ESCOLA POLITÉCNICA DA USP

Obtenção e análise de dados referentes à condução de carros

ARTHUR PIRES DA FONSECA - 10773096

ANTONIO PINHEIRO DA SILVA JUNIOR - 9004355

GABRIEL MORGHETT GABOARDI - 10773968

SÃO PAULO

2023

OBTENÇÃO E ANÁLISE DE DADOS REFERENTES À CONDUÇÃO DE CARROS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para o Trabalho de Conclusão de Curso em Engenharia de Computação. São Paulo 2023

Orientador: Prof. Dr. Edson Toshimi Midorikawa
Co-orientador: Prof. Dr. Reginaldo Arakaki

SÃO PAULO
2023

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Dedicatória

Arthur: Dedico este trabalho à Hihi e aos meus avós por terem cuidado de mim, ao meu irmão (o Di), e aos meus pais, Alfredo e Márcia.

Antonio: Dedico este trabalho ao meu amigo Mario Granziera e aos meus pais, Júlia e Antônio.

Gabriel: Dedico este trabalho à minha irmã, Giovanna Morghett Gaboardi, e aos meus pais, Dayse Morghett Gaboardi e Clovis Gaboardi, por todo o apoio que me deram.

Agradecimentos

Agradecemos ao Guilherme Migliati (https://www.linkedin.com/in/guimigli), que foi essencial para superar o obstáculo inicial no desenvolvimento do aplicativo, oferecendo suporte técnico, que permitiu a compilação dele.

Além disso, a contribuição significativa do Dr.-Ing. Philippe Jardin (https://www.linkedin.com/in/dr-ing-philippe-jardin-964733a3), que disponibilizou o simulador de OBD quando o Arthur ainda estava na Alemanha, ampliou consideravelmente as capacidades de teste do sistema.

Epígrafe

"Só se pode alcançar um grande êxito quando nos mantemos fiéis a nós mesmos."

-Friedrich Nietzsche[38]

"Liberdade não é fazer o que se quer, mas querer o que se faz."

-Jean-Paul Sartre[39]

"If you want something you've never had, you have to do something you've never done"

-Thomas Jefferson

Resumo

Com o objetivo de criar uma plataforma complementar ao motorista e seu veículo, este projeto aproveitou a infraestrutura presente em carros e *smartphones* para ser realizado.

A coleta de informações do veículo foi realizada por meio da porta OBD-II, presente em todos os veículos fabricados a partir de 2010 no Brasil, enquanto dados do celular incluíram informações cruciais sobre a movimentação do veículo, como geolocalização e aceleração.

A subsequente transferência desses dados para a nuvem da AWS viabilizou a análise do perfil de cada condutor. Essa abordagem não apenas proporciona uma compilação de dados provenientes de diferentes fontes, mas também permite uma avaliação aprofundada das preferências e comportamentos individuais de condução, contribuindo para uma compreensão mais completa do desempenho veicular e dos padrões de uso.

Abstract

With the aim of creating a complementary platform for the driver and his vehicle, this project took advantage of the infrastructure present in cars and *smartphones* to be carried out.

Vehicle information was collected through the OBD-II port, present in all vehicles manufactured from 2010 onwards in Brazil, while cell phone data included crucial information about the vehicle's movement, such as geolocation and acceleration.

The subsequent transfer of this data to the AWS cloud made it possible to analyze the profile of each driver. This approach not only provides a compilation of data from different sources, but also allows for an in-depth assessment of individual driving preferences and behaviors, contributing to a more complete understanding of vehicle performance and usage patterns.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

API	*Application Programming Interface*
AWS	*Amazon Web Services*
GiB	*Gibibyte*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
JSON	*Java Script Object Notation*
MVP	*Minimum Viable Product*
OBD-II	*On-board diagnostics*
PID	*Parameter Identifier*
RDS	*Amazon Relational Database Service*
RPM	*Revoluções por minuto*

CAPÍTULO 1

INTRODUÇÃO

Este documento detalha a concepção e implementação de um sistema dedicado à aquisição e análise de dados relacionados à condução de um motorista.

A abordagem central consiste na extração de informações provenientes de sensores já presentes em veículos contemporâneos, acessando a porta OBD-II, que adota um padrão internacional para o diagnóstico de parâmetros internos de um automóvel[2].

Adicionalmente, são coletadas informações fornecidas por um dispositivo móvel, incluindo dados de geolocalização e acelerômetros, e essas informações são centralizadas em uma base de dados hospedada na AWS.

Esta arquitetura permite uma análise posterior aprofundada desses dados, promovendo *insights* valiosos sobre os padrões e comportamentos de direção.

1.1 Motivação

Buscando traçar o perfil de condução de um motorista e desenvolver uma categorização clara para os condutores, este projeto empenha-se na criação de uma infraestrutura dedicada à captação e análise de dados em veículos de uso pessoal.

Com base nos dados de aceleração, é possível classificar o estilo de direção, identificar comportamentos potencialmente perigosos no trânsito e antecipar a necessidade de manutenção do veículo. Ao integrar a análise dos dados ao sistema de perfilamento, não apenas contribui-se para a segurança do veículo, mas também viabiliza-se a personalização de *feedbacks* e sugestões ao condutor, promovendo uma condução mais eficaz e segura.

Uma investigação preliminar sobre o tema revelou a existência de uma patente para um produto semelhante, registrada em 2013, que avalia o desempenho do

motorista a partir de dados previamente coletados de parâmetros relevantes à condução do veículo[1].

1.2 Objetivo

Este trabalho procura criar uma plataforma de disponibilização e captura de dados em um carro, coletando as informações *in loco* e apresentando estatísticas relevantes derivadas do que foi coletado.

Para isso, foi feito uso da infraestrutura já presente em carros atuais. Conforme pode ser visto na imagem 1, alguns carros podem ter dezenas de sensores embarcados dentro de si.

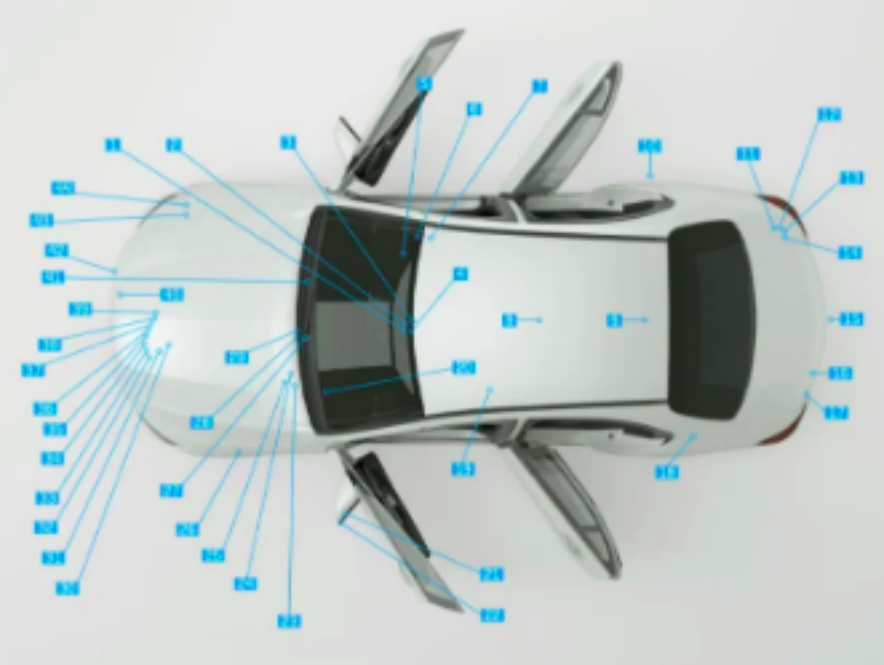

Figura 1: Veículos modernos estão equipados com dezenas de sensores[2].

Essas informações foram complementadas com o uso de um *smartphone*, o qual disponibiliza outros tipos de dados graças a seus sensores embarcados e à sua conexão com os satélites de GPS.

A análise da aceleração desempenha um papel essencial na avaliação do comportamento do motorista em sistemas de coleta de dados. Ao monitorar os padrões de aceleração, é possível extrair informações valiosas sobre a condução, tais como a suavidade nas transições de velocidade, a resposta a alterações nas condições de tráfego e o nível de agressividade ao volante.

A aceleração é um parâmetro que revela a habilidade do motorista em manter uma condução estável e antecipar suas ações em situações específicas.

Por isso, a fase de análise dos dados coletados deste projeto focou em analisar esta grandeza física para perfilar a direção de um motorista.

O trabalho também utilizou dados de violência no Brasil para contribuir com a classificação do que seria "dirigir com segurança".

Uma possível aplicação desse sistema encontra-se na área das empresas de seguro automotivo. Os dados e análises disponibilizados pelo aplicativo podem ser empregados na elaboração de políticas de precificação de seguros de veículos.

Essa abordagem oferece a possibilidade de estabelecer um preço mais personalizado e justo para cada indivíduo, promovendo, assim, comportamentos mais responsáveis no trânsito.

1.3 Justificativa

No ano de 2021, 11.647 pessoas perderam a vida em acidentes de trânsito no Brasil[6]. Além disso, a nível global, a média anual de fatalidades no trânsito atinge 1,35 milhão de pessoas[7], um número comparável às mortes por Covid-19 até abril de 2022[8].

Diante desses dados alarmantes, a importância deste projeto torna-se evidente, uma vez que estabelece métricas significativas para a avaliação do comportamento dos motoristas. Essas métricas não apenas contribuem para a prevenção de acidentes, mas, sobretudo, para a preservação da vida humana. O projeto emerge como uma ferramenta valiosa na promoção da segurança nas ruas e na busca por soluções que possam mitigar essas estatísticas trágicas.

Os veículos autônomos estão rapidamente conquistando popularidade, de forma que eles tornarem-se a maioria nas estradas é uma realidade iminente. Prevê-se, de fato, que carros com pelo menos nível 4 de automação tornem-se mais comuns a partir de 2025[9].

Dessa forma, embora a influência humana possa perder relevância em acidentes do futuro, torna-se imperativo estabelecer métricas para avaliar a condução de veículos autônomos.

Essa abordagem é essencial para assegurar a segurança e eficiência desses automóveis, promovendo um padrão consistente de avaliação que contribua para a confiança generalizada no crescente uso de tecnologias autônomas.

Este projeto tem, portanto, extrema relevância, pois é uma possível ferramenta para reduzir as fatalidades no trânsito, independentemente de serem causadas por condutores humanos ou por veículos autônomos.

1.4 Organização do trabalho

Os próximos capítulos explicitarão como o trabalho foi planejado a partir de seus requisitos e das tecnologias envolvidas.

O projeto consistiu em integrar a porta OBD-II de qualquer carro com a plataforma de armazenamento de dados que foi criada.

Para isso, o Capítulo 2 faz uma breve apresentação dos conceitos usados neste projeto. O Capítulo 3, por sua vez, traz as etapas de desenvolvimento do trabalho e o Capítulo 4 mostra quais requisitos foram definidos para o sistema. Além disso, o desenvolvimento inicial do trabalho é descrito no Capítulo 5. Por último, o Capítulo 6 traz as considerações finais com uma breve discussão das conclusões do trabalho, assim como contribuições e sugestões para a continuidade dele.

CAPÍTULO 2

ASPECTOS CONCEITUAIS

A seguir serão explicados os conceitos fundamentais que possibilitam a execução deste trabalho.

2.1 O padrão OBD-II

Criado na década de 1990 para gerar controle sobre as emissões de gás carbônico dos carros[5], hoje define um protocolo padronizado para comunicar parâmetros internos do veículo. O padrão OBD-II foi uma extensão do padrão OBD-I, uniformizando esse tipo de conector em casos em geral, começando a ser adotado no Brasil a partir de 2010[4].

A comunicação é feita através de uma conexão física que normalmente pode ser encontrada abaixo do volante do motorista[5], conforme indicado pela figura 2. Nesse padrão de conexão e comunicação, é definida uma interface por onde parâmetros internos de um veículo podem ser monitorados.

As mensagens definidas pelo OBD-II têm cada uma um PID (*Parameter ID*). O conjunto comum de PIDs de serviços que podem ser solicitados pela porta OBD-II e para que servem pode ser visto na tabela 1[3].

Tabela 1: Serviços OBD-II e suas descrições.

