segura Ortiz Novello
Luca Ezellner Miraglia
Lucas Marques de Araujo

Inteligência para Transporte Público

São Caetano do Sul
2020

Arthur Segura Ortiz Novello
Luca Ezellner Miraglia
Lucas Marques de Araujo

Inteligência para Transporte Público

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia Mauá do Centro Universitário do Instituto Mauá de Tecnologia como requisito parcial para a obtenção de título de Engenheiro de Computação.
Área de Concentração: Engenharia de Computação

Orientador Tiago Sanches da Silva
Coorientador Murilo Zanini de Carvalho

São Caetano do Sul
2020

Novello, Arthur Segura

Inteligência para Transporte Público / Arthur Segura Ortiz Novello , Luca Ezellner Miraglia , Lucas Marques de Araujo . - São Caetano do Sul: CEUN-IMT, 2020.

48 p.

Trabalho de Conclusão de Curso - Escola de Engenharia Mauá do Centro Universitário do Instituto Mauá de Tecnologia, São Caetano do Sul, SP, 2020.

Orientador: Prof. Me. Tiago Sanches da Silva

1. computação em nuvem. 2. painel de controle. 3. inteligência artificial. 4. análise de dados. I. Miraglia, Luca Ezellner. II. Araujo, Lucas Marques. III. Instituto Mauá de Tecnologia. Escola de Engenharia Mauá. IV. Título.

Arthur Segura Ortiz Novello
Luca Ezellner Miraglia
Lucas Marques de Araujo

Inteligência para Transporte Público

Trabalho de Conclusão de Curso aprovado como requisito parcial para a obtenção de título de Engenheiro de Computação pela Escola de Engenharia Mauá do Centro Universitário do Instituto Mauá de Tecnologia.
Área de Concentração: Engenharia de Computação

Banca examinadora:

Prof. Me. Tiago Sanches da Silva
Orientador

Prof. Murilo Zanini de Carvalho
Coorientador
Escola de Engenharia Mauá

Prof. Me. Murilo Zanini de Carvalho
Avaliador

Prof. Dr. Sergio Ribeiro Augusto
Avaliador

São Caetano do Sul, 01 de Dezembro de 2020.

Aos nossos pais, irmãos e irmãs, amigos e colegas de formação.

Agradecimentos

Ao nosso orientador, Prof. Me. Tiago Sanches, que nos auxiliou desde o início, que sempre nos motivou e nos ajudou a traçar um melhor caminho durante toda a elaboração do projeto. Seus conselhos, críticas, ideias e conhecimento foram fundamentais para a realização deste trabalho.

Ao Prof. Me. Murilo Zanini, nosso coorientador por estar sempre nos auxiliando quando preciso, aconselhando nos afazeres e entregas relacionadas a este trabalho.

Ao Prof. Dr. Sergio Ribeiro, pela sua paciência e presença contínua, nos assessorando, motivando e inspirando nos momentos difíceis. Seu amplo conhecimento teórico e prático também contrinuiu na elaboração deste documento, além de sua rica experiência e vivência acadêmica.

Ao Instituto Mauá de Tecnologia, por nos fornecer infraestrutura e suporte durante toda a formação acadêmica dos integrantes deste grupo ao longo de todo o período de graduação.

Aos amigos, familiares e todas as outras pessoas que também contribuíram direta ou indiretamente para o desenvolvimento deste trabalho.

“Technology will reinvent the business, but human relationships will remain the key to success”
("A tecnologia vai reinventar o negócio, mas as relações humanas continuarão a ser a chave para o sucesso.")
Stephen Covey

Resumo

O projeto tem como objetivo a criação de um painel de visualização (*dashboard*) que monitora as demandas das frotas de veículos, no caso as linhas de ônibus da SPTrans, empresa responsável pelos ônibus da cidade de São Paulo. Para esse monitoramento, foram utilizados dados públicos fornecidos pela SPTrans, como rotas dos ônibus, dados de GPS, linhas e paradas, por exemplo, que foram enriquecidos com outras informações, como datas e locais de eventos, situação das linhas de trens e metrôs, informações meteorológicas e o trânsito na cidade. Considerando que atualmente muitos dados do transporte coletivo são públicos, mas não existe uma plataforma centralizadora dessas informações que seja capaz de auxiliar as pessoas no que diz respeito a uma visão geral das condições das linhas de ônibus, o presente trabalho pretende contribuir positivamente para melhorar a experiência dos usuários e trazer uma visão mais transparente do serviço como um todo. Para a construção do sistema foram utilizadas ferramentas como Python, Django, Django-REST, Celery e Amazon Web Services (AWS), que possibilitaram o desenvolvimento do *dashboard* e todas suas funcionalidades, desde a aquisição de dados externos, até a manipulação desses dados e implantação da plataforma na nuvem.

Palavras-chaves: computação em nuvem. painel de controle. inteligência artificial. análise de dados. decisão. transporte público.

Abstract

This project aims to create a dashboard that monitors the demand of vehicles in bus lines of SPTrans, the company responsible for buses in the city of São Paulo. For this monitoring, public data provided by SPTrans was used, such as bus routes, GPS data, lines and stops, for example, which was enriched with other information, like dates and locations of events, status of train and subway lines, weather information and traffic around the city. Although a lot of public transport data is public available, there isn't a system to aggregate all this information that would be able to help people who use this kind of transport on a daily basis with a overview about the lines condition. The present work intends to contribute positively to improve the user experience and bring a more transparent and complete view of the service. During the development, tools such as Python, Django, Django-REST, Celery and Amazon Web Services (AWS) were used, which made it possible to create a dashboard and all its features, from the acquisition of external data, to the manipulation of that data and deployment of the platform in a cloud platform.

Key-words: cloud computing. dashboard. artificial intelligence. data analysis. decision. public transportation.

Lista de ilustrações

Lista de abreviaturas e siglas

API	*Application Programming Interface*
IPEA	Instituto de Pesquisa Econômica Aplicada
CPTM	Companhia Paulista de Trens Metropolitanos
IBGE	Instituto Brasileiro de Geografia e Estatística
IA	Inteligência Artificial
SPTrans	São Paulo Transportes
AWS	*Amazon Web Services*
EC2	*Elastic Compute Cloud*
SQS	*Simple Queue Service*
CRUD	*Create, Read, Update, Delete*
REST	*Representational State Transfer*
IoT	*Internet of Things*
GPS	*Global Positional System*
CCTV	*Closed Circuit Television*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
OpenCV	*Open Source Computer Vision*
GTFS	*General Transit Feed Specification*
UNINDU	*The International Congress on University Industry*
BI	*Business Inteligence*
PIB	Produto Interno Bruto
CAGR	Taxa Anual Composta de Crescimento
UFSC	Universidade Federal de Santa Catarina
Relu	*Rectified Linear Unity*
Elu	*Exponential Linear Unity*
BMTC	*Bengaluru Metropolitan Transport Corporation*

IISC	Instituto Indiano de Ciência
noSQL	*Not Only SQL*
SQL	*Structed Query Language*
ADMA	*Association For Data Driven Marketing Advertising*
YOLO	*You Only Look Once*
KMZ	*Keyhole Markup Language Zip*
XML	*Extensible Markup Language*
HTML	*Hypertext Markup Language*
CSS	*Cascading Style Sheets*
REST	*Representational State Transfer*
HTTP	*Hypertext Transfer Protocol*
JSON	*Javascript Object Notation*
RDS	*Relational Database Service*
CPU	*Central Process Unit*
IaaS	*Infrastructure as a Service*
SaaS	*Software as a Service*
PaaS	*Platform as a Service*
FaaS	*Function as a Service*
RAM	*Random Access Memory*
IBM	*International Business Machines*
ARM	*Advanced RISC Machine*
RISC	*Reduced Instruction Set Computer*
GPU	*Graphic Processor Unit*
SDK	*Software Development Kit*
GCP	*Google Cloud Platform*
TI	Tecnologia da Informação

PyPi	*Python Package Index*
MTV	*Model Template View*
MVC	*Model View Controller*

Sumário

1 Introdução

1.1 Transporte público

O transporte público caracteriza-se como uma opção amplamente utilizada por pessoas a fim de garantirem suas necessidades de locomoção. Por possuir um preço mais acessível e muitas vezes ser mais rápido e prático, 65% da população das capitais do Brasil utiliza essa forma de transporte, como aponta um estudo realizado pelo Instituto de Pesquisa Econômica Aplicada (Ipea) (PEDUZZI, 2011).

Foi direcionado o valor de R$ 707 milhões no ano de 2019 pela União para a área de mobilidade urbana e trânsito, como indica a Lei Orçamentária Anual (LOA n° 13.808/2019). Desse montante, especificamente para o transporte público coletivo, foram separados apenas R$ 348 milhões, uma fatia de 0,01% do orçamento total da União (NTURBANO, 2019).

De acordo com uma estimativa do BNDES, em 2015, seria necessário investir mais de R$ 234 bilhões em transporte público para resolver os problemas da área nas principais regiões metropolitanas do país, portanto, caso mantido o nível de investimento atual, levaria mais de 600 anos para atingir o montante proposto pelo BNDES (SANTOS, 2015). Tendo em vista o baixo investimento frente a demanda, é esperado que o transporte público gere um elevado nível de insatisfação em seus usuários, o que o IPEA demonstrou em outra pesquisa realizada em 2011 e 2012, na qual o transporte público foi avaliado por mais de 60% do público como "péssimo ou ruim"(SANTOS, 2015).

Para entender a situação em que se encontra o transporte, o primeiro passo é olhar como o país se urbanizou. No último século, o Brasil passou por um intenso processo de industrialização, o que gerou um forte êxodo rural e principalmente uma migração da população do Nordeste para o Sul e Sudeste, onde se concentrou a produção industrial do país.

Esse crescimento populacional nas metrópoles foi acompanhado por uma grande valorização dos terrenos e moradias nas áreas centrais das cidades. Com isso é observado um efeito de gentrificação, afastando a classe trabalhadora para regiões mais periféricas, longe de onde se concentra a maior parte da oferta de emprego, o que gerou uma demanda por transporte, crescente até os dias atuais. Tendo em vista esse cenário, a população começou a consumir cada vez mais carros populares, que contam com incentivo do governo para serem produzidos. Com isso, o que se vê nos ambientes urbanos são ruas congestionadas e ônibus lotados (PENA, s.d.).

1.2 Justificativa

O presente trabalho tem por motivação a grande quantidade de cidadãos de São Paulo

que apresentam adversidades quando se trata da utilização dos ônibus da cidade, como o elevado tempo de espera e as altas taxas de ocupação interna, que lideram as reclamações do Reclame Aqui (DEMORA, 2020) (DEMORA, 2018). Além disso, a criação de um processo inteligente e automatizado que auxilie os gestores da SPTrans (empresa responsável pelo gerenciamento dos transportes do município de São Paulo) no ajuste das frotas de ônibus justificam esse projeto.

Ainda olhando para o transporte público de São Paulo, hoje o sistema todo, composto de ônibus, trens, metrô e outros modais, transporta mais de 17 milhões de passageiros diariamente na capital (G1, 2013), sendo que dessa quantidade, 7 milhões utilizam as linhas de ônibus da cidade, que hoje conta com mais de 1.300 linhas (SAMPA, 2018), sendo as principais a Terminal Bandeira / Terminal Varginha, Terminal Jardim ngela / Metrô Santa Cruz e Hospital Itaim / Guaianazes , que juntas transportam cerca de 100 mil passageiros por dia (LOBO, 2013).