Service / Mode (hex)	Description
01	Show current data - I/M Monitors and Live Data
02	Show Freeze Frame (FF) Data
03	Show Stored Diagnostic Trouble Codes
04	Clear/Erase Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system (EVAP)
09	Request Vehicle Information (VIN)
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

É importante notar que os serviços do padrão OBD-II comuns a todos os carros sempre começam com o dígito zero (hexadecimal), por isso, serviços específicos de

cada fabricante precisam começam a partir do código 0x10.

Figura 2: Posição da porta OBD-II em um carro[5]

Os dados, que podem ser coletados de qualquer carro, são explicitados na tabela 2[3].

Embora não esteja explicitamente representado na tabela, os PIDs 0x00, 0x20, 0x40, etc., ou seja, a cada 32 valores de PID, possuem um parâmetro dedicado exclusivamente para indicar quais, dentre os PIDs subsequentes, são fornecidos pelo veículo em questão. Esses PIDs de marcação, por assim dizer, utilizam 4 *bytes* para comunicar quais dos próximos PIDs estão disponíveis, utilizando cada *bit* como uma *flag* binária.

Na figura 3 é possível visualizar como esse processo é feito, usando como exemplo o valor *0xBE1FA813* para representar os 4 *bytes* oferecidos[3].

Hexadecimal	B				E				1				F				A				8				1				3			
Binary	1	0	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1
Supported?	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	No	No	Yes	No	No	Yes	Yes
PID number	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20

Figura 3: Divisão bit a bit da mensagem OBD-II informando os serviços disponíveis[3]

A coleta de fato dos dados relevantes é feita por uma aplicação desenvolvida *a priori* ao começo do projeto, a qual já era capaz de comunicar-se corretamente com a interface OBD[43].

Essa aplicação acelerou a criação da infraestrutura proposta por este trabalho, uma vez que adiantou o primeiro passo da estratégia *bottom-up* que permeou o projeto.

Tabela 2: PIDs do OBD-II e seus valores de referência.

PIDs(hex)	PID(Dec)	Data bytes returned	Description	Min value	Max value	Units
04	4	1	Calculated engine load	0	100	%
0C	12	2	Engine speed	0	16,383.75	rpm
0D	13	1	Vehicle speed	0	255	km/h
0F	15	1	Intake air temperature	-40	215	℃
1F	31	2	Run time since engine start	0	65,535	seconds
33	51	1	Absolute Barometric Pressure	0	255	kPa
46	70	1	Ambient air temperature	-40	215	℃
47	71	1	Absolute throttle position B	0	100	%
49	73	1	Accelerator pedal position D	0	100	%
51	81	1	Fuel Type	-	-	-
52	82	1	Ethanol fuel %	0	100	%
5A	90	1	Relative accelerator pedal position	0	100	%
5B	91	1	Hybrid battery pack remaining life	0	100	%
5C	92	1	Engine oil temperature	-40	210	℃
5D	93	2	Fuel injection timing	-210.00	301992	°
5E	94	2	Engine fuel rate	0	3212.75	L/h
61	97	1	Driver's demand engine - percent torque	-125	130	%
62	98	1	Actual engine - percent torque	-125	130	%
63	99	2	Engine reference torque	0	65,535	Nm
64	100	5	Engine percent torque data	-125	130	%
70	112	10	Boost pressure control	-	-	-
83	131	9	NOx sensor	-	-	-
8E	142	1	Engine Friction - Percent Torque	-125	130	

2.2 Transmissão de dados

A implementação do *Bluetooth* neste projeto proporciona uma solução eficaz para a comunicação entre a porta OBD do veículo e o *smartphone* do motorista.

A abordagem *wireless* elimina a necessidade de cabos físicos, contribuindo para uma integração mais simplificada do sistema. A eficiência energética do *Bluetooth* assegura uma transmissão de dados sem fio sem prejudicar significativamente o consumo de bateria do aparelho celular.

2.3 Armazenamento em nuvem

O Amazon Web Services[10] (AWS) foi o serviço de nuvem escolhido para o armazenamento dos dados do projeto, destacando-se por suas vantagens em

Tabela 3: Custo projetado da instância da AWS.

Custo inicial	Custo Mensal	Custo Total em 12 Meses
0,00 USD	14,71 USD	176,52 USD

segurança, escalabilidade e praticidade.

A AWS oferece uma infraestrutura robusta com elevados padrões de segurança, garantindo a proteção avançada dos dados, enquanto sua escalabilidade permite uma adaptação eficiente às crescentes demandas do projeto, mantendo a eficiência operacional. Essa escolha proporciona um ambiente de nuvem que atende às exigências do projeto e estabelece uma base sólida para o desenvolvimento futuro.

A base de dados escolhida usa MySQL e foi hospedada em uma instância RDS. Todas as informações coletadas através do aplicativo desenvolvido foram armazenadas nesse serviço.

A combinação do Amazon RDS com a base de dados MySQL oferece uma solução eficiente e escalável para o armazenamento de dados, atendendo às necessidades do projeto e garantindo alta disponibilidade e segurança dos dados.

A instância escolhida para o RDS foi a **db.t3.micro**, pois ela é uma unidade de processamento adequada para cargas de trabalho leves e é comumente usada para ambientes de desenvolvimento, testes ou pequenas aplicações que não exigem muitos recursos.

Foi cogitado fazer uso de um RDS Proxy, que é um serviço que facilita a escalabilidade e a alta disponibilidade para as conexões de banco de dados, mas como este projeto trata-se de um protótipo, não houve preocupações quanto a grandes cargas de trabalho e essa decisão foi descartada.

Também não há necessidade de haver um ambiente de zona de disponibilidade múltipla para garantir escalabilidade, pois haveria custo adicional para o MVP. Por isso, a instância RDS foi configurada para o modo Single-AZ. E a capacidade de armazenamento foi limitada a 20 GiB.

Tabela 4: Parâmetros de escolha da instância da AWS.

Especificação das instâncias do MySQL	
Região	US East(N.Virginia)
Quantidade de instâncias MySQL	1
Tipo da instância	db.t3.micro
vCPU	2
Memória	1GiB

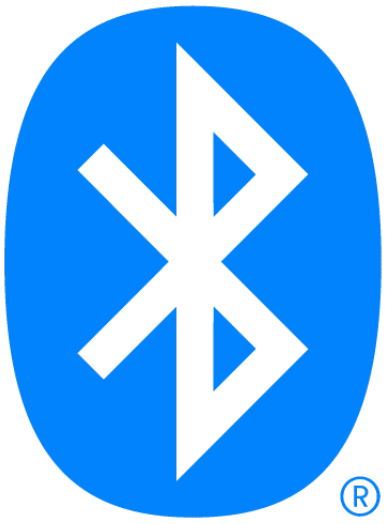

Figura 4: Logo do Bluetooth.

CAPÍTULO 3

METODOLOGIA DO TRABALHO

Esta seção trata das fases do trabalho: como foram executadas e em que sequência.

3.1 Coleta de dados via celular

Para coletar dados de aceleração, é possível utilizar os sensores embutidos nos *smartphones*. Através de aplicativos móveis desenvolvidos para este fim, pode-se interagir com os acelerômetros do dispositivo. Esses sensores registram a aceleração em relação a eixos tridimensionais e ortogonais definidos para dispositivos Android, como pode ser visto na figura 5.

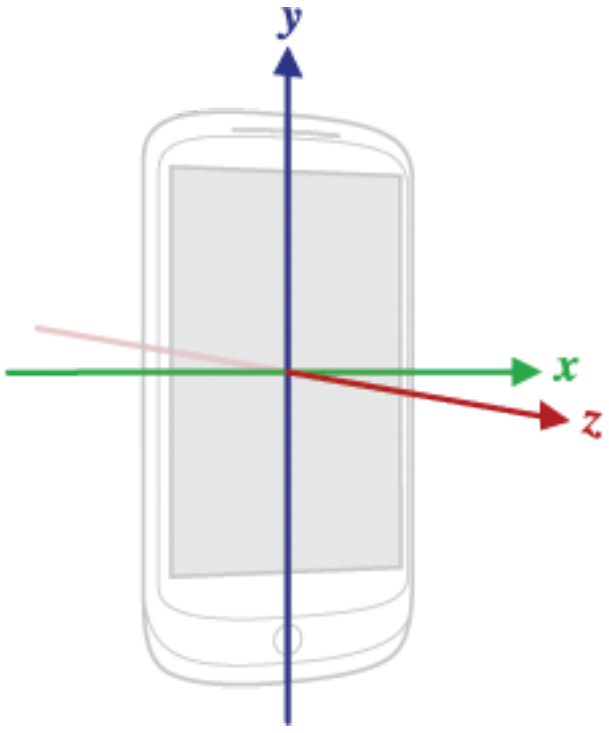

Figura 5: Sistema de coordenadas que rege os valores dos acelerômetros de um dispositivo Android[27].

Com o aplicativo desenvolvido, é possível obter informações detalhadas sobre a

aceleração do veículo em que o *smartphone* está instalado. A taxa de amostragem desses dados pode ser ajustada para atender às necessidades específicas do sistema por um parâmetro chamado *minMillisBetweenData*, conforme foi definido no repositório do projeto Android no GitHub *GitHub* `https://github.com/APF2000/android-obd-reader-pires`.

Os dados coletados são armazenados e processados para análises posteriores. Os códigos das APIs desenvolvidas para receber as requisições do *app* podem ser encontrados nos Apêndices 7.3 e 7.4 deste documento.

A figura 6 mostra os gráficos da aceleração a partir dos dados coletados do celular.

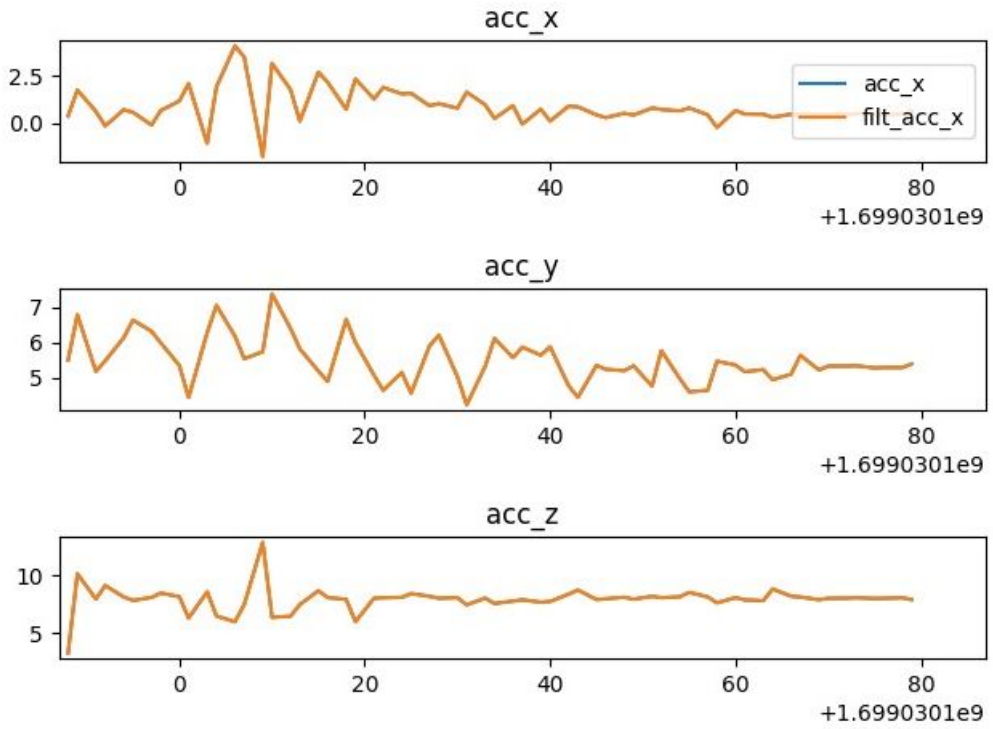

Figura 6: Gráfico da aceleração nos eixos x, y, z.

3.2 Transferência de dados para a nuvem e estruturação

A transferência de dados para a nuvem e a sua estruturação desempenham um papel crucial na eficiência e escalabilidade do sistema de rastreamento de veículos.

Ao adotar-se uma abordagem baseada na nuvem, os dados coletados podem ser transmitidos de maneira eficiente para servidores remotos, centralizando os

dados. A estruturação desses dados na nuvem envolve a organização relacional das informações em bancos de dados, possibilitando a recuperação e análise posterior delas.