Hoje, a SPTrans disponibiliza uma API, atualizada em tempo real, com informações de localização dos veículos, velocidades, pontos de parada e quantidade de ônibus ativos. Apesar de provavelmente a empresa contar com uma solução proprietária, decidimos coletar essas informações e enriquece-las com outros dados disponíveis para desenvolver o nosso trabalho.

1.3 Objetivos

Tendo em vista o cenário apresentado, pretende-se criar um *dashboard* para centralizar informações e agilizar a tomada de decisão. Tendo como base dados coletados durante os trajetos de ônibus, informações extraídas de APIs públicas, a ferramenta irá apoiar os gestores da SPTrans no controle do fluxo de tráfego, além de ser um ótimo indicador para a população que deseja ter uma visão geral do transporte público de São Paulo.

Além disso, o *dashboard* será atualizado em tempo real, exibindo gráficos e indicadores de maneira sucinta, com a possibilidade de aplicação de filtros nas consultas. Considerando o volume de passageiros para um determinado dia e horário somados a dados externos, a ferramenta permitirá (em trabalhos futuros) utilizar algoritmos de inteligência artificial para identificar padrões históricos e prever futuras demandas nas frotas de ônibus, facilitando o controle do serviço e auxiliando no dia a dia do transporte urbano na cidade.

1.4 Contribuições do trabalho

Este trabalho tem como contribuição a melhoria na tomada de decisões da administradora de ônibus (SPTrans) onde será possível, de forma centralizada, a visualização de dados e gráficos por meio de um *dashboard*. Além disso, pode-se citar como contribuição a diminuição do tempo de espera e ocupação dos transportes, resultando em uma melhor qualidade de vida para a população de São Paulo.

1.5 Questão central da pesquisa e seus impactos

Os benefícios esperados com esse trabalho de conclusão de curso são proporcionar uma melhor qualidade locomotiva dos ônibus de São Paulo e proporcionar uma melhor gestão aos envolvidos da SPTrans, otimizando e inovando a forma de ver os dados. O transporte público, mais especificamente, os ônibus, podem causar estresse. A sensação pode estar relacionada a fatores como as condições do transporte, o desconforto de estar entre muitas pessoas e o elevado tempo de espera.

Com a visualização dos dados disponíveis os gestores e responsáveis poderão ver a situação de lotação e do tempo de espera através do *dashboard*. Numa situação crítica, eles poderão solicitar um aumento da frota e reduzir o desconforto de andar em um veículo com excesso de pessoas além de reduzir o tempo de espera. Essas ações impactam diretamente na saúde das pessoas reduzindo o estresse gerado pelos transportes.

Nos próximos tópicos deste texto, serão apresentados o ferramental do trabalho realizado, desde os conceitos básicos, como Análise de Dados e o que é e como pode ser utilizado um *Dashboard*, passando por exemplos de uso desses conceitos em outros trabalhos, no Brasil e no exterior, e chegando até nos *softwares* e serviços que foram usados no desenvolvimento do sistema. Além disso, no capítulo de Metodologia, será mostrado onde cada um desses conceitos e ferramentas foram aplicados.

2 Revisão bibliográfica

2.1 Tecnologia de Monitoramento de Ônibus

De acordo com a informação exposta em uma publicação feita em 2017 pela Escola de Negócios da Universidade de Indiana, era previsto um crescimento anual de mais de 23% no mercado de *Big Data* durante o período de 2014 a 2019, com um custo de $48,6 bilhões no último ano. Isso inclui um crescimento de 30% entre 2014 e 2015 de aparelhos conectados e dispositivos de IoT. Estes aparelhos geram uma quantidade enorme de dados valiosos para quem tiver interesse de processá-los (LEE, 2017). Esse cenário hoje não é diferente para o setor de transporte público, com a prefeitura de São Paulo capturando dados em tempo real da sua frota de ônibus, informações como bilhetagem, velocidade e paradas dos veículos, porém não utilizando a informação para a tomada de decisão.

Um exemplo de uso prático desses dados foi apresentado na Conferência Internacional da Logística e Transporte Avançado. Em uma colaboração entre a IBM e o Conselho da Cidade de Dublin foi realizado um projeto de cidade inteligente entre 2010 e 2013 (Ben Ayed; Ben Halima; ALIMI, 2015). A IBM passou a processar os dados gerados pela frota de ônibus, além de outras fontes, com intenção de reduzir o trânsito na cidade sem precisar alterar a sua estrutura atual, que conta com vários pontos históricos. O inicio do processo se dava com informações coletadas do ônibus, como dados de GPS, velocidade, paradas e bilhetagem, e depois eram adicionadas novas informações vindas de sistemas de semáforos, *CCTV*, sistemas meteorológicos, entre outros. Todos esses dados então eram processados em um servidor da IBM e disponibilizados em mapa em tempo real do transporte público de Dublin. Com toda essa informação processada, a cidade teve uma maior capacidade de monitorar o seu sistema de transporte público, diminuindo o tempo para uma tomada de decisão.

Outro exemplo de aplicação de tecnologia no monitoramento de transporte coletivo é do USapiens, um sistema desenvolvido por um time de pesquisa da IBM do Brasil (VIEIRA et al., 2015). Essa equipe usou dados de transporte coletivo da cidade do Rio de Janeiro para desenvolver um sistema que processa os dados recebidos pelo GPS dos ônibus e depois disso analisa-se por diversos modelos esses dados. Para isso eles integraram os dados obtidos pelo GPS com informações disponíveis de GTFS (GOOGLE, 2020), sigla para *General Transit Feed Specification*, que contém dados mais gerais das rotas de ônibus, como paradas e horários esperados. Feita essa integração, os dados são limpos para prevenir problemas como latitude/longitude imprecisas ou dados com intervalo de tempo muito grande. Com os dados prontos, é feita uma comparação com a rota do GTFS, se descobre a direção do veículo e com isso tem seu trajeto normalizado em uma escala de distância e tempo acumulativas.

Com a informação normalizada, o sistema pode ajudar a responder três perguntas

principais. Através de uma análise descritiva histórica podemos responder "O que aconteceu e porque?", analisando os dados em tempo real se responde "O que está acontecendo e porque?"e uma análise preditiva responde "O que vai acontecer e porquê?". Os pesquisadores da IBM depois aplicaram esse sistema a 5 casos de estudo. O primeiro foi uma Análise de Uniformidade dos Ônibus, para evitar um agrupamento de veículos na linha, o segundo caso foi uma verificação na rotas dos ônibus, para avaliar a aderência do veículo a sua rota pré-definida, o terceiro uma análise de fluxo no trânsito, o quarto a variância do tempo de viagem do veículo, que permite avaliar a consistência da rota analisada, e para o quinto ele usaram a análise preditiva para prever o tempo de chegada do ônibus.

2.2 *Dashboard*

O *Dashboard*, também chamado de painel de controle, é uma ferramenta que auxilia os gestores a terem uma visão mais sistemática das principais informações do negócio. Em outras palavras, é um recurso que visa consolidar os dados de maior relevância em um painel, facilitando o processo de análise e a tomada de decisão (INTELIPOST, 2017).

O uso de planilhas e relatórios já são ultrapassados para análises de dados, não sendo suficientes para suprir as necessidades mais urgentes. Conforme a tecnologia foi evoluindo no mundo corporativo, surgiram os *dashboards* que evitam esforços desnecessários e ter uma visão mais ampla de todo o cenário corporativo para, assim, tomar decisões estratégicas e assertivas.

A visualização de dados através de *dashboards* já é uma realidade em softwares de gestão empresarial, integrando painéis de controle com inteligência artificial e fornecendo informações atualizadas automaticamente. É possível personalizá-los e comparar dados através de filtros, que facilitam análises de indicadores (TECNICON, 2019).

2.2.1 Uma ferramenta de apoio à decisão

Segundo um estudo feito pela UNINDU (The International Congress on University Industry), 83% das pessoas absorvem melhor as informações através da visão. Isso demonstra a importância do *dashboard* para tomada de decisões rápidas e melhor análise dos indicadores,melhorando metas e atingindo objetivos.

O *Business Intelligence* (BI) (TABLEAU, 2020), por exemplo, é uma área que exige precisão na coleta e controle das informações para gerar insights em base desta ferramenta. Os dados são agrupados em conjuntos de registros e disponibilizadas por meio de *dashboards* para mensurar o desempenho atual e futuro da empresa de acordo com o cenário.

Além disso, os *dashboards* monitoram os dados, com o intuito de melhorar todos os processos. Eles permitem que o usuário personalize painéis e filtre informações para a

visualização dos resultados, como quantidade, tempo e outras opções (TECNICON, 2019).

A aplicação dos *dashboards* na gestão empresarial traz muitos benefícios para a tomada de decisão e a visão estratégica do seu negócio, que fica cada vez mais fácil através da centralização, visualização e compreensão das informações, o que possibilita uma visão ampla do seu negócio. Na gestão, é importante que todas as equipes tenham acesso aos indicadores da empresa, mantendo a transparência das informações. As ferramentas de *dashboard* tem o objetivo de facilitar a comunicação interna entre todos os profissionais, otimizando o tempo para tomarem decisões e evitarem trabalhos manuais e complexos com a organização de dados, passando a priorizar outras atividades mais relevantes. Com as informações consolidadas em um único painel, a gestão se torna mais ágil e efetiva, possibilitando o alinhamento de estratégias e decisões para o negócio (TECNICON, 2019).

Como o *dashboard* trabalha com atualizações constantes e análises mais específicas, é mais fácil prever problemas e tendências negativas que podem vir a acontecer. Estes problemas ficam mais explícitos com o uso da tecnologia de inteligência artificial que os identifica de maneira mais fácil.

Qualquer mudança é detectada com mais simplicidade e o tempo para pensar nas possíveis soluções se torna muito maior. Isso melhora o processo de tomada de decisão e evita possíveis prejuízos (SYSTEMSAT, s.d.).

Uma boa interface e uma boa experiência de uso se dá pela arquitetura das informação do *dashboard*. É fundamental que seja organizado, coerente e intuitivo. O objetivo é tornar o mais fácil possível encontrar o que se procura. Através de menus, cores e símbolos é possível saber quais são as opções e deixar claro as consequências que cada ação irá gerar. Dessa forma, a experiência do usuário ao usar o *dashboard* será rápida e efetiva, atendendo suas expectativas (KLEVERTON, 2019).

2.3 Desenvolvimento *web* com Django

Django é um *framework web* de alto nível, escrito em Python que encoraja o desenvolvimento rápido, limpo e pragmático de aplicações *web* (DJANGO, 2020c). Com ele é possível que o desenvolvedor se preocupe mais com as regras de negócio do que com a programação em si, uma vez que ele já implementa tratamento de requisições, mapeamento objeto-relacional, preparação de respostas HTTP, sistema de autenticação, entre outros. Além disso, o Django foi desenvolvido com um foco extra em segurança, implementando camadas que evitam os ataques mais comuns conhecidos.

A arquitetura do Django se baseia no padrão MTV (*Model-Template-View*) (PINHEIRO, 2020), que por sua vez é bem parecido com o padrão MVC (*Model-View-Controller*) (HIGOR,

2013), mas possui suas peculiaridades. Primeiramente, a camada conhecida como *Model* é a responsável pelo mapeamento do banco de dados para o projeto. A camada *Template* é a responsável pelo *layout* da página, ou seja, como os dados serão exibidos para o usuário. Por fim, na *View* é onde o processamento dos dados irá ocorrer antes de serem exibidos e onde ficam localizadas as regras de negócio.

Devido a essa arquitetura utilizada por padrão no Django, cada projeto é subdividido em diversos aplicativos, que são criados conforme a necessidade do desenvolvedor, os quais implementam individualmente o modelo MTV. Isso permite que esses aplicativos se tornem partes independentes do sistema, aumentando a sua portabilidade para outros projetos e consequentemente, o reaproveitamento de código.

Assim sendo, um aplicativo Django criado por algum desenvolvedor pode ser facilmente transformado em um pacote Python e disponibilizadao no repositório de *softwares* oficial do Python, o PyPI (PYPI, 2020). Isso permite que a comunidade reutilize um mesmo código que já foi desenvolvido para uma finalidade específica. Dois importantes pacotes utilizados neste projeto são o Django REST *Framework* (DJANGO, 2020b), que permite a fácil criação de APIs integradas com os modelos do Django e o Django *Plotly Dash* (DJANGO, 2020a), que implementa múltiplos estilos de gráficos e visualizações de dados escrevendo apenas código Python, que por sua vez podem ser facilmente incluídos em um template do Django.

2.4 Agendamento e execução de tarefas com Celery

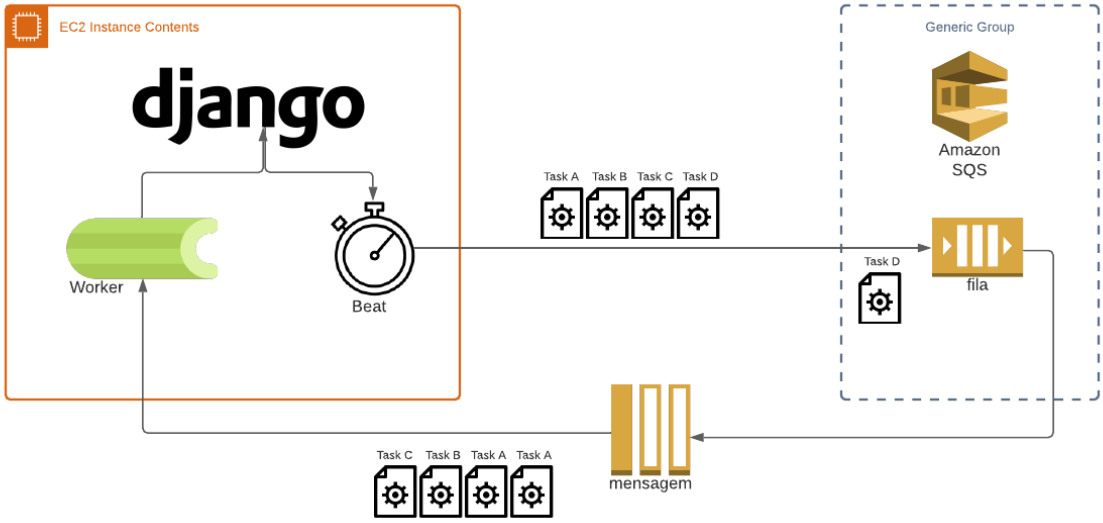
O Celery é um sistema em Python simples, flexível e confiável para processar grandes quantidades de mensagens ou tarefas (CELERY, 2018). Além disso é fácil de integrar com sistemas *web*. Ele permite a delegação de tarefas a diversos *workers*, localizados em outros servidores, ou seja, as tarefas são processadas de forma apartada da aplicação. De forma automática é criada uma fila de tarefas com foco no processamento em tempo real, ao mesmo tempo que oferece suporte ao agendamento de tarefas. É importante salientar que o Celery pode ser divididos em alguns componentes tais quais aplicação, *broker*, *worker* e *beat*.

A aplicação se refere ao próprio projeto ao qual o Celery vai ser implementado. Nela é necessário que existam funções ao qual devem ser executadas de forma apartada. Isso auxilia no desempenho principal do projeto.

O *broker* se refere ao sistema de gerenciamento de filas responsável por gerenciar todo o sistema de envio de tarefas, seja de forma assíncrona ou não. Existem várias opções para escolha de *broker* como RabbitMQ, Redis, Amazon SQS, Zookeeper. Dependendo do sistema operacional ou do local onde o projeto funcionará verifica-se a melhor opção. Neste projeto, como foi utilizado o sistema de nuvem da Amazon (AWS), foi escolhido o SQS, pois trata-se de um serviço oferecido pelo própria AWS.

O *worker* é quem executa de fato as tarefas agendadas e enviadas pela aplicação no *broker*. É nele que todo o processamento de execução fica. Existem algumas opções de execução que podem ser configuradas dependendo da necessidade do projeto. Dentre elas estão *prefork*, *solo*, *eventlet* e *gevent*. Basicamente o *prefork* e o *solo* trabalham de forma síncrona e utiliza consideravelmente a CPU por travá-la enquanto a solicitação não termina até começar a próxima. Por outro lado, o *eventlet* e o *gevent* funcionam de forma assíncrona e são mais indicados para requisições HTTP pois levam em conta o tempo de resposta do servidor e não consomem a CPU (STIEL, 2018).

Figura 1 – Diagrama de funcionamento do Celery



Fonte: Arquivo dos autores (2020)

O *beat* é um componente opcional que que dispara de tempos em tempos as tarefas para o *broker*, como se fosse a aplicação. No projeto decidimos utilizá-lo pois é preciso que de tempos em tempos, de forma programada, uma tarefa seja executada de forma assíncrona (SILVA, 2016).

2.5 Computação em Nuvem

Eric Knorr, editor chefe da InfoWorld, dá dois significados para a expressão Computação em Nuvem (KNORR, 2018), o primeiro, e mais comum, refere-se a serviços que executam processos remotamente, em servidores de empresas que provém esse tipo de serviço, como a Amazon Web Services, Microsoft Azure e Google Cloud Platform. O segundo significado

é a descrição de como esses serviços funcionam, um conjunto de recursos computacionais disponíveis sob demanda.

Essas infraestruturas permitem seus clientes usarem os recursos de maneira escalável e instantânea, sem precisar se preocupar com a infraestrutura física ou com a necessidade de *softwares* novos. As empresas provedoras garantem a manutenção das máquinas em si e permitem que os usuários foquem no serviço que estão prestando ou desenvolvendo.

Entre os serviços que esse modelo de computação fornece estão o SaaS, *software* como serviço ou *software* as a *service*, que são as aplicações que acessamos pelo navegador, como o Google Suit ou o Office 365, o IaaS, infraestrutura como serviço ou *infrastructure as a service*, que são soluções completas de computação na nuvem, permitindo desde de máquinas virtuais simples até servidores que precisam atender grandes demandas de processamento a todo momento, o PaaS, plataforma como serviço ou *platform* as a service, que são serviços com plataformas já minimamente configuradas, voltadas a desenvolvedores que precisam de agilidade para disponibilizar uma aplicação e que não querem ter o trabalho de configurar um IaaS mais completa, e por último o FaaS, função como serviço ou *function as a service*, que é uma camada de abstração do PaaS, onde o desenvolvedor precisa de apenas alguns trechos de código para se comunicar com funções específicas do seu provedor de nuvem, como AWS Lambda ou o IBM OpenWhisk (KNORR, 2018).

2.5.1 Amazon Web Services

Em 2006 a gigante do varejo eletrônico Amazon decidiu disponibilizar e rentabilizar a sua infraestrutura de TI, tecnologia da informação, para outros usuários através de um serviço, o Amazon Web Services, ou simplesmente AWS (AWS, 2020a). Hoje esse serviço gera uma renda anual de mais de 35 bilhões de dólares (COPPOLA, 2020) e atende mais de 1 milhão de usuários, incluindo empresas como Johnson & Johnson, McDonaldś e Samsung (SAUNDERS, 2020).

Hoje, a AWS, oferece mais de 175 serviços diferentes, partindo de armazenamento de dados até ferramentas para IoT (FREAK, 2019). Seus principais serviços incluem o *Elastic Compute Cloud* (EC2), o *Simple Storage Service* (S3) e o AWS Lambda (BRANDON, 2020).

2.5.1.1 Elastic Compute Cloud

Jeff Barr, atual vice presidente da AWS, anunciou em agosto de 2006 o lançamento da versão de testes do *Elastic Compute Cloud* (BARR, 2006), que se tornaria o principal recurso dentro das opções oferecidas pela AWS. Inicialmente oferecendo um sistema com um processador Intel Xeon de 1,7 GHz, 1,75 GB de RAM, 160 GB de disco rígido e uma conexão de 260 Mb/s, tinha o custo de 10 centavos de dólar por hora, podendo executar distribuições de

Linux, e já contava com um alto nível de escalabilidade, com o usuário tendo a possibilidade de executar quantos sistemas, chamados de "*virtual CPU*", ele achasse necessário.

Hoje o EC2 suporta várias distribuições de Linux, incluindo Red Hat Enterprise, Ubuntu e a distribuição própria da Amazon, o Amazon Linux, Windows Server e Raspbian para seus servidores com plataforma ARM (AWS, s.d.e). Além disso, também é possível escolher várias configurações de hardware em várias faixas de preço partindo de versões básicas, com apenas um núcleo e 1 GB de RAM, sem custo, até máquinas com múltiplos processadores físicos e múltiplas GPUs, que cobram mais de 10 dólares a hora, ficando a cargo do usuário escolher a opção que o melhor atende (AWS, s.d.d).

2.5.1.2 Simple Queue Service

O *Simple Queue Service* (SQS) é um dos serviços web mais antigos fornecido pela Amazon, sendo anterior até mesmo a AWS. Lançado em 2004 (BARR, 2014), ele fornece uma API genérica para ser usada em gerenciamentos de fila em linguagens suportadas pelo AWS SDK, por meio de mensagens armazenadas e enviadas pelo servidor.

As vantagens de usar um serviço como o SQS no lugar de soluções locais, como Redis ou o RabbitMQ, é a disponibilidade e a escalabilidade que um serviço como a AWS fornece. Por meio de redundância de servidores a AWS garante que o SQS estará sempre disponível para processar e consumir as mensagens e poderá escalar o seu serviço caso receba uma carga muito grande dessas chamadas (AWS, s.d.c).

2.5.1.3 Relational Database Service

O AWS *Relational Database Service*, ou simplesmente RDS, é um serviço de banco de dados pré configurado, em que o usuário interage através de uma API em não tem o trabalho de configurar uma instância apenas para o seu banco de dados e também não precisa se preocupar com o uso de recursos da infraestrutura onde a sua aplicação principal está sendo executada, seja em uma instância EC2, um serviço como o AWS Lambda ou um servidor local (AWS, s.d.a).

Lançado em 2009 com suporte apenas a MySQL, hoje o serviço pode executar vários tipos de banco de dados relacionais, como MariaDB, PostgreSQL e Oracle (AWS, s.d.b). Por funcionar como uma abstração de uma instância EC2, o RDS oferece as mesmas vantagens, incluindo um alto nível de escalabilidade, o que permite que seja usado por grandes empresas como Netflix, Expedia e Unilever.