O uso do RDS na nuvem oferece uma solução escalável e resiliente, garantindo que o sistema possa crescer de maneira sustentável ao longo do tempo.

3.3 Geração dos dados

O *app* desenvolvido foi utilizado para gerar dados para o projeto. Isso foi feito apenas pelos próprios integrantes do grupo, uma vez que a demora no desenvolvimento ao longo do trabalho impossibilitou que houvesse grande diversidade entre as informações coletadas.

A princípio, foi decidido que arquivos *.txt* conteriam as informações coletadas, para acelerar o desenvolvimento. Os documentos gerados nessa fase do projeto podem ser encontrados em uma pasta do GoogleDrive criada para o projeto: `https://drive.google.com/drive/folders/1Ho0uEHH8MIrNq_SV59YFZzFHzZr1qgSq?usp=sharing`.

Esses arquivos ainda são salvos durante a execução do *app* final, pois é uma forma de *backup* em caso de falha na *internet*, momento durante o qual nenhum dado poderia ser armazenado na nuvem.

A figura 7 ilustra uma das viagens feita para provar o funcionamento do sistema.

Figura 7: Rota feita por um dos integrantes do grupo ao redor do seu bairro.

3.4 Visualização dos dados

A análise de dados no contexto deste projeto pode ser eficientemente conduzida utilizando um Jupyter Notebook (exetnsão *.ipynb*). Ao importar os conjuntos de informações coletadas pelo sistema para esse tipo de arquivo, é possível explorar, visualizar e interpretar os dados de maneira flexível, como pode ser visto no repositório onde a análise dos dados foi feita: `https://github.com/APF2000/data-analysis`.

Os Notebooks gerados foram, no entanto, apenas rascunhos para o que viria a seguir no projeto: o desenvolvimento do arquivo *ride_parser.py*, que é o responsável por transformar em gráficos e tabelas as informações armazenadas no banco de dados.

A API que gera o relatório final sobre o motorista em questão faz uso desse módulo e seu código fonte encontra-se no Apêndice 7.4.

3.5 Análise dos dados

Conforme o artigo citado anteriormente, é possível usar métricas de movimentação brusca do carro para classificar motoristas segundo a segurança de sua direção[15].

Mais especificamente, o estudo determina quatro tipos de condutores: muito cautelosos, cautelosos, agressivos e muito agressivos[15].

A classificação depende basicamente de quantos movimentos bruscos são feitos e em que situações eles ocorrem. Isso significa que em um ponto de maior convergência de estradas, por exemplo, haverá maior penalidade para o *driver safety index* que em um local de menos movimento ou.

Dessa forma, esse estudo procura ranquear melhor os motoristas que têm menos probabilidade de gerar acidentes.

Inspirado nesse outro trabalho, o grupo focou em gerar métricas qualitativas para perfilar motoristas.

A análise dos dados foi feita através de um processo cuidadoso de tratamento e integração de informações provenientes de diferentes fontes, incluindo os dados do sistema de diagnóstico a bordo (OBD), os dados de aceleração e os dados de GPS

(provindos do celular). Inicialmente, esses conjuntos de dados foram processados individualmente para serem salvos em suas próprias tabelas (*DataFrames* na terminologia da biblioteca *pandas* do Python).

Posteriormente, geraram-se metadados de direção apontados por alguns estudos como sendo os mais relevantes para explicar incidentes com um carro (sejam assaltos ou eventos de freada brusca), como era o objetivo inicial do trabalho.

Uma aplicação adicional que surgiu durante o desenvolvimento do projeto foi a validação das informações do OBD com base nos dados do GPS. Essa abordagem permitiu a calibração e a verificação da precisão dos dados do OBD, em particular no que diz respeito à velocidade do veículo.

Utilizando-se as mudanças na latitude e longitude ao longo do tempo, foi possível calcular indiretamente a velocidade do veículo. Isso proporcionou uma maneira de corroborar ou ajustar a velocidade fornecida pelo OBD, garantindo maior confiabilidade nos resultados obtidos e abrindo margem para futuras redundâncias no sistema.

Contudo, é importante destacar uma ressalva em relação aos acelerômetros utilizados. Observou-se que a componente de aceleração na direção do movimento do veículo nem sempre correspondia adequadamente à variação da velocidade obtida pelo GPS. Essa discrepância levantou questões sobre a precisão dos acelerômetros ou sobre possíveis interferências externas que poderiam afetar a medição da aceleração.

3.6 Simulador de OBD

O uso de um simulador de OBD, da marca Freematics, foi necessário durante o desenvolvimento do projeto, uma vez que a disponibilidade de carros para teste esteve quase sempre escassa.

O aparelho usado emula as funcionalidades de um veículo, o que permitiu que o grupo pudesse testar o sistema desenvolvido sem a necessidade de um carro real.

Ademais dessa necessidade inicial do projeto, o simulador é extremamente importante, pois, ao emular dados típicos como velocidade, rotações por minuto (RPM), temperatura do motor, e outros parâmetros do OBD, cria-se um ambiente controlado para verificar a eficácia do sistema em diferentes cenários de condução.

Isso possibilita testes abrangentes, desde a coleta de dados até a análise de comportamentos específicos, garantindo a robustez e confiabilidade do sistema.

Além disso, o uso do simulador de OBD permite a realização de testes de maneira repetitiva e consistente, fornecendo dados padronizados para avaliação de desempenho e detecção de possíveis problemas antes da implementação em um ambiente de produção.

CAPÍTULO 4

ESPECIFICAÇÃO DE REQUISITOS DO TRABALHO

Neste capítulo serão descritas as necessidades básicas levantadas para que o projeto funcione como esperado.

Na figura 8 pode ser visto o diagrama de classes que define o sistema como um todo.

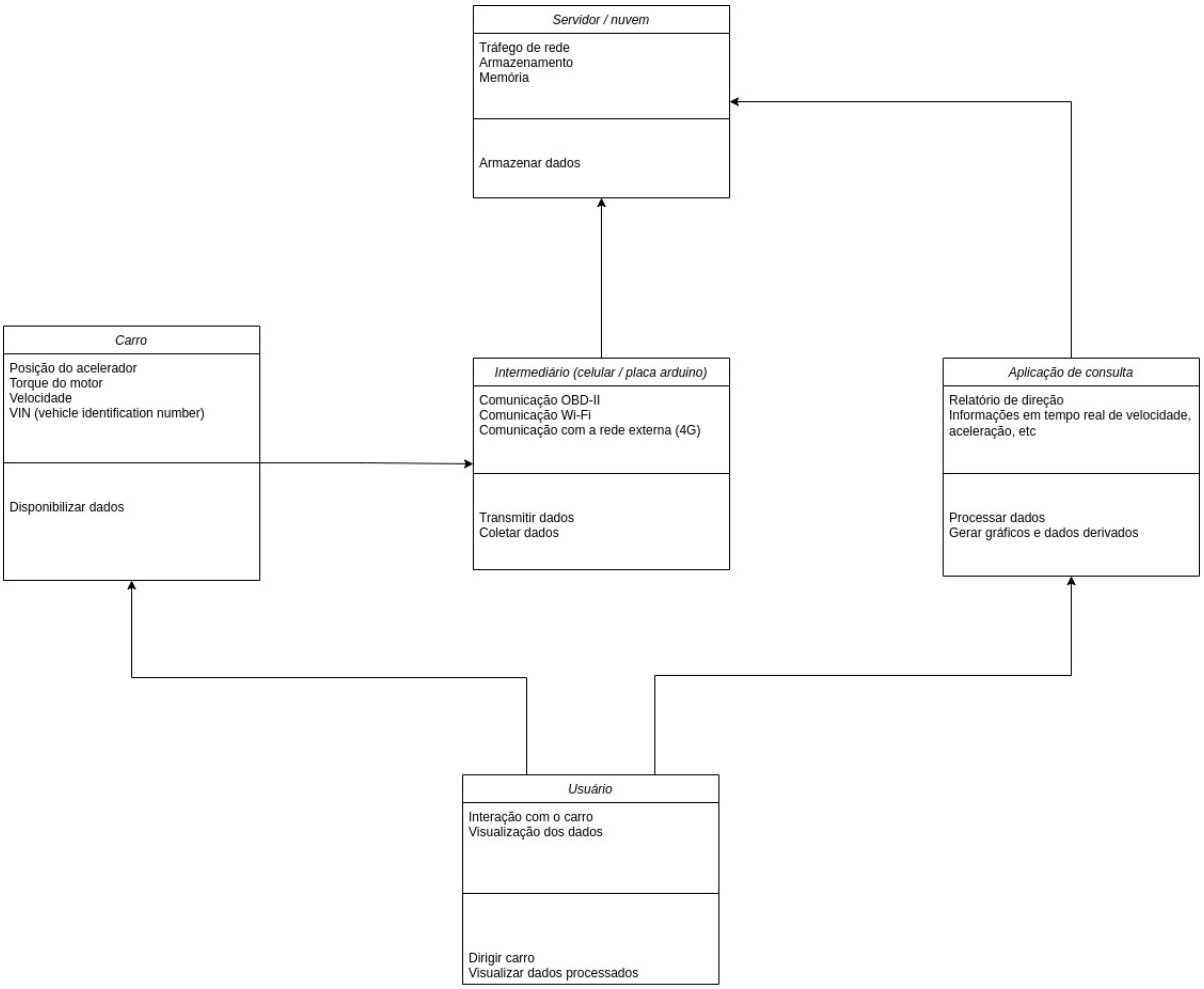

Figura 8: Diagrama de classes com visão mais abstrata do sistema

4.1 Requisitos funcionais

- **Comunicação OBD-II:** o aplicativo faz comunicação com a interface do carro, requisitando apenas as informações definidas, durante o projeto, como relevantes.
- **Conexão do celular à *internet*:** usando-se a tecnologia 4G, deve ser responsável por mandar os dados coletados para a nuvem.
- **Plataforma de recepção na nuvem:** os dados brutos coletados do carro e do celular (GPS e acelerômetro), assim como as informações depois já processadas, serão todos armazenados na plataforma de *cloud* definida.
- **Histórico de Rotas:** o sistema deve ter a capacidade de armazenar e acessar informações detalhadas sobre os trajetos percorridos por um veículo ao longo do tempo. Este recurso é crucial para oferecer aos usuários e administradores uma visão retrospectiva das atividades de um veículo, permitindo uma compreensão abrangente dos padrões de movimentação e comportamento de condução.

4.2 Requisitos não-funcionais

- **Conexão contínua com a *internet*:** caso a conexão pare por muito tempo, a análise dos dados coletados pode ser prejudicada, além de gerar desconfiança por parte do usuário, ao perceber que não há consistência no sistema.
- **Interface sem reconexão manual com a *internet* e *bluetooth*:** se o sistema desligar, a conexão com o OBD e com a *internet* deve ser re-estabelecida sem interferência do usuário assim que for ligada novamente; em outras palavras, o sistema deve ser o mais *plug and play* possível.
- **Informações relevantes e corretas:** os dados apresentados no relatório de direção precisam ter sentido para o usuário; meta-informações são as únicas que devem chegar até o usuário final, e também precisam passar por algum tipo de "filtro de plausibilidade", para que, mais uma vez, a confiança de quem usará o sistema não seja perdida ao deparar-se com valores *outliers* nas análises geradas.

- **Memória e armazenamento em nuvem suficientes:** o espaço máximo em nuvem que pode ser usado deve ficar claro para o usuário quando ele usar o sistema; o motorista precisa ficar ciente de que suas informações não serão mais coletadas assim que o armazenamento seja ocupado por completo.
- **Segurança de dados:** pessoas não-autorizadas não pode ter acesso aos dados coletados no veículo de cada usuário.

CAPÍTULO 5

DESENVOLVIMENTO DO TRABALHO

Este capítulo tem como objetivo explicitar quais foram as decisões de projeto que foram tomadas ao longo do desenvolvimento do sistema.

5.1 Justificativas das métricas escolhidas

A escolha das métricas para o desenvolvimento desse projeto foi determinante para avaliar e interpretar resultados com precisão e, como era o objetivo do trabalho, avaliar o perfil de motoristas.

Uma conversa com um funcionário da empresa Porto Seguro foi determinante para ratificar a relevância das métricas definidas. Essa conversa pode ser vista no Apêndice 7.1 deste documento.