2.5.2 Google Cloud Platform

A Google tem uma plataforma que concorre com a AWS, a Google Cloud Platform

(GCP), que a apesar de oferecer serviços como o Google *Cloud Engine* e o Google *Cloud Storage*, equivalentes ao AWS EC2 e AWS S3, tem um foco maior em containers e serviços próprios da Google, como o Google Maps e o Google Suit (III, 2019).

Através da GCP é possível fazer consultas por API ao Google Maps, o que permite consultas de endereços partindo do nome de locais ou mesmo o uso de mapas completos adaptados a necessidade da aplicação (GOOGLE, s.d.).

2.6 Análise de Dados

Apesar de estar cada vez mais em destaque nestes últimos anos, podemos traçar o uso de estatística para tomada de decisão desde do Egito Antigo, de acordo com um artigo publicado por Keith D. Foote (FOOTE, 2018) os egípcios já usavam cálculos estatísticos para construir as pirâmides. Já na década de 1880, o governo americano levou pelo menos 7 anos para completar o censo da população, processo que foi reduzido para um ano e meio na década seguinte por causa do desenvolvimento de uma máquina de tabulação, por Herman Hollerith, que processava os dados de forma sistêmica em cartões perfurados.

2.6.1 Fases da análise de dados

Trazendo para os dias atuais, a Universidade de Villanova separa a análise de dados em três estágios. O primeiro deles seria o início do que chamamos de *Business Intelligence*, que surgiu por volta de 1950 como uma forma de processar pequenas quantidades de informações estruturadas. Esse estágio, que podemos chamar de *Analytics 1.0*, durou até cerca de 2009, quando foi consolidado o termo *"big-data"*.

O fim do primeiro estágio se deu pelo aumento exponencial de dados sendo produzidos diariamente, podendo vir de qualquer lugar e forma, desde de informações simples e estruturadas, como quais produtos alguém comprou em um determinado site, ou coisas mais complexas, como quais sites fizeram essa pessoa chegar até uma determinada página e qual a posição geográfica dessa pessoa quando acessou a página. Se convencionou a chamar esses dados em grande quantidade e pouco estruturados de *"big-data"*.

Pela dificuldade de armazenar e processar essa massiva quantidade de dados, se fez necessário o desenvolvimento de novos métodos e ferramentas para o processamento deles, o que deu início ao *Analytics 2.0*, que trouxe alternativas como Hadoop, que pode processar essas grandes massas de dados, e o NoSQL, que permite armazenar toda a informação de forma mais eficiente que os bancos de dados relacionais faziam até então. Além disso, cada vez mais se fez necessário um conhecimento tecnológico para fazer essas análises, quando até então bastava o conhecimento de métodos estatísticos.

Hoje em dia, muitos especialistas dizem que chegamos a um terceiro estágio, o *Analytics 3.0*, onde esses dados que são produzidos a qualquer momento podem ser usados como moeda de troca entre consumidor e fornecedor. Essa moeda pode ser analisada imediatamente pelo fornecedor, que passa a entregar uma experiência personalizada para quem está consumindo o seu produto (VILLANOVA, s.d.).

2.6.2 Análise de dados no dia a dia

Com a disponibilidade de dados que temos, as empresas estão se dedicando cada vez mais em coletar e auxiliar as suas decisões nas análises feitas com eles. O Diretor de Estratégia e Marketing de Precisão da Coca-Cola, Justin De Graaf, afirmou em entrevista para ADMA, Association for Data-Driven Marketing & Advertising, que cada vez mais a empresa usa informações coletadas diretamente dos consumidores, como por meio de telefone, redes sociais ou *e-mail*, para criar desde campanhas publicitárias até novos produtos (TAN, 2017). Outra marca consolidada usando análise de dados para a tomada de decisão é a rede de hotéis Marriott, que, de acordo com o seu *chief development officer*, Eric Jacobs, tem usado esse tipo de informação para decidir como identificar, atrair e manter os seus cliente mais lucrativos (EISEN, 2018). Para isso eles coletam desde dados como padrão social dos clientes até se eles consomem mais jeans Levi's ou Gap.

Apesar disso, alguns resultados dessas análises podem ter um efeito contrário do que era pretendido no início. Caso a análise não seja feita com atenção, verificando sempre quem está sendo atingido, ela pode levar a alguns atritos e desgastar a imagem de quem usa essas informações.

Em 2012, o New York Time publicou uma história de como um pai descobriu a gravidez da filha através de uma promoção oferecida pela rede varejista Target, nos Estados Unidos (DUHIGG, 2012). Andrew Pole, um estatístico da rede, foi designado para desenvolver um "preditor de gravidez", no fim do estudo ele conseguiu chegar a uma lista de 25 produtos, que geram uma probabilidade da cliente estar grávida de acordo com o seu preditor. Esses produtos contêm itens como manteiga de cacau, bolsas grandes o suficiente para caber pacotes de fraldas, e suplementos como magnésio e zinco.

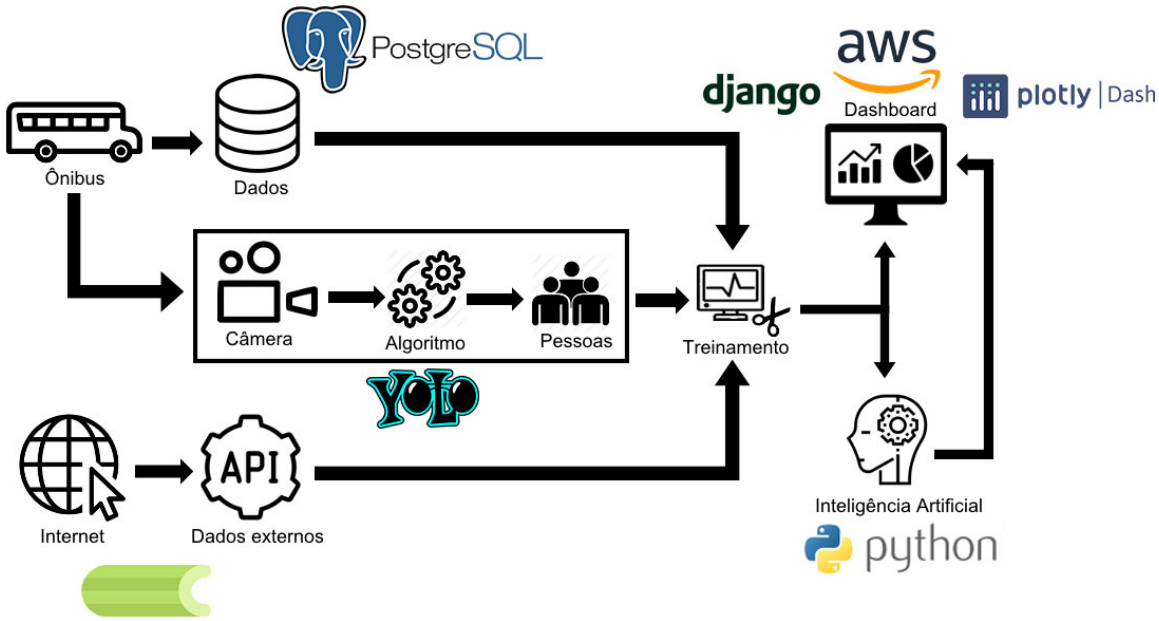
A Target vinculava essa informação a um identificador do cliente, e então ofereceria um desconto na próxima visita que essa mesma pessoa fizesse a loja. Um ano após a aplicação deste preditor, o pai de uma adolescente foi a uma das lojas reclamar que sua filha tinha recebido um desconto relacionado a esse programa específico para grávidas, com isso, o gerente da loja se desculpou pelo erro e alguns dias depois ligou para se desculpar novamente. Entretanto, para a surpresa do gerente, durante essa ligação o cliente disse que a filha não tinha contado para ele que estava grávida, e que a Target soube antes dele do acontecido. Após esse caso, o departamento de *marketing* decidiu desacelerar o programa de análises de dados, e passar mais tempo avaliando

qual impacto cada iniciativa pode ter.

3 Metodologia

A fim de solucionar o problema identificado, foram definidos métodos e estratégias com o objetivo de criar um fluxo de trabalho. A Figura 2 fornece um diagrama ilustrativo e simplificado da solução proposta.

Figura 2 – Ilustração da solução



Fonte: Arquivo dos autores (2020)

Primeiramente alguns dados dos ônibus como, por exemplo, velocidade, tempo de parada, linhas, posição dos veículos são coletados juntamente com imagens do interior dos veículos, que passam por um algoritmo de reconhecimento para identificar a lotação dos mesmos. Paralelamente, informações obtidas por meio de APIs externas enriquecem o conjunto de dados, que por sua vez passa por um tratamento e um treinamento. Após esse processo algumas informações são direcionadas diretamente para o painel de controle enquanto outras passam pela inteligência artificial para depois serem colocadas no *dashboard*.

3.1 Aquisição de dados externos por meio de tarefas

No que diz respeito à aquisição de dados para alimentação do *dashboard* e sucessiva análise, foi utilizada como principal fonte, a API pública da SPTrans, mais especificamente os *endpoints* (SCHULTZ, 2020) "Parada", "Posicao" e "KMZ". Cada *endpoint* é responsável por trazer informações como as paradas dos corredores dos ônibus de São Paulo, a posição dos veículos, assim como sua respectiva linha e frota em um dado instante e um arquivo (TECHTUDO, 2016) contendo informações de velocidade das linhas, respectivamente.

Vale ressaltar que para a aquisição dos dados do arquivo KMZ, foi realizada uma quebra do arquivo, que originalmente possui as informações em formato XML (MAGALHAES, 2020). Esse procedimento foi necessário para obter todas as informações já tratadas do arquivo em formato JSON (JSON, 2020) para carga do banco de dados.

Adicionalmente a API da SPTrans, são coletados dados de outras fontes, como Climatempo, Direto dos Trens e *Tickets for Fun*, que são responsáveis por dados como condições meteorológicas, situação das linhas de trens e metrôs em tempo real e localização e data de eventos, respectivamente.

Tendo em vista que todos os dados são coletados com informações em tempo real, é necessário que eles sejam armazenados para que se construa um histórico a fim de se analisar os dados futuramente. Com esse objetivo, foi utilizado o Celery (CELERY, 2018), um módulo Python que permite a criação de tarefas assíncronas e agendadas. Tais tarefas foram modeladas para coletar as informações das fontes mencionadas anteriormente em um determinado período de tempo de forma assíncrona e salvá-las em um banco de dados PostgreSQL (POSTGRESQL, 2020), criado exclusivamente para o projeto.

Paralelamente à utilização do Celery, foi utilizado um serviço da AWS (AWS, 2020b), o *Amazon Simple Queue Service* (SQS), que recebe as tarefas agendadas do Celery e as administra em uma fila, enviando novamente para execução quando for a hora certa. Dessa maneira, as informações em tempo real passaram a ser extraídas ciclicamente por meio das tarefas e salvas em um banco de dados dedicado, gerando um histórico para análise.

3.2 Criação de gráficos com Django

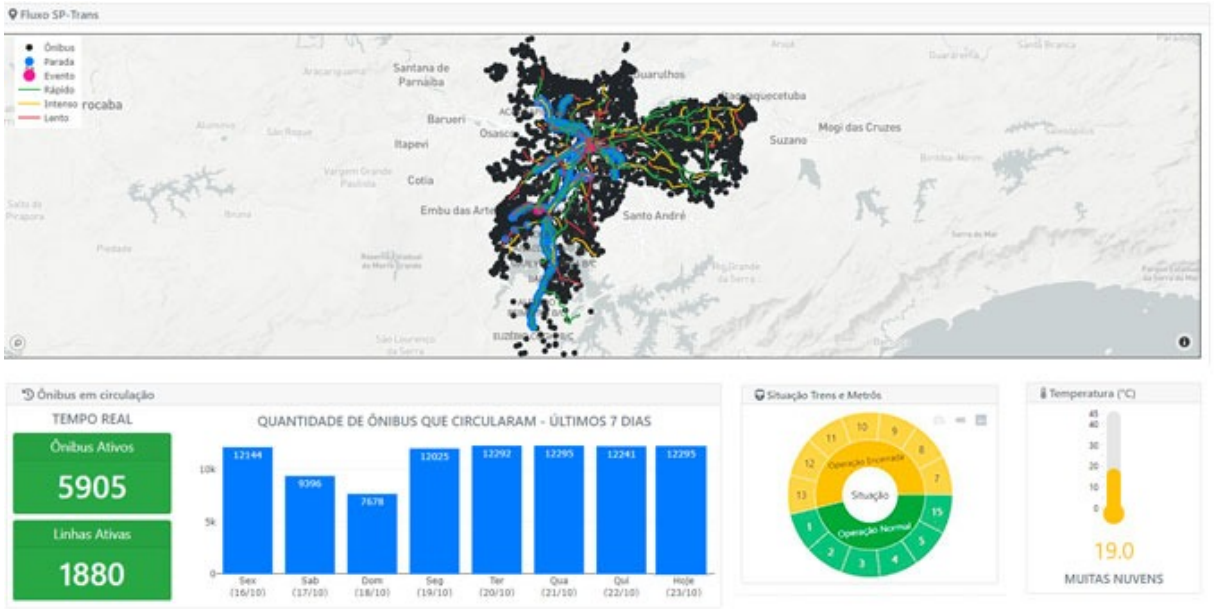
Para a criação do *dashboard* do projeto que exibe informações extraídas a partir dos dados disponíveis no banco de dados, foi utilizado o Django (DJANGO, 2020c), um *framework* Python de desenvolvimento rápido para *web*.

O Django utiliza o padrão *model-template-view*, que fornece todas as ferramentas necessárias para o desenvolvimento *web*, desde a criação de um modelo associado ao banco de dados,

até o processamento de requisições e criação de páginas *web* dinâmicas de forma robusta e simples. Um de seus principais diferenciais é a padronização dos aplicativos e portabilidade dos mesmos para outros projetos. Isso possibilita uma fácil implementação de módulos e aplicativos externos.

Assim sendo, um dos principais aplicativos utilizados nesse projeto foi o *django-plotly-dash*, que possibilita uma fácil criação de gráficos e painéis que são atualizados em tempo real utilizando apenas código Python, sem a necessidade de escrever código HTML, CSS ou Javascript. Isso possibilitou uma integração direta dos modelos criados no Django com os gráficos. A Figura 3 mostra alguns exemplos de gráficos desenvolvidos com o *django-plotly-dash* para o projeto.

Figura 3 – Gráficos criados pelo django-plotly-dash



Fonte: Arquivo dos autores (2020)

Quanto aos modelos mencionados anteriormente, foi criado um modelo para cada informação extraída das APIs externas, estando eles associados à posição dos ônibus, velocidade das vias, linhas e paradas de ônibus, informações das linhas de trens e metrôs, informações meteorológicas e eventos. Todos os modelos estão associados a um banco de dados PostgreSQL, que por possuir uma melhor integração com Python e Django, além de se tratar de um banco de dados relacional e rápido, foi a opção escolhida para o projeto.

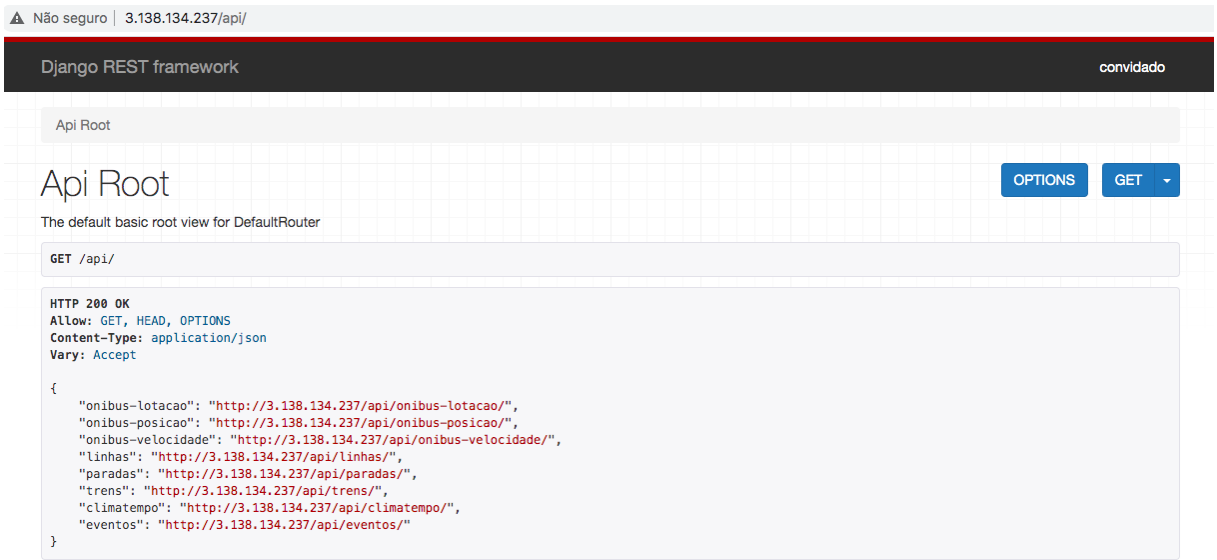
3.3 APIs com Django REST *Framework*

O Django REST *Framework* é uma biblioteca para o *Framework* Django que disponibiliza funcionalidades para implementar APIs REST de forma extremamente rápida e fácil. É extremamente simples configurar e criar rotas que aceitam todos os verbos HTTP se comunicando diretamente com o banco de dados e atendendo ao CRUD (*Create* (Criação), *Read* (Consulta), *Update* (Atualização) e *Delete* (Destruição)) (ESTEVAO, 2016).

Outra grande vantagem, além da facilidade de implementação, é a criação automática de uma plataforma *web* que centraliza todas as possibilidades de requisições de todas as rotas disponíveis. Nela é possível escolher a forma que deseja visualizar os dados, seja como JSON ou XML o tipo de retorno.

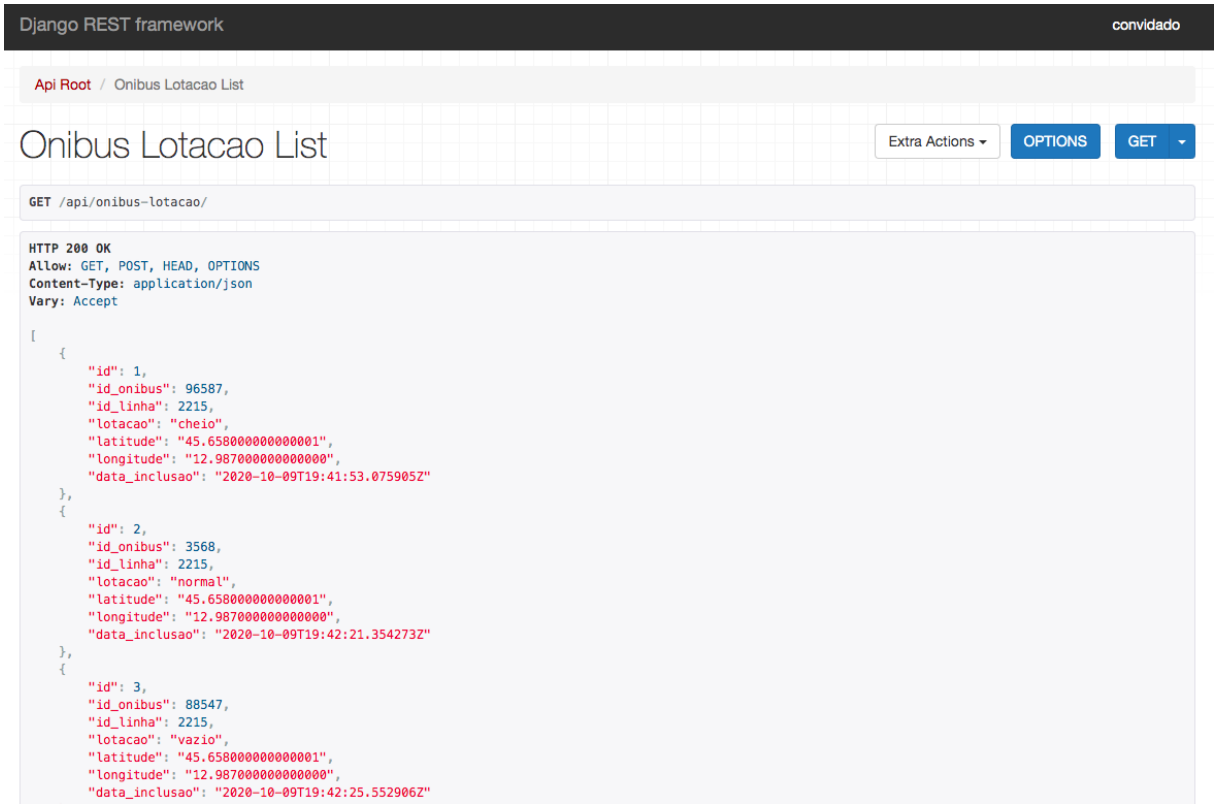
Com as API's implementadas, é possível acessar os dados da aplicação de forma independente do *dashboard*. Assim, outras pessoas podem ter acesso a esses dados para novas e futuras análises. Alguns dos *endpoints* criados disponibilizam dados como de velocidade, posição, linhas, paradas, metrô, trem e clima.

Figura 4 – Rotas das APIs dentro da plataforma *web* disponibilizada pelo Django



Fonte: Arquivo dos autores (2020)

Figura 5 – Exemplo de retorno JSON do *endpoint* /api/onibus-lotacao



Fonte: Arquivo dos autores (2020)

Na Figura 4, é possível visualizar uma lista com todas as rotas disponíveis. Um exemplo de requisição nessas rotas pode ser visualizado na imagem 5, onde é realizada uma requisição do tipo *GET* no *endpoint* 'onibus-lotacao'. A resposta é do tipo JSON, retornando todas as informações disponíveis como, id, id_onibus, id_linha, lotação, latitude, longitude, data_inclusao.

3.4 *Endpoints*

Na imagem 4, é possível visualizar todos os *endpoints* disponíveis, cada um deles possui suas informações e funções. No apêndice 6, uma breve descrição de cada uma delas, e como utilizá-las com os verbos *GET* e *POST*, assim como uma prévia do retorno das informações. Foi escolhido apresentar as requisições *GET* e *POST* pois são através delas que o *dashboard* consome os dados e que as APIs externas salvam as informações dentro do nosso banco de dados.