5.1.1 Gasto do motor a partir do RPM

Em um motor de combustão interna, o RPM indica a velocidade com que os pistões se movem para cima e para baixo no cilindro. O controle preciso do RPM é essencial para otimizar a eficiência do motor e a entrega de potência, sendo um indicador chave para os motoristas ajustarem suas velocidades de condução.

A figura 9 mostra que a potência gasta por um motor aumenta quando a rotação dele também ascende[28].

Dado que a amplitude de tais ondulações (alto RPM) reduz ao longo da vida útil de um pistão, deve-se levar em consideração o efeito do desgaste ao especificar a rugosidade superficial da saia do pistão. Caso contrário, o pistão poderá vir a engripar no cilindro por não ser capaz de reter óleo em sua superfície [40].

Durante o funcionamento do motor, a biela fica sujeita a forças de compressão muito elevadas, provenientes da fase de expansão do cilindro e a forças de tração,

nas fases de admissão do motor. Sendo assim, as bielas são mais solicitadas nas condições de plena carga e de elevadas rotações do motor[40].

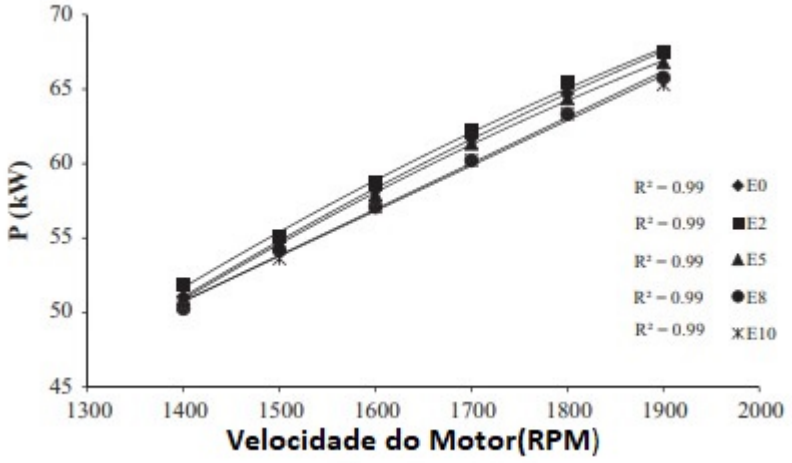

Figura 9: Potência gasta pelo motor em função de sua rotação.

5.1.2 Trajetos e rotas perigosas

A relação entre rotas perigosas, marcadas por assaltos e roubos de carros e o perfil do motorista torna-se um elemento crítico na gestão da segurança. A escolha da rota desempenha um papel significativo na mitigação desses perigos.

Motoristas que optam por passar frequentemente por áreas consideradas de alto risco podem estar mais suscetíveis a incidentes indesejados. Portanto, compreender o perfil do motorista em relação a essas rotas perigosas é importante para a implementação de estratégias eficazes de prevenção.

A conscientização dos motoristas sobre áreas de risco pode contribuir para um comportamento mais prudente ao planejar e seguir rotas.

Na cidade de São Paulo, locais como Paulista, Augusta e Sé possuem altos índices de furtos[34]. A figura 10 mostra as áreas com maior índice de assaltos na metrópole.

A figura 11 ilustra alguns dos metadados que podem ser gerados com a análise dos dados obtidos. De forma semelhante, as tabelas 5 e 6 detalham quanto por cento do tempo o motorista passa acelerando o carro e também a fração de tempo passada em cada região da cidade, respectivamente. Ambas as métricas são subdivididas em três categorias: baixo, médio e alto risco.

A construção dessa figura foi feita graças a uma base de dados encontrada no site

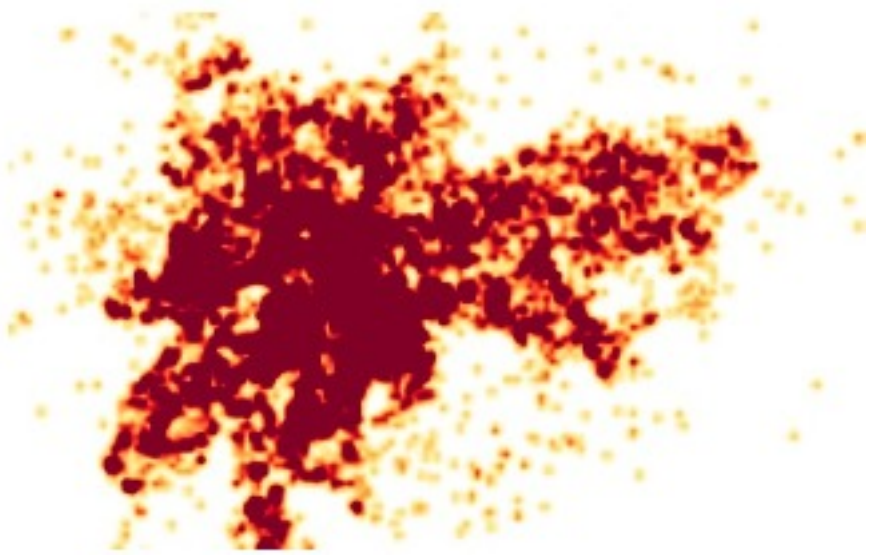

Figura 10: Mapa de calor dos incidentes criminais na cidade de São Paulo. Com atenção, é possível ver o contorno da cidade nesta imagem.

Tabela 5: Exemplo de análise qualitativa da aceleração do carro durante a coleta de dados.

	Fração do tempo com cada tipo de aceleração
Aceleração normal	0,00%
Aceleração média	55,30%
Aceleração muito alta	44,70%

Tabela 6: Exemplo de análise qualitativa de por onde o carro passa durante a coleta de dados.

	Fração do tempo passada em cada tipo de lugar
Baixo risco	54,55%
Médio risco	45,45%
Alto risco	44,70%

Kaggle[33], conhecido por hospedar *Jupyter Notebooks* analisando certos conjuntos de informações.

Embora a descrição do *dataset* mencione que os dados referem-se só a São Paulo (o que já gera ambiguidade entre cidade e estado), existem lá registros de Curitiba, Rio de Janeiro e Pará também.

Importante mencionar que, devido ao tamanho dessa base de dados, de 12899 linhas, foi preciso pensar-se em como comparar os dados nela contidos com cada um dos registros de GPS fornecidos pelo *app* sem que o desempenho dessa análise fosse comprometido.

Para tal, as coordenadas geográficas da planilha foram indexadas conforme indicado no código-fonte presente no Apêndice 7.5.

Alguns números mágicos (constantes sem derivação algorítmica), que foram utilizados nesse código, levaram em consideração a equivalência entre graus e quilômetros da Terra e as distâncias entre os pontos mais a sul, norte, leste e oeste do Brasil[35, 36].

A ideia dessa indexação é dividir o mapa em *chunks*, os quais limitam o trabalho computacional a um subconjunto dos dados. Cada *chunk* é um quadrado de lado 500m, o que obriga o programa a olhar alguns blocos adjacentes ao onde está a coordenada cuja segurança quer-se verificar.

5.2 Tecnologias utilizadas

O projeto definiu algumas tecnologias de base para serem usadas durante o desenvolvimento de cada fase dele.

5.2.1 Amazon Web Services

A escolha da AWS para o projeto é fundamentada em três pilares essenciais: escalabilidade, segurança e viabilidade econômica.

A capacidade desse serviço de nuvem de se adaptar dinamicamente às demandas do sistema assegura uma infraestrutura flexível, capaz de lidar eficientemente com variações na carga de trabalho.

Além disso, a reputação consolidada da Amazon em termos de segurança oferece uma base robusta para proteção dos dados e operações.

Por fim, a viabilidade econômica se destaca, uma vez que a AWS disponibiliza uma variedade de serviços e modelos de precificação que se alinham de maneira eficaz às necessidades do projeto, otimizando custos operacionais.

É evidente que todas essas vantagens só são concretizadas caso os implementadores do sistema façam uso de toda a funcionalidade da nuvem: projetos monolíticos, ainda que rodados em servidores distribuídos, não exploram a flexibilidade de serviços como a AWS.

5.2.2 Conexão OBD-II

A integração do OBD-II (*On-Board Diagnostics*) neste projeto de sistema de coleta de dados representa um ponto importante na obtenção de dados precisos e abrangentes sobre o desempenho do veículo.

O OBD-II, um padrão presente em muitos veículos modernos, fornece acesso a uma variedade de parâmetros, como velocidade, rotações por minuto (RPM), temperatura do motor, e códigos de diagnóstico de falhas.

Ao conectar-se a plataforma de captura de dados ao leitor de OBD-II do veículo, é possível extrair informações em tempo real sobre a condução, condições do motor e possíveis problemas mecânicos.

Uma plataforma disponibilizada no GitHub que consegue comunicar-se com o veículo através do protocolo OBD-II[12] acelerou o desenvolvimento do projeto, uma vez que o grupo não teve que implementar esse protocolo.

Esse outro projeto implementa o protocolo OBD-II e também uma interface gráfica básica para mostrar os dados fornecidos pelo carro.

A figura 12 mostra a tela principal do *app*, onde é possível ver alguns dos dados gerados durante o dirigir do usuário.

5.2.3 Simulador de OBD-II

A ausência de um carro físico no início do projeto foi contornada por meio da utilização de um simulador de OBD. Esse dispositivo desempenhou um papel crucial

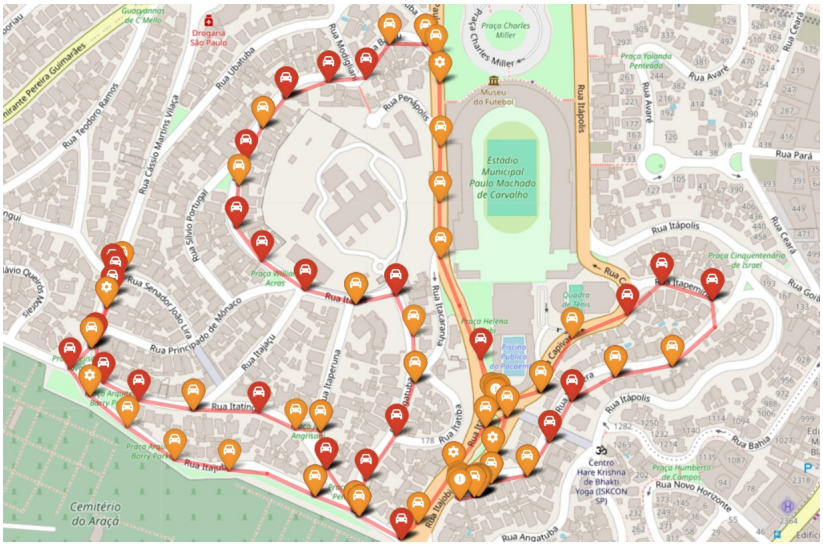

Figura 11: Mapa de risco em acelerações (símbolo de carro) e valores de RPM (símbolo de engrenagem).

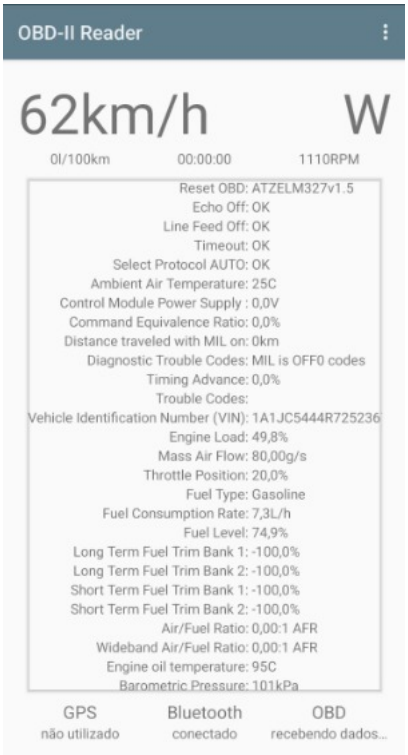

Figura 12: Interface inicial do *app* que se comunica com a porta OBD-II[12].

como um substituto temporário do automóvel, permitindo que o trabalho continuasse normalmente.

Em vez de depender de um veículo real, a equipe conseguiu controlar e reproduzir dados e comportamentos típicos de um carro a partir do programa de controle do simulador[21].

Além disso, o uso desse dispositivo proporcionou um ambiente controlado para a validação do *software* em diferentes cenários, contribuindo para a eficiência e progresso do projeto enquanto não se podia realizar testes práticos.

5.2.4 Android Studio e Java

A utilização da IDE Android Studio neste projeto de sistema de coleta de dados desempenha um papel central no desenvolvimento do aplicativo móvel já mencionado.

O Android Studio, sendo a principal ferramenta de desenvolvimento para aplicativos Android, oferece um ambiente integrado e robusto que simplifica a criação, os testes e a depuração do *software*.

A plataforma fornece recursos avançados de *design* de interface do usuário, facilitando a criação de aplicativos intuitivos e visualmente atraentes para os usuários finais.

A integração perfeita com o Android SDK (*Software Development Kit*) permite o acesso às APIs e recursos específicos do sistema operacional Android, garantindo, através de interfaces do sistema operacional, o uso de *hardwares* de propósito específico, como o que faz a comunicação Bluetooth.

Além disso, as ferramentas de emulação e depuração incorporadas no Android Studio simplificam o processo de teste em diferentes dispositivos, contribuindo para a criação de aplicativos estáveis e adaptáveis.

Nenhuma tecnologia multiplataforma (React Native ou Flutter, por exemplo) ou específica de dispositivos iOS (Swift ou Objective C, por exemplo) foi escolhida, pois o intuito do projeto era ser compatível com os celulares dos membros do grupo e devido à familiaridade com a linguagem Java, usada para esse tipo de desenvolvimento.

O aplicativo em si originalmente foi desenvolvido para a API de versão 22 do Android. Condição que foi alterada para comportar funcionalidades que foram

adicionadas ao sistema operacional desde 2013. A versão atual de compilação (versão do SDK) é a 33.

O desenvolvimento do aplicativo Android envolveu várias etapas, sendo o primeiro passo a modificação apropriada do arquivo *build.gradle*. Este processo, no entanto, consumiu considerável tempo devido à falta de experiência dos membros do grupo.

Após a configuração inicial, a equipe se dedicou a identificar onde os parâmetros do OBD estavam disponíveis no código-fonte. Esta fase de investigação foi crucial para possibilitar o subsequente envio desses dados para a nuvem.

No entanto, as dificuldades técnicas foram tão significativas que inicialmente não foi possível realizar o envio de dados para a nuvem.

Diante desse impasse, uma exceção teve que ser aberta no projeto, resultando na geração de arquivos *.txt* locais no *smartphone* em que o aplicativo estivesse rodando.

Essa solução permitiu que o trabalho de análise dos dados começasse, já que a produção de informações já estava funcional naquele momento.

Posteriormente, a equipe conseguiu superar os obstáculos e implementou com sucesso o envio de dados para a nuvem, empregando a biblioteca Volley do Android. Essa abordagem proporcionou uma resolução eficaz para a integração com serviços de armazenamento na nuvem, consolidando a funcionalidade do *app*.

Figura 13: Logo do Android Studio.

5.2.5 Banco de dados

A integração do MySQL e do AWS RDS (Relational Database Service) em um projeto de sistema de compilação de dados de veículos é uma estratégia robusta para o gerenciamento eficiente e escalável dos dados do sistema.

O MySQL, um sistema de gerenciamento de banco de dados relacional de código aberto, proporciona uma estrutura confiável para armazenar e organizar informações como histórico de rotas e dados do OBD-II.

Ao escolher o AWS RDS como a plataforma de hospedagem para o MySQL,

ganha-se os benefícios adicionais de escalabilidade automática, alta disponibilidade e segurança avançada oferecidos pela infraestrutura em nuvem da Amazon. A integração dessas tecnologias permite o acesso eficiente aos dados, consultas rápidas e uma gestão simplificada do banco de dados.

Além disso, o AWS RDS lida com tarefas operacionais, como *backup* automático e manutenção.

Essa combinação proporciona uma base sólida para o armazenamento e recuperação de dados, promovendo a confiabilidade e eficácia do sistema.

A estrutura definida para as tabelas do banco de dados foi:

- **tcc-schema** (esquema do banco de dados onde se encontram as tabelas)
  - **info:** Armazena as informações coletadas através da porta OBD
  - **acceleration:** Contém dados dos acelerômetros do celular do usuário
  - **location:** Guarda latitude e longitude obtidas por GPS pelo celular

A especificação exata das colunas de cada tabela pode ser encontrada no Apêndice 7.2.

Figura 14: Logo do Mysql.

Importante mencionar que o tamanho do banco de dados foi limitado a 20GiB, visando impedir que o serviço da AWS cobrasse qualquer valor monetário dos membros do grupo.

O *Free Tier* é um conjunto de limites de execução e armazenamento em nuvem definido para projetos-teste de contas novas na plataforma (com menos de 1 ano de existência[40]). A forma de zerar os gastos com esse projeto é não exceder a fronteira dessas cortesias.

5.2.6 *Python*

A versão do Python usada para todos os arquivos *.py* do projeto foi a 3.12.0.

É importante mencionar que junto a cada módulo feito em Python neste projeto, as dependências necessárias para rodá-los foram escritas em arquivos *requirements.txt*, como pode ser visto nos repositórios outrora mencionados.

Para poder-se executar os *Jupyter Notebooks* desenvolvidos, é preciso desencadear os seguintes comandos em um terminal (em sistema operacional *Linux*):

```
python3 -m venv .venv
source .venv/bin/activate
python3 -m pip install requirements.txt
```

5.3 Projeto e implementação

5.3.1 Análise de Dados - RideParser

A função da classe RideParser é analisar dados dos trajetos feitos pelos motoristas.

Alguns filtros são recebidos pelo construtor dessa classe, para personalizar a geração de gráficos. Esses parâmetros são usados para restringir o que vem do banco de dados a uma certa janela de tempo e ao usuário que solicita aquela informação.

Dentro desse mesmo arquivo, algumas bibliotecas típicas de ciência de dados foram utilizadas para manipular os gráficos mencionados. Foram elas: pandas, matplotlib, numpy e scipy.

A figura 15 mostra o video e o gráfico da aceleração de um carro. Os dados foram pegos de um repositório chamado UAH-Driveset[42], disponível na *internet*.

5.3.2 API Rest - Flask

A implementação da API centraliza-se em uma arquitetura Python utilizando a biblioteca Flask, proporcionando uma estrutura ágil e eficiente para a comunicação com a base de dados MySQL, hospedada em um serviço de RDS na AWS. A escolha da linguagem Python e do framework Flask foi motivada pela sua simplicidade

e flexibilidade, permitindo a rápida construção dos *endpoints* responsáveis pela interação entre o aplicativo e a base de dados.

A lógica da API foi encapsulada em funções específicas, acionadas por meio de requisições POST. O corpo do JSON enviado nessas requisições determina a ação a ser executada, que pode ser uma entre seis operações distintas. Três destas operações referem-se a consultas (GET) e três a modificações (POST), cada uma vinculada a uma das três tabelas da base de dados.

Para facilitar a utilização da API pelo *app*, o código Python foi integrado ao ambiente *serverless* da AWS Lambda.

Essa escolha estratégica permite que as funções sejam invocadas sob demanda, proporcionando uma resposta ágil às requisições do aplicativo Android, que, por sua vez, está sendo desenvolvido no ambiente Java do Android Studio.

A combinação dessas tecnologias promove uma arquitetura eficiente para gerenciar a comunicação entre o aplicativo e a infraestrutura de banco de dados na nuvem.

5.3.3 Folium

Para criar os mapas do aplicativo, foi utilizado o *framework* Folium. Essa biblioteca é uma poderosa ferramenta para manipular e visualizar dados geoespaciais usando Python.

Com o Folium, é possível criar mapas interativos personalizados e incorporar dados neles de várias maneiras. Para manipular dados usando essa ferramenta, pode-se começar importando a biblioteca e criando um objeto *Map*, que representa o mapa.

Em seguida, pode-se adicionar camadas, como marcadores, polígonos e *popups*, para exibir dados de forma intuitiva no mapa.

A figura 17 mostra as bibliotecas utilizadas para desenvolver os mapas do projeto.

Para traçar uma trajetória no mapa usando a biblioteca Folium em Python, é necessária uma lista de pontos de latitude e longitude que representam a trajetória.

A partir dos pontos colhidos pelo GPS do celular, cria-se um mapa da trajetória descrita. A demonstração de uma trajetória recriada em um mapa usando Folium

pode ser vista na figura 18.

O grupo procurou correlacionar eventos de anormalidade nos dados de aceleração do UAH Drivest observando o comportamento da curva dessa grandeza física no tempo e comparando com os eventos do vídeo que acompanhava cada uma das pastas disponibilizadas, mas nenhuma funcionalidade foi adicionada à análise dos dados devido a essa escolha.

5.3.4 Criação de relatório de direção em PDF

A geração de *insights* visuais e análise métrica a partir dos dados armazenados na base MySQL é um componente essencial do projeto.

A decisão de fazer isso através da geração de um PDF para o motorista baseou-se na clareza e simplicidade desse formato de arquivo.

Essa escolha é respaldada também pela facilidade de implementação, disponibilizada pelas bibliotecas *Matplotlib* e *Reportlab*, que permitem a criação de PDFs formatados contendo os gráficos com as métricas geradas.

Um segundo AWS Lambda desempenha papel central nesse processo, funcionando como uma ponte entre a base de dados e as ferramentas de visualização.

Ao ser acionado, o Lambda gerador de PDF realiza uma chamada ao outro Lambda, que extrai as informações relevantes da base de dados MySQL referentes ao usuário, para realizar a análise.

Utilizando a biblioteca *Matplotlib* em Python, a função Lambda então emprega métodos da classe RideParser, mencionada na seção 5.3.1, para criar um relatório do motorista, com gráficos e métricas relevantes referentes a sua condução. Isso proporciona aos usuários uma compreensão mais aprofundada das métricas associadas às suas atividades.

Para proporcionar aos usuários uma maneira acessível de interagir com esses *insights*, um botão "gerar PDF"foi incorporado no aplicativo. Este botão aciona o segundo Lambda, que compila os gráficos e métricas gerados em um arquivo PDF.

A utilização da biblioteca *smtplib* facilita o envio desse PDF diretamente para o usuário via *email*. Essa solução integrada oferece uma experiência completa aos usuários, permitindo-lhes não apenas visualizar, mas também compartilhar os

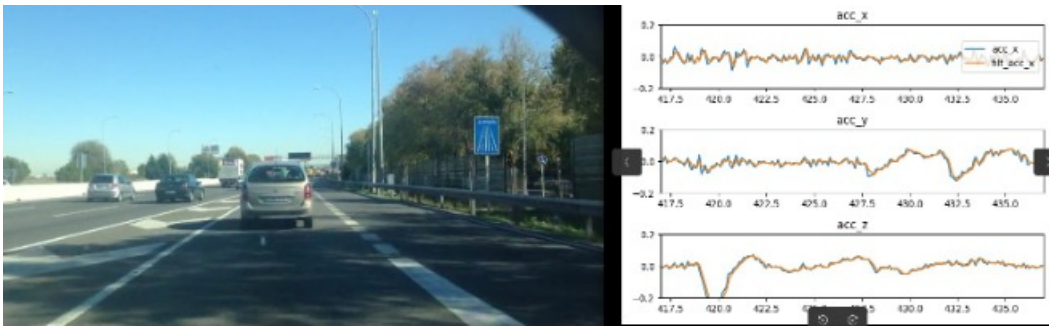

Figura 15: Rota de uma viagem de um dos integrantes do grupo.

Figura 16: Logos do Python e do Folium.

```
import folium
import pandas as pd
import os
from ride_parser import RideParser
import matplotlib
```

Figura 17: Bibliotecas utilizadas para gerar o mapa da rota do usuário.

Figura 18: Rota de uma das viagens fornecidas pelo *UAH Driveset*.

resultados de suas análises de maneira simples.

5.3.5 Autenticação com *Firebase*

A escolha do *Firebase* para a identificação única do usuário no sistema decorre da capacidade da plataforma em oferecer eficiência e segurança nesse processo, diminuindo-se a carga de responsabilidade do sistema em armazenar informações sensíveis de cada pessoa.

Essa decisão é fundamentada pela praticidade proporcionada pela integração do *Firebase* com aplicativos *Android* e páginas web[29], simplificando a implementação de autenticação e gestão de usuários, assegurando uma identificação singular e segura no âmbito do sistema em questão.

5.3.6 Dependências do Gradlle

Em um primeiro instante o *app* android-obd-reader não rodava na IDE Android studio.

Para que fosse possível compilar o projeto, alterações no arquivo **gradle/wrapper/gradle-wrapper.properties** foram necessárias.