3.5 Infraestrutura de nuvem

Pela necessidade de estar disponível a qualquer momento e de capturar o máximo de informação possível, o sistema foi disponibilizado na nuvem, utilizando a plataforma da Amazon, principalmente os recursos da *Elastic Cloud Compute* (EC2) (AWS, 2020c). A infraestrutura utilizada gira em torno de uma máquina Linux, da categoria t2.medium, que se provou o mínimo necessário para executar o nosso sistema e aguentar a carga de algumas visitas simultâneas.

Começando pela máquina, as instâncias EC2 t2.medium são máquinas com 2 núcleos dedicados, 4GB de memória RAM, indicadas tanto para o uso como plataforma de desenvolvimento quanto para servidores *web*. Suas características são processadores com um *clock* relativamente alto e ao mesmo tempo um custo baixo em comparação com as outras soluções oferecidas pela Amazon, como instâncias t3 ou A1.

Figura 6 – Tipos de instância

T2	T3	A1
Arquitetura x86, *base clock* mais alto que instâncias T3 (3.3 Ghz), capacidade razoável de escalar o processamento da instância em trabalhos intensos.	Arquitetura x86, base clock mais baixo que instâncias T2 (2.5 Ghz), grande capacidade de escalar o processamento da instância em trabalhos intensos.	Arquitetura ARM, capacidade razoável de escalar o processamento da instância, mais caro do que instâncias do tipo T2 e T3.
$0.04/Hora* (T2.medium, 2 vCPU e 4 GB de RAM)	$0.04/Hora* (T3.medium, 2 vCPU e 4 GB de RAM)	$0.05/Hora* (A1.large, 2 vCPU e 4 GB de RAM)

* Preços em dólar.

AWS, s.d. (https://aws.amazon.com/ec2/instance-types/)

Foi decidido usar a instância t2.medium após realizar testes tanto com a t2.micro, que é gratuita para as 750 primeiras horas por mês, quanto a t2.small, que é totalmente paga, mas mais econômica que a t2.medium. A modalidade micro apresentou dificuldades logo nos primeiros instantes, apesar de ser capaz de executar o *dashboard* em Django paralelo ao banco de dados PostgreSQL, quando as tasks do Celery para captura de dados começaram a ser executadas, o processamento disponível para ela não foi o suficiente, atingindo 100% de uso do núcleo disponível e deixando o sistema inutilizável, mesmo com uma única pessoa acessando. Já a instância t2.small se mostrou capaz de executar as *tasks* simultaneamente ao *dashboard* e o banco de dados, porém ao começar a receber acessos no *dashboard* o sistema não conseguia dar conta das requisições e apresentava uma constante lentidão.

Com isso, conclui-se que a instância mínima para rodar o projeto com segurança é a t3.medium, e mesmo assim rapidamente todo o seu armazenamento foi consumido em pouco menos de uma semana. Após isso o disco foi expandido de 7GB para 50GB, o que daria uma margem de segurança até a conclusão do projeto. Apesar da instância média ser o suficiente para o uso durante o desenvolvimento, sua configuração não seria o suficiente para um momento de fluxo constante no sistema por conta do uso do processador, que se aproxima de 100% sempre que tem mais de dois navegadores acessando simultaneamente, por isso durante o Eureka, foi utilizada a instância t2.2xlarge, com 8 vCores e 32GB de memória RAM, o que se mostrou suficiente para momentos com cerca de 10 acessos simultâneos.

Figura 7 – Configurações da instância

T2.micro	T2.small	T2.medium	T2.2xlarge
1 vCPU	1 vCPU	2 vCPU	8 vCPU
1 GB de RAM	2 GB de RAM	4 GB de RAM	32 GB de RAM
$0,0116/Hora* (primeiras 750 horas grátis)	$0,023/Hora*	0,0464/Hora*	$0,3712/Hora*

* Preços em dólar.

AWS, s.d. (https://aws.amazon.com/ec2/instance-types/)

Outra forma de contornar o problema de eficiência da instância seria trabalhando melhor a parte do banco de dados. Como a captura de informações do sistema é feito em tempo real, temos intervalos muito pequenos de ociosidade do banco de dados, e quando os acessos faziam alguma consulta na tabela ocorrem picos de processamento em que a instância não tinha capacidade de absorver. As soluções possíveis seriam o uso de um banco de dados não relacional ou o uso de um banco de dados sendo executado a parte do sistema, como em outra instância ou em um serviço como o AWS RDS. Apesar de bancos noSQL, como MongoDB ou o Cassandra, favorecerem usos em situações em que existem dados coletados em tempo real e em grande escala (MONGODB, s.d.), para um sistema desenvolvido em Django acarretaria em profundas alterações no projeto, dado que o framework já conta com um ORM (Mapeamento objeto-relacional, do inglês Object-relational mapping), que trabalha unicamente com bancos relacionais (LIMA, 2020). A forma mais factível de resolver esse problema otimizando o banco de dados seria transferindo ele para um RDS, o que criaria um novo custo, mas com a

possibilidade de redução de gastos com a instância EC2, porém, dado a proximidade da Eureka, não existiu tempo hábil entre a detecção do problema e a data de apresentação do sistema.

Para o acesso à instância pelo navegador, uma implantação simples do NGINX (NGINX, 2020) foi utilizada, um servidor web muito popular por sua facilidade de uso e segurança. O servidor é o primeiro lugar que as requisições web chegam na instância e o NGINX distribui essas requisições para cada local definido em suas configurações. No sistema desenvolvido existe uma rota só para os arquivos estáticos e outra para o sistema em si. Isso permite uma velocidade de resposta maior que se todos os elementos estivessem na mesma rota e facilita a manutenção dos arquivos do sistema.

4 Testes e Resultados

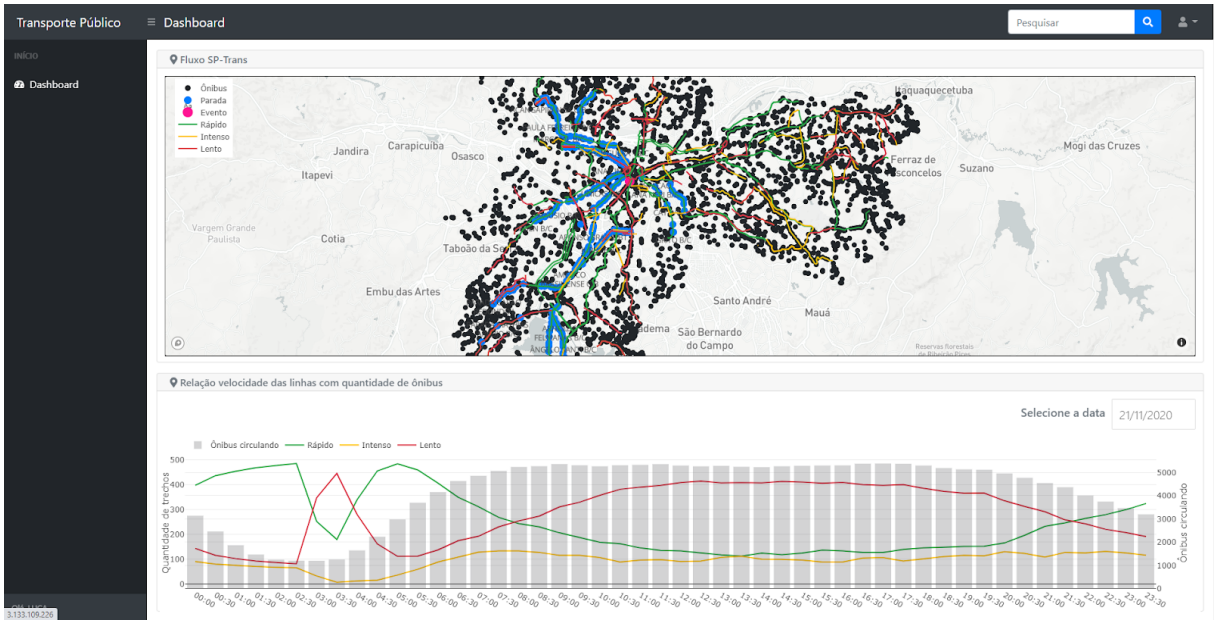
Os resultados do projeto podem ser divididos em três grupos principais, resultados percebidos pelos usuários dos ônibus, pelos gestores das linhas e por último, por desenvolvedores que tem interesse em usar a estrutura desenvolvida. O principal benefício para quem usa o sistema de ônibus de São Paulo, que pode ser sentido durante a apresentação na Eureka, foi na experiência com o mapa em tempo real e o acompanhamento das linhas do metrô dentro do *dashboard*. Para quem usa diariamente o serviço de transporte público da capital, poder acompanhar a situação de cada trecho dos corredores do ônibus, suas posições e no mesmo lugar, consultar como estava as linhas de metrô é algo de grande valor. Apesar da ferramenta não permitir, grande parte das pessoas expressou a vontade de consultar o *dashboard* em sua rotina, para conferir a situação de uma linha que é utilizada no seu trajeto, e alguns indo além, procurando explorar outros serviços, por exemplo aplicativos de carros, como Uber e Cabify.

Para gestores dos serviços, a possibilidade de ter em uma única ferramenta informações essenciais para a tomada de decisão durante a operação das linhas é algo fundamental, principalmente em tempo real. Os gestores podem verificar uma redução da velocidade média de algum trecho e comparar com o histórico de outros dias, e ao mesmo tempo verificar se há alguma ocorrência de um grande evento na região, ou se a situação meteorológica está afetando o tráfego na via, como em uma enchente por decorrência de fortes chuvas.

Também para empresas que tiverem interesse de implementar a ferramenta, a arquitetura que escolhemos, de Django mais Plotly em conjunto da nossa API para a consulta dos dados armazenados, permite que com poucas linhas de Python e SQL sejam desenvolvidas novas visões que o gestor sinta a necessidade de acompanhar. É possível implementar coisas como volume de veículos em zonas específicas da cidade ou velocidade média em função da temperatura de uma região, por exemplo, podem ser implementadas com conhecimento básico de Python e PostgreSQL.

Já para os desenvolvedores que tem interesse no assunto de transporte público, a ferramenta fornece endpoints públicos, que permitem consultas que nem sempre são possíveis nos serviços oficiais, principalmente no caso das APIs da SPTrans. Foram desenvolvidos caminhos que permitem não só consultas em tempo real, mas também informações históricas, que inclusive durante o desenvolvimento do projeto fizeram falta para análises mais profundas. As informações são disponibilizadas de forma agnóstica por meio de APIs, com as consultas podendo ser feitas com o uso de qualquer linguagem que suporte requisições HTTP, não sendo limitada pelo uso do Python ou SQL. Além disso, todos os dados já são tratados antes de serem armazenados, mesmo em situações como dos Eventos, em que a captura deles foi feita através de raspagem de tela, com auxílio do Selenium.