Atualizar as dependências do Gradle é muitas vezes pré-condição para garantir que um aplicativo possa se beneficiar das correções de *bugs*, melhorias de desempenho e novos recursos fornecidos pelas versões mais recentes das bibliotecas que estão sendo utilizadas. Este ponto na história do repositório mostra as alterações que foram necessárias: `https://github.com/APF2000/android-obd-reader-pires/commit/def70ed262b1fad28cc9e2b98a48af45cfac5695`.

5.3.7 Proteção dos dados

Na Lei Geral de Proteção de Dados (Lei n. 13.709/18-LGPD), parte-se da ideia de que todo dado pessoal tem importância e valor.

Por essa razão, este projeto levou em consideração o conceito amplo de dado pessoal, assim como estabelecido no Regulamento Europeu [30] (*GDPR-General Data Protection Regulation*).

Quando se trata de um aplicativo projetado para analisar o perfil do motorista de carros, a LGPD impõe uma série de responsabilidades e requisitos ao desenvolvedor e operador do aplicativo.

É necessário obter o consentimento do usuário antes de coletar e processar quaisquer dados pessoais relacionados ao seu perfil de condução.

Além disso, a LGPD exige que medidas de segurança apropriadas sejam implementadas para proteger esses dados contra acessos não autorizados e vazamentos. Os usuários têm o direito de acessar, corrigir e excluir suas informações pessoais, e o aplicativo deve fornecer meios para que esses direitos sejam exercidos.

Por simplicidade, este trabalho não cumpriu essas regras, mas o grupo tem ciência das vulnerabilidades e dos pontos de melhoria nesse aspecto.

5.4 Testes e avaliação

Alguns testes foram definidos para a ratificação das funcionalidades do sistema:

- **Conferir conexão OBD-II:**
  - Conectar o dispositivo de leitura na entrada do carro
  - Ligar a comunicação *bluetooth* do celular
  - Parear os dois dispositivos
  - Digitar a senha 1234, padrão dos dispositivos de leitura OBD
  - Ligar a coleta de dados através do aplicativo
  - A tela deve sinalizar que o protocolo está sendo feito da forma correta
- **Teste de continuidade de transferência de dados:**
  - Fazer conexão com o OBD conforme descrito no teste anterior
  - Andar com o carro por alguns minutos com a tela do celular desbloqueada (de preferência mudar o *timeout* da tela nas configurações do celular)
  - Atestar que nenhum dado foi perdido por falta de conexão (pode ser visto por descontinuidade dos gráficos)
- **Equivalência de envio e recepção de dados:**

  - Comparar o *log* de envio de dados do aplicativo de interface com a porta OBD-II com o *log* de salvamento do banco de dados hospedado na nuvem
  - A quantidade de dados, os *timestamps* deles e seus conteúdos devem ser idênticos

- **Testes de segurança:**
  - Ratificar que não é possível ter acesso aos dados de um certo usuário sem ter as informações de *login*
  - Verificar o certificado SSL do remetente e só aceitar a mensagem se a informação condisser com as credenciais armazenadas daquele usuário

- **Geração de dados *outliers*:**
  - Verificar se o sistema descarta corretamente as informações que fogem totalmente do padrão esperado ou se consegue pelo menos esconder isso do usuário
  - Os PDFs gerados com o sistema devem sofrer um processo de análise de plausibilidade, o que deve averiguar que a análise automática está sendo feita corretamente

5.4.1 Fluxo de uso do sistema

- **Autenticação com o Google:** O aplicativo no celular deve ser iniciado para realizar a autenticação com as credenciais do Google, assegurando uma identificação segura do usuário.
- **Conexão com o OBD:** A conexão entre o celular e o OBD deve ser estabelecida, garantindo que o leitor esteja corretamente emparelhado para iniciar a comunicação.
- **Início da Coleta de Dados:** A coleta de dados deve ser ativada por meio do aplicativo.
- **Condução do Veículo:** O veículo deve ser conduzido normalmente, permitindo que o aplicativo faça a coleta de dados em tempo real.
- **Posicionamento Fixo do Celular:** O celular deve ser fixado em uma posição estável dentro do veículo usando um suporte apropriado. Isso garantirá que

os dados de aceleração sejam confiáveis, o que evitará qualquer tipo de interferência na análise posterior da direção.

5.4.2 Carros utilizados para teste

A escolha dos carros onde o leitor de OBD seria conectado foi feita a partir dos veículos de familiares.

Os modelos em que o sistema foi testado, foram em maioria para analisar quais parâmetros de OBD haviam sido implementados em todos eles.

Os carros usados foram:

- Hyundai Ix35 - 2012
- Mercedes C200 CGI - 2014
- Ford Xsport - 2008

O modelo Ix35 foi a principal fonte de dados do projeto, pois praticamente todos os testes feitos para a nova base de dados foram feitos com ele.

5.4.3 Aparelhos *smartphone*

O aplicativo foi rodado em dois modelos diferentes de celular:

- Samsung Galaxy A71
- Samsung Galaxy S9
- Samsung Galaxy S22
- Motorola Moto G

5.4.4 Definição de dados relevantes para o perfilamento

Os parâmetros identificados em comum em todos os carros testados foram: *AIR-FUEL-RATIO, AIR-INTAKE-TEMP, AMBIENT-AIR-TEMP, BAROMETRIC-PRESSURE, CONTROL-MODULE-VOLTAGE,*

DISTANCE-TRAVELED-MIL-ON, DTC-NUMBER, ENGINE-COOLANT-TEMP, ENGINE-LOAD, ENGINE-OIL-TEMP, ENGINE-RPM, ENGINE-RUNTIME, EQUIV-RATIO, Echo Off, FUEL-CONSUMPTION-RATE, FUEL-LEVEL, FUEL-PRESSURE, FUEL-RAIL-PRESSURE, FUEL-TYPE, INTAKE-MANIFOLD-PRESSURE, Line Feed Off, Long Term Fuel Trim Bank 1, Long Term Fuel Trim Bank 2, MAF, Reset OBD, SPEED, Select Protocol AUTO, Short Term Fuel Trim Bank 1, Short Term Fuel Trim Bank 2, THROTTLE-POS, TIMING-ADVANCE, TROUBLE-CODES, Timeout, VIN, WIDEBAND-AIR-FUEL-RATIO.

No entanto, muitas dentre essas informações ou apresentavam o valor *NODATA*, que significa que estavam sendo amostrados pelo OBD, mas não continham dados válidos, ou apresentavam dados constantes, os quais independem da direção do motorista.

Os **parâmetros descartados** foram *CONTROL-MODULE-VOLTAGE, DISTANCE-TRAVELED-MIL-ON, DTC-NUMBER, Echo Off, ENGINE-OIL-TEMP, FUEL-CONSUMPTION-RATE, FUEL-LEVEL, FUEL-PRESSURE, FUEL-RAIL-PRESSURE, FUEL-TYPE, Line Feed Off, Long Term Fuel Trim Bank 2, MAF, Reset OBD, Select Protocol AUTO, Short Term Fuel Trim Bank 2, Timeout, TROUBLE-CODES, VIN, WIDEBAND-AIR-FUEL-RATIO*.

Os **parâmetros considerados** para uso foram *AIR-FUEL-RATIO, AIR-INTAKE-TEMP, AMBIENT-AIR-TEMP, BAROMETRIC-PRESSURE, ENGINE-COOLANT-TEMP, ENGINE-LOAD, ENGINE-RPM, ENGINE-RUNTIME, EQUIV-RATIO, INTAKE-MANIFOLD-PRESSURE, Long Term Fuel Trim Bank 1, Short Term Fuel Trim Bank 1, SPEED, THROTTLE-POS, TIMING-ADVANCE*.

Entre os dados candidatos para serem usados no projeto, apenas os que foram **citados em artigos científicos** como determinantes para o acontecimento de acidentes e a deterioração do carro foram de fato levados em consideração: *ENGINE-RPM*[28], *SPEED*[32]

CAPÍTULO 6

CONSIDERAÇÕES FINAIS

Este capítulo descreve as observações posteriores à conclusão do projeto.

6.1 Conclusões do projeto de formatura

Neste projeto foram explorados diversos tópicos relacionados ao desenvolvimento de um sistema de compilação de dados, abrangendo desde os requisitos funcionais até a implementação prática de tecnologias e ferramentas específicas.

Foram discutidos aspectos cruciais, como segurança da informação, integração de *hardware* e análise de dados. A abordagem envolveu desde a coleta de dados por meio de sensores até a visualização interativa em mapas.

A integração de tecnologias como AWS RDS, MySQL, OBD-II, Python e Folium destacou a diversidade de ferramentas disponíveis para a construção de sistemas robustos de captura de dados.

Dessa forma, a concepção e implementação de uma plataforma de captura de dados exigiu uma abordagem multidisciplinar, considerando não apenas a tecnologia, mas também os aspectos éticos e práticos para oferecer uma solução eficaz, segura e alinhada às necessidades do usuário.

6.2 Contribuições

6.2.1 Análise em *chunks* da Base de Dados de Crimes no Brasil

O projeto proporciona uma análise eficiente e segmentada da base de dados de crimes no Brasil. A divisão em *chunks* permite uma análise mais gerenciável e escalonável, possibilitando *insights* mais detalhados sobre padrões criminais em diferentes regiões do mapa.

6.2.2 Geração de PDF com Informações Relevantes sobre o Comportamento do Motorista

O projeto oferece a funcionalidade de gerar um PDF contendo informações sobre o comportamento do motorista.

Esse documento inclui dados essenciais provenientes da coleta de dados do veículo sob um ponto de vista qualitativo (desgaste de motor e nível de perigo corrido), proporcionando uma visão abrangente do estilo de condução do motorista.

6.3 Dificuldades encontradas e lições aprendidas

6.3.1 Salvamento de dados

As dificuldades encontradas ao salvar arquivos no celular Android e na nuvem da AWS podem ser atribuídas à falta de clareza na documentação desses serviços.

A ausência de informações completas e operacionais (códigos-fonte que funcionam) resultou em desafios significativos durante a implementação deste projeto.

Dessa forma, o grupo começou a valorizar a busca constante por códigos chamados *Hello World's*, os quais provam um conceito que deve ser parte de uma solução mais complexa.

O maior aprendizado deste projeto é a definição de passos o mais atômicos possíveis para que o trabalho possa caminhar em solo firme do começo ao fim, isto é, todos os submódulos do sistema devem ter um plano de testes claro para que os engenheiros que lidarão com ele no futuro sejam capazes de solucionar problemas o mais rápido possível.

Tecnologias desconhecidas O grupo chegou à conclusão que a especificação das tecnologias utilizadas para um projeto deve sempre levar em conta o conhecimento e a capacidade da equipe envolvida no desenvolvimento.

Embora sempre haja novos conhecimentos a serem explorados, é importante limitar a quantidade de dúvidas inciais do projeto e saná-las o mais rápido possível.

Essa abordagem não foi adotada da melhor forma neste trabalho, o que gerou muito *stress* para a geração de código, principalmente durante o desenvolvimento do aplicativo *Android*.

6.4 Perspectivas de continuidade

A oferta deste sistema para empresas de seguro pode representar uma proposta de valor significativa. Ao utilizar uma solução que integra tecnologias avançadas de análise de dados, as seguradoras podem aprimorar suas avaliações de risco de maneira personalizada.

O sistema oferece a capacidade de monitorar o comportamento do condutor, avaliar padrões de direção segura e identificar incidentes, permitindo uma avaliação mais precisa e personalizada do risco associado a cada segurado.

Além disso, com as métricas recolhidas, é possível fornecer *feedback* direto aos motoristas ou oferecer descontos e incentivos para práticas de condução seguras. O histórico de rotas e a telemetria abrangente proporcionam transparência e eficiência no processo de precificação de seguro.

Ao posicionar esse sistema como uma ferramenta inovadora para melhorar a gestão de riscos, as empresas de seguro podem não apenas otimizar seus processos, mas também criar um diferencial competitivo valioso, fortalecendo a fidelidade do cliente e promovendo a segurança viária de maneira colaborativa.

CAPÍTULO 7

APÊNDICES

7.1 Conversa com um profissional da Porto Seguro sobre seguros de carro

Pergunta 1: Quanto aos locais por onde o motorista passa ao longo do dia, é mais importante saber o nome do bairro por onde ele passa ou se soubermos a rua é melhor pra sabermos o risco de assalto?