Figura 8 – Tela do *dashboard*



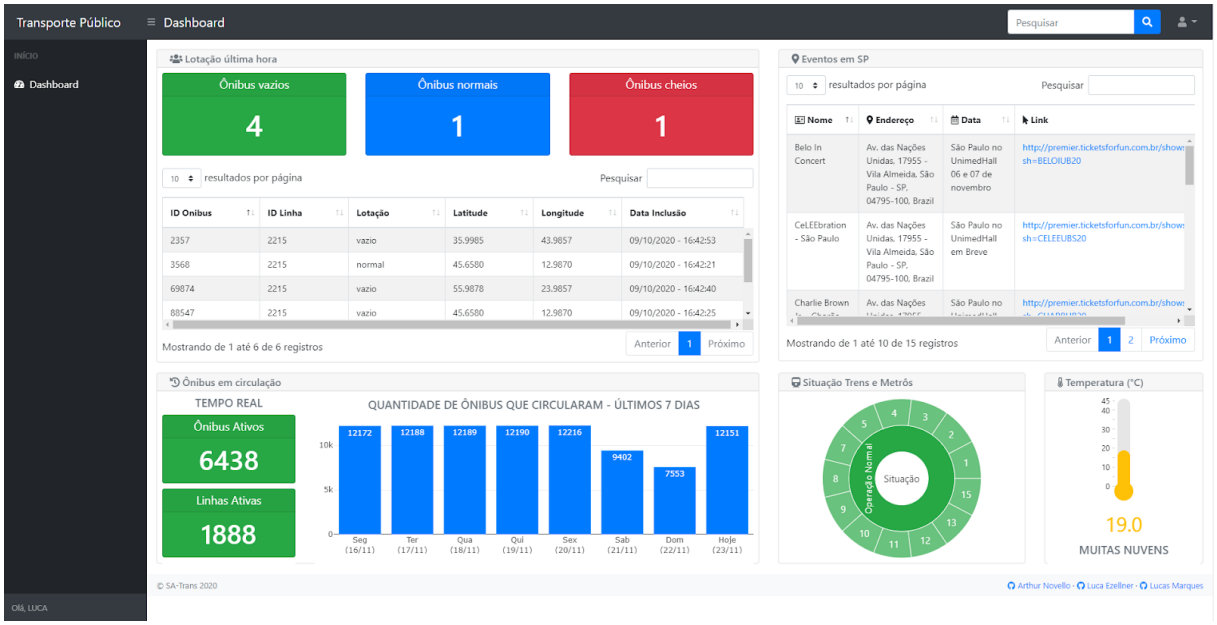
Fonte: Arquivo dos autores (2020)

Na Figura 8 é possível observar um mapa com os ônibus da cidade de São Paulo em tempo real, juntamente com informações das suas respectivas linhas, tais como nome da via, velocidade da via, trecho, velocidade do trecho, extensão e tempo para percorrer o trecho. Além disso, o usuário também pode ver os eventos que ocorrerão na cidade e a situação do trânsito das linhas de uma forma ampla.

Um pouco mais abaixo o usuário tem acesso a informações históricas da quantidade de ônibus que circularam por período e o relativo o número de linhas que possuiam trânsito rápido, intenso e lento nesse mesmo intervalo de tempo.

Na Figura 9 é possível observar a lotação dos ônibus, o detalhamento dos eventos que acontecerão em São Paulo, a quantidade de ônibus e linhas ativas no momento, o histórico de quantos ônibus circularam por dia na última semana, a situação das linhas de trens e metrôs e a condição metereológica em tempo real.

Figura 9 – Tela do *dashboard*



Fonte: Arquivo dos autores (2020)

5 Conclusão e Trabalhos futuros

Considerando o objetivo deste projeto de reunir diversas informações de fontes diferentes, disponibilizá-las de uma forma prática e transparente para toda a população e para gestores da SPTrans como ferramenta de apoio à decisão, pode-se concluir que o *dashboard* desenvolvido atende às demandas citadas. Por meio dele, painéis e gráficos em tempo real exibem informações relevantes do transporte público, que são facilmente acessadas por qualquer cidadão por meio de uma página web.

Além disso, a possibilidade de implementação do sistema em outras empresas de transporte público e privado, juntamente com a diversificação dos dados obtidos e armazenados, aumentam a aplicabilidade do projeto. Vale ainda ressaltar que como próximos passos, a aplicação ainda pode ser continuamente aprimorada com a análise dos dados armazenados, que irão aumentar ao longo do tempo.

Tendo em vista que a aplicação salva dados diariamente, a quantidade de informações armazenadas tende a crescer rapidamente. Com isso, o espaço em disco, memória e CPU necessários na instância EC2 que armazena o banco de dados precisaria aumentar paralelamente aos dados. Por esse motivo, seria válido que o banco de dados ficasse em uma instância apartada da instância de aplicação, garantindo maior escalabilidade e persistência dos dados. Esse objetivo poderia ser alcançado utilizando o serviço RDS da AWS.

Como dito anteriormente, outra forma de resolver essa questão do aumento de dados armazenados pelo banco, é com o uso de uma plataforma noSQL, seja na mesma instância que a aplicação principal, ou em um serviço como o DynamoDB, que é o equivalente ao RDS para bancos de dados não relacionais. Bancos noSQL são reconhecidos pela sua maior capacidade em lidar com grandes quantidades de dados, estruturados ou não, e em tempo real.

Juntamente com a criação de uma nova instância para o banco de dados, considerando que a aplicação já esteja armazenando informações há um longo período, também seria viável um estudo mais aprofundado das informações e a aplicação de técnicas de inteligência artificial, por meio das quais seria possível a extração de informações como a previsão de velocidade e atrasos de uma via, previsão de frota, tempo, trânsito etc. Essas e outras informações poderiam ser obtidas, analisadas e incluídas no painel.

Por fim, outra informação que agregaria valor no projeto seria a lotação dos ônibus, que poderia ser extraída por meio de análise de imagens de câmeras já existentes, instaladas dentro dos veículos. Essas imagens passariam por um tratamento e treinamento em uma rede artificial que seria utilizada pelo yolo, que por sua vez identificaria a quantidade de pessoas presentes dentro de cada veículo, classificando cada ônibus como “vazio”, “normal” ou “cheio”.

6 APÊNDICE

6.1 *ENDPOINTS*

6.1.1 /api/onibus-lotacao/

GET


```
{
    "id": 1,
    "id_onibus": 96587,
    "id_linha": 2215,
    "lotacao": "cheio",
    "latitude": "45.658000000000001",
    "longitude": "12.987000000000000",
    "data_inclusao": "2020-10-09T19:41:53.075905Z"
}
```


POST

```
{
    "img": caminho_img,
    "id_onibus": id_onibus,
    "id_linha": id_linha,
    "latitude": latitude,
    "longitude": longitude
}
```

6.1.2 /api/onibus-posicao/

GET


```
{
    "quantidade": 11992
}
```


POST

```
{
    "o": [
        {
            "id_onibus": id_onibus,
            "onibus_deficiente": onibus_deficiente,
            "horario_atualizacao_localizacao": horario_atualizacao_localizacao,
            "latitude": latitude,
            "longitude": longitude,
            "id_linha": id_linha,
            "frota": frota,
        }
    ]
}
```

6.1.3 /api/onibus-velocidade/

GET

```
{
    "id": 1067508,
    "nome": "BUTANTA (BAIRRO - CENTRO)",
    "vel_trecho": 27,
    "vel_via": 27,
    "trecho": "de R. AMARO CAVALHEIRO ate R. PAES LEME",
    "extensao": 650,
    "tempo": "00:01",
    "coordenadas": [
        {
            "latitude": "-23.567653",
            "longitude": "-46.695722",
            "id": 6217
        },
        {
            "latitude": "-23.568001",
            "longitude": "-46.696452",
            "id": 6218
        },
        {
            "latitude": "-23.568063",
            "longitude": "-46.696595",
            "id": 6219
        }
}
```


POST

```
{
    "o": [
        {
            "name": nome,
            "description": {
                'vel_trecho': vel_trecho,
                'vel_via': vel_via,
                'trecho': trecho,
                'extensao': extensao,
                'tempo': tempo
            },
            "coordinates": {
                'lat': latitude,
                'lon': longitude
            }
        }
    ]
}
```

6.1.4 /api/linhas/

GET


```
{
    "id_linha": 264,
    "letreiro": "509J-10",
    "sentido": 1,
    "letreiro_destino": "PQ. IBIRAPUERA",
    "letreiro_origem": "JD. SELMA"
}
```

POST

```
{
    "l": [
        {
            "id_linha": id_linha,
            "letreiro": letreiro,
            "sentido": sentido,
            "letreiro_destino": letreiro_destino,
            "letreiro_origem": letreiro_origem,
        }
    ]
}
```

6.1.5 /api/paradas/

GET

```
{
    "id": 1,
    "id_parada": 4203724,
    "nome": "",
    "endereco": "R. Agamenon Pereira da Silva",
    "latitude": "-23.692865000000001",
    "longitude": "-46.778350000000003"
}
```

POST

```
{
    "p": [
        {
            "id_parada": id_parada,
            "nome": nome,
            "endereco": endereco,
            "latitude": latitude,
            "longitude": longitude,
        }
    ]
}
```

6.1.6 /api/trens/

GET

```
{
    "id": 1,
    "id_linha": 1,
    "data_ocorrencia": "2020-10-07T07:42:01.039607Z",
    "descricao": null,
    "ultima_atualizacao": "2020-10-07T21:14:01.162598Z",
    "situacao": "Operacao Normal"
}
```

POST

```
{
    "t": [
```

```
        {
            "id_linha": id_linha,
            "data_ocorrencia": data_ocorrencia,
            "descricao": descricao,
            "ultima_atualizacao": ultima_atualizacao,
            "situacao": situacao,
        }
    ]
}
```

6.1.7 /api/climatempo/

GET

```
{
    "id_cidade": 3477,
    "temperatura": "27.00",
    "direcao_vento": "S",
    "velocidade_vento": "9.00",
    "umidade": "66.00",
    "condicao": "Nuvens esparsas",
    "pressao": "1015.00",
    "sensacao": "28.00",
    "date": "2020-10-07T21:14:34.451818Z"
}
```

POST

```
{
    "ct": [
        {
            "id_cidade": id_cidade,
            "temperatura": temperatura,
            "direcao_vento": direcao_vento,
            "velocidade_vento": velocidade_vento,
            "umidade": umidade,
            "condicao": condicao,
            "pressao": pressao,
            "sensacao": sensacao,
        }
    ]
}
```

6.1.8 /api/eventos/

GET

```
{
    "id": 26,
    "nome": "Maria Bethania",
    "link": "http://premier.ticketsforfun.com.br/shows/show.aspx?sh=MARIBUB19",
    "data_info": "Sao Paulo no UnimedHall 27 de junho",
    "data": "2020-06-27",
    "endereco": "Av. das Nacoes Unidas, 17955 - Vila Almeida, Sao Paulo - SP, 04795-100, Brazil",
    "latitude": "-23.647672600000000",
    "longitude": "-46.723812100000004",
    "data_inclusao": "2020-10-14T00:17:58.612131Z"
}
```

POST

```
{
    "e": [
        {
            "nome": nome,
            "link": link,
            "data_info": data_info,
            "data": data,
            "endereco": endereco,
            "latitude": latitude,
            "longitude": longitude
        }
    ]
}
```

6.2 Código fonte do projeto

O código fonte do projeto está disponível no GitHub e pode ser acessado através do link <https://github.com/TransportePublico-IMT/tcc_TransporteUrbano> ou via QRCode

Figura 10 – QRCode do GitHub do projeto

Fonte: Arquivo dos autores (2020)

Referências

AWS. **About AWS**. 2020. Disponível em: <https://aws.amazon.com/about-aws/>. 23

AWS. **Amazon Web Services (AWS) Serviços de computação em nuvem**. 2020. Disponível em: <https://aws.amazon.com/pt/>. 29