Resposta 1: A precificação do seguro já é baseada em três fatores: onde a pessoa mora, onde ela trabalha e o trajeto feito por ela para ir de um local ao outro. Na nossa plataforma, chamada *Corretor Online*, temos acesso a esses e outro dados do cliente, o que nos possibilita precificar o seguro de forma personalizada.

Pergunta 2: É importante a hora em que a pessoa passa em cada lugar? Ou dia da semana é um fator mais determinístico?

Resposta 2: Não há correlação provada entre o dia da semana em que a pessoa dirige e a probabilidade de acidente de carro.

A hora do dia, no entanto, como pode ser intuitivo para muitos, determina a chance de o carro ser roubado. A maior parte dos crimes ocorre à noite.

Pergunta 3: O sistema de precificação por uso é interessante pra seguros de motoristas de aplicativos? Quais parâmetros são mais relevantes para quem usa o carro prolongadamente?

Resposta 3: Sim, com certeza seria interessante um sistema desse tipo.

Sobre os parâmetros exatos que seriam mais interessantes, preciso consultar a área estatística da empresa para ter mais certeza, mas certamente informações sobre como o motorista usa o carro (a quantidade de freadas bruscas ao longo do trajeto, por exemplo) e por onde passa com ele, conforme mencionei anteriormente, são fundamentais, na minha experiência.

Pergunta 4: Empresas podem se beneficiar da redução do seguro também? Levando-se em conta viagens feitas por funcionários, a trabalho

Resposta 4: Hoje em dia nós seguramos empresas baseado puramente em quanto veículos têm na frota delas e quanto eles serão usados em média.

Se fosse possível diferenciar um motorista do outro, o cálculo do risco seria mais preciso, além de poder incentivar a boa condução.

Fizemos uma campanha com ideia parecida uma vez, chamada *Trânsito Mais Gentil*. A ideia era diminuir o prêmio cobrado de cada segurado caso ele comprovasse, ao fim do ano, que cometeu poucas ou nenhuma infração no trânsito, dando-nos a quantidade de pontos recebidos na carteira.

Pergunta 5: Existe o interesse de ser feita uma precificação mais personalizada de seguros de carro, usando-se dados de estilo de direção de cada indivíduo em vez de estatísticas gerais?

Resposta 5: Definitivamente sim, e acredito que essa seja a tendência do mercado no futuro próximo.

Existe uma plataforma que implementa algo parecido com o que vocês estão desenvolvendo, chama-se Azos[30].

Essa empresa precifica seguros de vida de forma dinâmica, levando em consideração não apenas estatísticas gerais sobre doenças que cada pessoa pode desenvolver, mas também e principalmente os hábitos do dia a dia de cada um e quais as novas condições de saúde do segurado a cada renovação de contrato.

Pergunta 6: Alguma consideração final?

Resposta 6: Quanto mais ajustado ao verdadeiro risco, melhor será o produto tanto para o cliente como para a seguradora, pois o preço mais justo para aquele serviço será o efetivamente cobrado.

Dito isso, vejo com muito bons olhos o projeto de vocês e contribuo com uma sugestão: algo tão personalizado assim poderia implementar um sistema equivalente ao *Open Banking*, de que tanto se fala hoje em dia, uma vez que o histórico de cada cliente não é transferível entre empresas de seguro.

7.2 Colunas da tabela do banco de dados

```
CREATE TABLE `tcc_main`.`info` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `timestamp` TIMESTAMP(3) NOT NULL,
  `user_token` VARCHAR(40),
  `name` VARCHAR(45) NOT NULL,
  `result` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE);

CREATE TABLE `acceleration` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `timestamp` TIMESTAMP(3) NOT NULL,
  `user_token` VARCHAR(40),
  `acceleration_x` VARCHAR(10) NOT NULL,
  `acceleration_y` VARCHAR(10) NOT NULL,
  `acceleration_z` VARCHAR(10) NOT NULL,
  `gravity_x` VARCHAR(10) NOT NULL,
  `gravity_y` VARCHAR(10) NOT NULL,
  `gravity_z` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE);

CREATE TABLE `location` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `timestamp` TIMESTAMP(3) NOT NULL,
  `user_token` VARCHAR(40),
  `latitude` VARCHAR(20) NOT NULL,
  `longitude` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE);
```

7.3 Código para o Lambda da AWS de comunicação com o banco de dados

```
import awsgi
import pymysql
from flask import Flask, request

app = Flask(__name__)

@app.route("/app_data", methods=["POST"])
def api_data():
    host = "open-db.clznpmxaqrkl.us-east-1.rds.amazonaws.com"
    user = "testadm"
    password = "testadm#1"
    database = "tcc_schema"

    #Connection
    connection = pymysql.connect(host=host, user=user,
    password=password, database=database)
    cursor = connection.cursor()

    data = request.get_json()

    method_name = data["method"]

    if method_name.split("_")[0] == "get" :
        filters = {
            "time_min" : data["time_min"],
            "time_max" : data["time_max"],
            "user_token" : data["user_token"]
        }

    data = data.get("data", [])

    if(method_name == "add_obd_info"):
        print("inserto data to info")
```

```
        query_insert_cmd = "INSERT INTO info (timestamp, user_token, name,
        result) VALUES (\%(timestamp)s, \%(user_token)s, \%(name)s, \%(result)s)"
        cursor.executemany(query_insert_cmd, data)

        connection.commit()
        return "deu bom"

    elif(method_name == "add_acceleration"):
        print("insert data to acceleration")

        query_insert_cmd = "INSERT INTO acceleration (timestamp, user_token,
        acceleration_x, acceleration_y, acceleration_z, gravity_x, gravity_y,
        gravity_z) VALUES (\%(timestamp)s, \%(user_token)s, \%(acceleration_x)s,
        \%(acceleration_y)s, \%(acceleration_z)s, \%(gravity_x)s, \%(gravity_y)s,
        \%(gravity_z)s)"
        cursor.executemany(query_insert_cmd, data)

        connection.commit()
        return "deu bom"

    elif(method_name == "add_location"):
        print("insert data to location")

        query_insert_cmd = "INSERT INTO location (timestamp, user_token,
            latitude, longitude) VALUES (\%(timestamp)s, \%(user_token)s,
            \%(latitude)s, \%(longitude)s)"
        cursor.executemany(query_insert_cmd, data)

        connection.commit()
        return "deu bom"

    elif(method_name == "get_obd_info"):
        print("get data from info")

        query_select_cmd = "SELECT * FROM info where timestamp >=
```

```
            '\%(time_min)s' and timestamp <= '\%(time_max)s'
            and user_token = '\%(user_token)s'" \%filters

        cursor.execute(query_select_cmd)

        return {"data": cursor.fetchall()}

    elif(method_name == "get_acceleration"):
        print("get data from acceleration")

        query_select_cmd = "SELECT * FROM acceleration where timestamp >=
            '\%(time_min)s' and timestamp <= '\%(time_max)s'
            and user_token = '\%(user_token)s'" \% filters
        cursor.execute(query_select_cmd)

        return {"data": cursor.fetchall()}

    elif(method_name == "get_location"):
        print("get data from location")

        query_select_cmd = "SELECT * FROM location where timestamp >=
            '\%(time_min)s' and timestamp <= '\%(time_max)s'
            and user_token = '\%(user_token)s'" \%filters
        cursor.execute(query_select_cmd)

        return {"data": cursor.fetchall()}
    else:
        return {"status": "error, route not found"}

if __name__ == "__main__":
    app.run(debug=True, port=5000)


def lambda_handler(event, context):
    return awsgi.response(app, event, context)
```

7.4 Código para o Lambda da AWS de geração do relatório sobre o perfil do motorista

```
import awsgi
from flask import Flask, request

import requests
import json

import matplotlib.pyplot as plt
from svglib.svglib import svg2rlg

import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication

from io import BytesIO
from reportlab.lib.pagesizes import letter
from reportlab.lib import utils
from reportlab.platypus import SimpleDocTemplate, Paragraph, Image
from reportlab.lib.styles import getSampleStyleSheet

from ride_parser import RealRideParser



app = Flask(__name__)

@app.route("/create_pdf", methods=["POST"])
def app_pdf():
    data = request.get_json()

    body_json_obd_info = {
        "method": "get_obd_info",
```

```
    "time_min" : data["time_min"],
    "time_max" : data["time_max"],
    "user_token" : data["user_token"]
}

body_json_acceleration = {
    "method": "get_acceleration",
    "time_min" : data["time_min"],
    "time_max" : data["time_max"],
    "user_token" : data["user_token"]
}

body_json_location = {
    "method": "get_location",
    "time_min" : data["time_min"],
    "time_max" : data["time_max"],
    "user_token" : data["user_token"]
}

data_obd_info = get_data_from_lambda(body_json_obd_info)
data_acceleration = get_data_from_lambda(body_json_acceleration)
data_location = get_data_from_lambda(body_json_location)


params = {
    "user_id" : data["user_token"],
    "date_beg" : data["time_min"],
    "date_end" : data["time_max"]
}

ride_parser = RealRideParser(True, **params)
metrics_map = ride_parser.generate_pdf_metrics()
img = ride_parser.risk_table_graph

fig = plt.figure(figsize=(4, 3))
```

```
    plt.plot([1,2,3,4])
    plt.ylabel('some numbers')

    create_PDF(img)
    return("pdf criado")



def get_data_from_lambda(body_json):
    url = "https://pntdpvkdsc.execute-api.us-east-1.amazonaws.com/default/app_data"

    headers = {
        'Content-Type': 'application/json',
    }

    response = requests.post(url, headers=headers, data=json.dumps(body_json))

    if response.status_code == 200:
        print("data successfully obtained from the server")
        return response
    else:
        print("failed to get data from the server")



def send_email(recipient_email, pdf_buffer):
    # Configura es do e-mail
    sender_email = "gabiru.bx@gmail.com"
    sender_password = "vbwiquhkjpjsmwdu"
    subject = "test email pdf"

    # Configura o do server SMTP do Gmail
    server_smtp = "smtp.gmail.com"
    port_smtp = 587

    # Cria o do objeto MIMEMultipart
```

```
    message = MIMEMultipart()
    message["From"] = sender_email
    message["To"] = recipient_email
    message["Subject"] = subject

    # Adiciona o corpo do e-mail (opcional)
    corpo_email = "Corpo do teste pdf"
    message.attach(MIMEText(corpo_email, "plain"))

    # Adiciona o arquivo PDF como anexo
    attached_document = MIMEApplication(pdf_buffer.read(), _subtype="pdf")
    attached_document.add_header(
        "Content-Disposition", f"attachment; filename=anexo.pdf")
    message.attach(attached_document)

    # Inicia a conexão com o server SMTP
    print("starting server")
    server = smtplib.SMTP(server_smtp, port_smtp)
    server.starttls()

    # Efetua login no server
    server.login(sender_email, sender_password)

    print("sending email")
    # Envia o e-mail
    server.sendmail(sender_email, recipient_email, message.as_string())

    # Fecha a conexão com o server
    server.quit()

def create_PDF(metrics_map):
    # Create a PDF document
    pdf_buffer = BytesIO()
    doc = SimpleDocTemplate(pdf_buffer, pagesize=letter)
```

```
    # Define a style sheet
    styles = getSampleStyleSheet()

    # Create a list of paragraphs for the content
    content = []

    # Add title
    content.append(Paragraph("My Data Report", styles['Title']))

    # Add space
    content.append(Paragraph("<br/>", styles['BodyText']))

    # Salvar a figura em um buffer de bytes
    metrics_map.savefig(pdf_buffer, format='png')
    pdf_buffer.seek(0)

    drawing=svg2rlg(pdf_buffer)

    doc.build(content)

    # Reset the buffer position to the beginning before sending
    pdf_buffer.seek(0)

    send_email("gabriel.morghett@gmail.com", pdf_buffer)

if __name__ == "__main__":
    app.run(debug=True, port=5000)

def lambda_handler(event, context):
    return awsgi.response(app, event, context)
```

7.5 Conversão de coordenadas geográficas em índices dos blocos no mapa

```
def convert_coord_to_chunk(self, coord_1, coord_2, max_coord_diff):
    coord_diff = coord_1 - coord_2
    chunk = (self.segmentation_rate * coord_diff) // max_coord_diff
    return chunk

    self.north = car_crimes_df["latitude"].max() + (0.5 / 111.11)
    self.south = car_crimes_df["latitude"].min() - (0.5 / 111.11)
    self.west = car_crimes_df["longitude"].max() + (0.5 / 111.11)
    self.east = car_crimes_df["longitude"].min() - (0.5 / 111.11)

    # maior distancia de norte a sul: 4378.4 km
    # maior distancia de leste a oeste: 4326.6 km
    self.segmentation_rate = 4400

    self.lat_diff = self.north - self.south
    self.long_diff = self.east - self.west


    car_crimes_df["chunk_i"] = car_crimes_df["latitude"]
        .apply(lambda x :
        self.convert_coord_to_chunk(x, self.south, self.lat_diff))
    car_crimes_df["chunk_j"] = car_crimes_df["longitude"]
        .apply(lambda x :
        self.convert_coord_to_chunk(x, self.west, self.long_diff))
```

CAPÍTULO 8

REFERÊNCIAS

[1] **US20130052614A1 - Driver Performance Metric**. Google Patents, 2012. Disponível em <patents.google.com/patent/US20130052614>. Acesso em 30 de jan. de 2023.