AWS. **Visão geral Amazon EC2**. 2020. Disponível em: <https://aws.amazon.com/pt/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc>. 33

AWS. **Amazon RDS**. s.d. Disponível em: <https://aws.amazon.com/pt/rds/>. 24

AWS. **Amazon RDS: Create DB Instance**. s.d. Disponível em: <https://docs.aws.amazon-.com/pt_br/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.htm>. 24

AWS. **Amazon SQS Developer Guide**. s.d. Disponível em: <https://docs.aws.amazon.com-/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html>. 24

AWS. **Instance Types**. s.d. Disponível em: <https://aws.amazon.com/ec2/instance-types/>. 24

AWS. **Supported operating systems**. s.d. Disponível em: <https://docs.aws.amazon.com-/systems-manager/latest/userguide/prereqs-operating-systems.html>. 24

BARR, J. **Amazon EC2 Beta**. 2006. Disponível em: <https://aws.amazon.com/blogs/aws-/amazon_ec2_beta/>. 23

BARR, J. **My First 12 years at amazon.com**. 2014. Disponível em: <http://jeff-barr.com-/2014/08/19/my-first-12-years-at-amazon-dot-com/>. 24

Ben Ayed, A.; Ben Halima, M.; ALIMI, A. M. Big data analytics for logistics and transportation. **2015 4th IEEE International Conference on Advanced Logistics and Transport, IEEE ICALT 2015**, p. 311–316, 2015. 18

BRANDON, J. **Complet List Of Amazon Web Services**. 2020. Disponível em: <https://www.techradar.com/news/aws#complete-list-of-amazon-web-services>. 23

CELERY. **Celery - Distributed Task Queue — Celery 5.0.2 documentation**. 2018. Disponível em: <https://docs.celeryproject.org/en/stable/>. 21, 29

COPPOLA, D. **Development of Amazon Web Services Revenue**. 2020. Disponível em: <https://www.statista.com/statistics/233725/development-of-amazon-web-services-revenue/>. 23

DEMORA. **Demora do ônibus**. 2018. Disponível em: <https://www.reclameaqui.com.br-/sptrans/demora-do-onibus_3dNERaztrwT9ckEK/>. 16

DEMORA. **Demora da linha**. 2020. Disponível em: <https://www.reclameaqui.com.br/sptrans-/demora-da-linha_AZuNpkdOrYwgBj-z/>. 16

DJANGO. **Django Plotly Dash**. 2020. Disponível em: <https://django-plotly-dash.readthedocs-.io/en/latest/introduction.html>. 21

DJANGO. **Django Rest Framework**. 2020. Disponível em: <https://www.django-rest-framework.org/>. 21

DJANGO. **The Web framework for perfectionists with deadlines Django**. 2020. Disponível em: <https://www.djangoproject.com/>. 20, 29

DUHIGG, C. **How Companies Learn Your Secrets**. 2012. Disponível em: <https://www-.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=1\&_>. Acesso em: 2020-04-26. 26

EISEN, D. **Marriott bets on predictive analytics for brand growth**. 2018. Disponível em: <https://www.hotelmanagement.net/tech/marriott-builds-its-brands-by-knowing-more-about-you>. Acesso em: 2020-04-26. 26

ESTEVAO. **Principais verbos HTTP utilizados em um serviço REST**. 2016. Disponível em: <https://www.devmedia.com.br/servicos-restful-verbos-http/37103>. 31

FOOTE, K. D. **A Brief History of Analytics**. 2018. Disponível em: <https://www.dataversity-.net/brief-history-analytics/>. Acesso em: 2020-04-26. 25

FREAK, C. **Top 10 AWS Services according to popularity**. 2019. Disponível em: <https://medium.com/faun/top-10-aws-services-according-to-popularity-bd78eea2f7a9>. 23

G1. **Pesquisa traça perfil dos usuários do transporte público em São Paulo** . 2013. Disponível em: <http://g1.globo.com/sao-paulo/anda-sp/noticia/2013/06/pesquisa-traca-perfil-dos-usuarios-do-transporte-publico-em-sao-paulo.html>. 16

GOOGLE. **The General Transit Feed Specification (GTFS)**. 2020. Disponível em: <https://developers.google.com/transit/gtfs/>. 18

GOOGLE. **GMP: Get Started**. s.d. Disponível em: <https://developers.google.com/maps/gmp-get-started>. 25

HIGOR. **Introdução ao padrão MVC**. 2013. Disponível em: <https://www.devmedia.com.br-/introducao-ao-padrao-mvc/29308>. 21

III, S. F. **What Google Cloud Platform is and why you would use it**. 2019. Disponível em: <https://www.zdnet.com/article/what-google-cloud-platform-is-and-why-youd-use-it/>. 25

INTELIPOST. **A importância do dashboard para e-commerce**. 2017. Disponível em: <https://www.intelipost.com.br/blog/a-importancia-do-dashboard-para-e-commerce/>. Acesso em: 2020-04-28. 19

JSON. **JSON**. 2020. Disponível em: <https://www.json.org/json-pt.html>. 29

KLEVERTON. **O que é UX – User Experience**. 2019. Disponível em: <https://www.intelipost-.com.br/blog/a-importancia-do-dashboard-para-e-commerce/>. Acesso em: 2020-05-01. 20

KNORR, E. **What is cloud computing?** 2018. Disponível em: <https://www.infoworld.com-/article/2683784/what-is-cloud-computing.html>. 22, 23

LEE, I. Big data: Dimensions, evolution, impacts, and challenges. **Business Horizons**, "Kelley School of Business, Indiana University", v. 60, n. 3, p. 293–303, 2017. ISSN 00076813. Disponível em: <http://dx.doi.org/10.1016/j.bushor.2017.01.004>. 18

LIMA, G. **Django Querysets e ORM**. 2020. Disponível em: <https://www.alura.com.br-/artigos/django-query-sets-e-orm>. 34

LOBO, R. **Top 10 das linhas de ônibus mais movimentadas de São Paulo** . 2013. Disponível em: <https://viatrolebus.com.br/2013/07/top-10-das-linhas-de-onibus-mais-movimentadas-de-sao-paulo/>. 16

MAGALHAES, A. **O que é e para que serve o arquivo XML Canaltech**. 2020. Disponível em: <https://canaltech.com.br/software/xml-o-que-e/>. 29

MONGODB. **Advantages of NoSQL Databases**. s.d. Disponível em: <https://www.mongodb-.com/nosql-explained/advantages>. 34

NGINX. **NGINX High Performance Load Balancer, Web Server, Reverse Proxy**. 2020. Disponível em: <https://www.nginx.com/>. 35

NTURBANO. Só 0,01% do Orçamento Federal para o transporte coletivo urbano. **Revista NTUrbano**, 2019. Disponível em: <https://www.ntu.org.br/novo/NoticiaCompleta-.aspx?idArea=10\&idNoticia=1>. Acesso em: 2020-04-18. 15

PEDUZZI, P. **Estudo do Ipea mostra que 65% da população usam transporte público nas capitais**. 2011. Disponível em: <https://memoria.ebc.com.br/agenciabrasil/noticia/2011-05-04-/estudo-do-ipea-mostra-que-65-da-populacao-usam-transporte-publico-nas-capitais>. Acesso em: 2020-04-18. 15

PENA, R. F. A. **Problemas no transporte público**. s.d. Disponível em: <https://brasilescola-.uol.com.br/geografia/problemas-no-transporte-publico.htm>. Acesso em: 2020-04-18. 15

PINHEIRO, F. **Entendendo o MTV do Django**. 2020. Disponível em: <https://www-.treinaweb.com.br/blog/entendendo-o-mtv-do-django/>. 20

POSTGRESQL. **PostgreSQL: The world's most advanced open source database**. 2020. Disponível em: <https://www.postgresql.org/>. 29

PYPI. **Find, install and publish Python packages with the Python Package Index**. 2020. Disponível em: <https://pypi.org/>. 21

SAMPA, M. **Números da Mobilidade Urbana em São Paulo e região metropolitana** . 2018. Disponível em: <https://mobilidadesampa.com.br/mobilidade-urbana/>. 16

SANTOS, G. **Transporte público precisa de R$ 235 bi em investimento, estima BNDES**. 2015. Disponível em: <https://www1.folha.uol.com.br/mercado/2015/09/1685379-transporte-publico-precisa-de-r-235-bi-em-investimento-estima-bndes.shtml>. Acesso em: 2020-04-18. 15

SAUNDERS, B. **Who's Using Amazon Web Services?** 2020. Disponível em: <https://www-.contino.io/insights/whos-using-aws>. 23

SCHULTZ, F. **O que é um endpoint: Como proteger este tipo de dispositivo Milvus**. 2020. Disponível em: <https://milvus.com.br/o-que-e-endpoint/>. 29

SILVA, G. **Celery e Tarefas assícronas com Python**. 2016. Disponível em: <https://geoadmin.com.br/celery-python-/#:~:text=O%20Celery%20%C3%A9%20um%20aplica%C3%A7%C3%A3o,workers%2C%20localizados%2 .&text=O%20Celery%20permite%20o%20uso,brokers%2C%20como%20redis%20e%20RabbitMQ>. 22

STIEL, B. **Celery Execution Pools: What is it all about?** 2018. Disponível em: <https://www.distributedpython.com/2018/10/26/celery-execution-pool/>. 22

SYSTEMSAT. **DASHBOARD OPERACIONAL: TUDO O QUE VOCÊ PRECISA SABER**. s.d. Disponível em: <https://www.systemsat.com.br/dashboard-operacional/>. Acesso em: 2020-04-29. 20

TABLEAU. **O que é business intelligence? Seu guia sobre o BI e por que ele é importante**. 2020. Disponível em: <https://www.tableau.com/pt-br/learn/articles/business-intelligence>. 19

TAN, A. **How Coca-Cola uses data to supercharge its superbrand status**. 2017. Disponível em: <https://www.adma.com.au/resources/how-coca-cola-uses-data-to-supercharge-its-superbrand-status>. Acesso em: 2020-04-26. 26

TECHTUDO. **Como abrir mapas .KML, .KMZ, .CSV, .TSV, .GPX ou .XLSX no Google Maps | Dicas e Tutoriais | TechTudo**. 2016. Disponível em: <https://www.techtudo.com.br-/dicas-e-tutoriais/noticia/2016/11/como-abrir-mapas-kml-kmz-csv-tsv-gpx-ou-xlsx-no-google-maps.html>. 29

TECNICON. **5 benefícios do uso de dashboard na gestão empresarial**. 2019. Disponível em: <https://www.tecnicon.com.br/blog/438-5%7B/_%7Dbeneficios%7B/_%7Ddo%7B-/_%7Duso%7B/_%7Dde%7B/_%7Ddashboard%7B/_%7Dna%7B/_%7Dgestao%7B-/_%7Dempresarial>. Acesso em: 2020-04-24. 19, 20

VIEIRA, M. R.; BARBOSA, L.; KORMÁKSSON, M.; ZADROZNY, B. USapiens: A System for Urban Trajectory Data Analytics. **Proceedings - IEEE International Conference on Mobile Data Management**, v. 1, p. 255–262, 2015. ISSN 15516245. 18

VILLANOVA. **The Evolution of Data Collection and Analytics**. s.d. Disponível em: <https:-//taxandbusinessonline.villanova.edu/blog/the-evolution-of-data-collection-and-analytics/>. Acesso em: 2020-04-26. 26