[2] **CAMPO GRANDE**, Paulo. "Novas Tecnologias: Carros Atuais Têm Até 100 Sensores a Bordo."Revista Quatro Rodas, Brasil, 12 de jun. de 2018. Disponível em <quatrorodas.abril.com.br/noticias/novas-tecnologias-carros-atuais-tem-ate-100-sensores-a-bordo/>. Acesso em 20 de fev. de 2023.

[3] WIKIMEDIA Foundation. **OBD-II PIDs**. Wikipedia, 2023. Disponível em <en.wikipedia.org/wiki/OBD-II_PIDs>. Acesso em 20 de mar. de 2023

[4] FINCO, Nina. **OBD**: o Que é e Para Que Serve o Protocolo OBD2?. Cobli Blog, 2021. Disponível em <www.cobli.co/blog/o-que-e-protocolo-obd2/>. Acesso em 20 de mar. de 2023.

[5] BARRETO, Victor. **What Is OBDII?** History of on-Board Diagnostics. Geotab, 2020 Disponível em <www.geotab.com/blog/obd-ii/>. Acesso em 20 de mar. de 2023.

[6] SBT News. **No Brasil, cerca de 32 pessoas morrem por dia em acidentes de trânsito**. SBT, Brasil, 22 de jan. de 2022. Disponível em <www.sbtnews.com.br/noticia/brasil/194388-no-brasil--cerca-de-32-pessoas-morrem-por-dia-em-acidentes-de-transito#:~:text=Em2021,foram11.647mortes,incidentesporhoranoBrasil.>. Acesso em 23 de abr. de 2023.

[7] **ACIDENTES De Trânsito São a Maior Causa De Morte De Pessoas De 5 a 29 Anos**. ONU News. Nações Unidas, 21 de nov. de 2021. Disponível em <news.un.org/pt/story/2021/11/1771092>. Acesso em 18 de abr. de 2023.

[8] **TOTAL Confirmed COVID-19 Deaths**. Our World in Data, 2023. Disponível em <ourworldindata.org/grapher/covid-deaths-income> . Acesso em 18 de abr. de 2023.

[9] **Em Quanto Tempo Os Carros Autônomos Serão o Novo 'Padrão'?**. Jaguar

Brasil. Disponível em <www.jaguarbrasil.com.br/news/em-quanto-tempo-os-carros-autonomos-serao-o-novo-padrao.html#:~:text=AindÃžstriadepesquisaIHS,podedemorarumpoucomais>. Acesso em 23 de abr. de 2023.

[10] **PREÇO sob demanda do Amazon EC2**. Amazon. Disponível em <aws.amazon.com/pt/ec2/pricing/on-demand/>. Acesso em 23 de abr. de 2023.

[11] **CALCULADORA de preço**. Microsoft Azure. Disponível em <azure.microsoft.com/de-de/pricing/calculator/>. Acesso em 23 de abr. de 2023.

[12] **PIRES**. "Android OBD-II Reader Application That Uses Pure OBD-II PID's Java API."GitHub. Disponível em <github.com/pires/android-obd-reader>. Acesso em 23 de abr. de 2023.

[13] **PAGE**, Vanessa. "Waze: The Pros and Cons."Investopedia, 12 de jan de 2023. Disponível em <www.investopedia.com/articles/investing/060415/pros-cons-waze.asp#:~:text=Wazeusesdatafromapp,thatcouldslowdowndrivers>. Acesso em 23 de abr. de 2023.

[14] **Scanner Automotivo Conector Obd2 Elm327 Bluetooth**. Mercado Livre, <produto.mercadolivre.com.br/MLB-2147325216-scanner-automotivo-conector-obd2-elm327-bluetooth-_JM#position=1&search_layout=grid&type=pad&tracking_id=509523dc-c530-481c-954c-0ecbe9c6a94c&is_advertising=true&ad_domain=VQCATCORE_LST&ad_position=1&ad_click_id=ODlhMjFlMzctNjQ5Ni00MTg1LWIzYWYtZjk3Y2U5MjlmNjNm>. Acesso em 23 de abr. de 2023.

[15] **SAIPRASERT**, Chalermpol. et al. "Driver Behaviour Profiling Using Smartphone Sensory Data in a V2I Environment". 2014 International Conference on Connected Vehicles and Expo (ICCVE), 2014. Disponível em <https://ieeexplore.ieee.org/abstract/document/7297609>. Acesso em 24 de abr. de 2023.

[16] **Imagem MySql e AWS RDS**. Disponível em <https://medium.com/99dotco/our-mysql-rds-upgrade-journey-cutting-down-downtime-by-11200-and-lessons-learned-1fa828e6009c>. Acesso em 20 de nov. de 2023.

[17] **Imagem Python e Folium**. Disponível em <https://www.google.com/url?sa=i&url=https%3A%2F%2Fm.youtube.com%2Fwatch%3Fv%3D4RnU5qKTfYY&psig=AOvVaw05zKbpT4lYD0v7v8M3hIZe&ust=1700594334592000&source=images&cd=vfe&opi=89978449&ved=0CA8QjRxqFwoTCMCfq-Kl04IDFQAAAAAdAAAAABAD>. Acesso em 20 de nov. de 2023.

[18] **Simulador Carla.** Disponível em <https://carla.org/>. Acesso em 11 de dez. de 2023.

[19] **Site Rota2030.** Disponível em <https://www.rota2030.com.br/>. Acesso em 11 de dez. de 2023.

[20] **Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V. (2017, October)**. "CARLA: An open urban driving simulator."In Conference on robot learning (pp. 1-16). PMLR.

[21] **Simulador OBD-II** Disponível em <https://freematics.com/store/index.php?route=product/product&product_id=71>. Acesso em 11 de dez. de 2023.

[22] **Ferreira Júnior, Jair da Silva**. "Análise de Perfil de Motoristas : Uma Investigação com Diferentes Sensores e Técnicas de Aprendizado de Máquina / Jair da Silva Ferreira Júnior. — 2018"101 f. : il. color.

[23] **Shibata, Danilo Jun; Soares, Leandro Donizetti**. "Análise de Direção de motoristas Utilizando o Protocolo OBD2 e sensores Embarcados."

[24] **Bethge, Johanna, et al**."Model Predictive Control with Gaussian-Process-Supported Dynamical Constraints for Autonomous Vehicles."arXiv preprint arXiv:2303.04725 (2023).

[25] **Ferreira J Júnior, Carvalho E, Ferreira BV, de Souza C, Suhara Y, et al. (2017)**. "Driver behavior profiling: An investigation with different smartphone sensors and machine learning."PLOS ONE 12(4): e0174959.

[26] **Júnior, J. Ferreira, and Gustavo Pessin**. "Análise de perfil de motoristas: Detecção de eventos por meio de smartphones e aprendizado de máquina."Anais do WOCCES 2016 Workshop de Comunicação em Sistemas Embarcados Críticos. 2016.

[27] **Documentação dos acelerômetros para desenvolvimento Android.** Disponível em <https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview>. Acesso em 11 de dez. de 2023.

[28] **Seifi, Mohammad Reza, et al.** "Experimental investigation of a diesel engine power, torque and noise emission using water–diesel emulsions."Fuel 166 (2016): 392-399.

[29] **Documentação do OneTapSignIn do Firebase.** Disponível em <https://firebase.google.com/docs/auth/android/google-signin?hl=pt-br>. Acesso em 11 de dez. de 2023.

[30] **Plataforma de seguro de vida com preço dinâmico.** Disponível em <https://www.azos.com.br>. Acesso em 11 de dez. de 2023.

[31] **de TEFFÉ, Chiara Spadaccini, and Mario Viola.** "Tratamento de dados pessoais na LGPD: estudo sobre as bases legais."**civilistica.com**, v. 9, n. 1, p. 1-38, 9 maio 2020. Acesso em 11 de dez. de 2023.

[32] **Garber, Nicholas J.; Gadirau, Ravi.** "Speed Variance and Its Influence on Accidents."AAA Foundation for Traffic Safety, 1730 M Street, N.W., Suite 401, Washington, DC 20036.

[33] **Geospatial Sao Paulo Crime Database.** Disponível em <https://www.kaggle.com/datasets/danlessa/geospatial-sao-paulo-crime-database/>. Acesso em 11 de dez. de 2023.

[34] **Paulista, Augusta, Sé: veja as ruas e as regiões com mais roubos e furtos de celular em SP.** Disponível em <https://g1.globo.com/monitor-da-violencia/noticia/2023/03/26/paulista-augusta-se-veja-as-ruas-e-as-regioes-com-mais-roubos-e-furtos-de-celular-em-sp.ghtml>. Acesso em 11 de dez. de 2023.

[35] **Cálculo das coordenadas geográficas.** Disponível em <https://www.teleco.com.br/tutoriais/tutorialsmsloc2/pagina_5.asp#:~:text=Cada%20grau%20de%20uma%20latitude,a%20aproximadamente%20111%2C11%20km.>. Acesso em 11 de dez. de 2023.

[36] **Cálculo das coordenadas geográficas.** Disponível em <https://www.teleco.com.br/tutoriais/tutorialsmsloc2/pagina_5.asp#:~:text=Cada%20grau%20de%20uma%20latitude,a%20aproximadamente%20111%2C11%20km.>. Acesso em 11 de dez. de 2023.

[37] **Pontos extremos do Brasil.** Disponível em <https://brasilescola.uol.com.br/brasil/pontos-extremos-do-brasil.htm>. Acesso em 11 de dez. de 2023.

[38] **17 frases de Nietzsche que qualquer pessoa deveria conhecer.** Disponível em <https://www.pensador.com/reflexoes_de_nietzsche/>. Acesso em 11 de dez. de 2023.

[39] **Liberdade não é fazer o que se quer,... Jean-Paul Sartre.** Disponível em <https://www.pensador.com/frase/Mzg2MzU/>. Acesso em 11 de dez. de 2023.

[40] **Nível gratuito da AWS.** Disponível em <https://aws.amazon.com/pt/free/?gclid=CjwKCAiAg9urBhB_EiwAgw88mZh2amGqJb7wPm7xH8CmUzSD9OF2PD8fRgq

R56vKEbjxGjuY4b8mTRoCCa4QAvD_BwE&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=categories%23compute&trk=d0b462ed-a9ff-4714-8a75-634758c49d4c&sc_channel=ps&ef_id=CjwKCAiAg9urBhB_EiwAgw88mZh2amGqJb7wPm7xH8CmUzSD9OF2PD8fRgqR56vKEbjxGjuY4b8mTRoCCa4QAvD_BwE:G:s&s_kwcid=AL!4422!3!531081610020!e!!g!!amazon%20free%20cloud%20server!12024810921!121376982172>. Acesso em 11 de dez. de 2023.

[41] **Loureiro, Thiago Wilhelmsen, and Celso Pupo Pesce.** "Análise paramétrica do conjunto pistão, biela e árvore de manivelas com foco na redução de perdas por atrito e de consumo de combustível.- Trabalho de conclusão de curso (Mestrado Profissional em Engenharia Automotiva) - Escola Politécnica da Universidade de São Paulo (2009).

[42] **Base de dados UAH-Driveset.** Disponível em <https://www.robesafe.uah.es/personal/eduardo.romera/uah-driveset/>. Acesso em 12 de dez. de 2023.

[43] **Versão inicial do aplicativo desenvolvido.** Disponível em <https://github.com/pires/android-obd-reader>. Acesso em 12 de dez. de 2023.