Um Estudo da Mobilidade Urbana em Belo Horizonte Utilizando Mobile Crowdsensing

Pesquisa Tecnológica

André Luiz M. Barreto[1], Clodoveu A. Davis Júnior[1]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG) - Belo Horizonte - MG - Brasil

{clodoveu,andrembarreto}@dcc.ufmg.br

Abstract. *This study presents the structure of a system that enables the analysis of the comfort level experienced during journeys on public transportation routes in Belo Horizonte, using mobile crowdsensing. Communication is established with acceleration and rotation sensors of mobile devices, as well as retrieving their location. The data is then used to generate georeferenced visual representations with an emphasis on parameters related to the perceived discomfort problem.*

Resumo. *Este estudo apresenta a estrutura de um sistema que habilita análises do nível de conforto experienciado em jornadas por rotas do sistema de transporte público em Belo Horizonte, utilizando mobile crowdsensing. É feita a comunicação com sensores de aceleração e rotação de dispositivos móveis, assim como é recuperada a localização desses, e então os dados são utilizados para gerar representações visuais georreferenciadas com ênfase em parâmetros relacionados ao problema de desconforto percebido.*

1. Introdução

Em grandes cidades de países em desenvolvimento como o Brasil, é comum que se verifiquem condições inadequadas ou mesmo precárias do sistema de transporte público, que passa por processos contínuos de manutenção ou expansão em razão da necessidade de adaptação ao aumento da população e da complexidade do ambiente urbano. Quando se consideram também os níveis crescentes de congestionamentos, poluição e acidentes de trânsito (Cardoso, 2007), conclui-se que o cidadão que já enfrenta cotidianamente diversos problemas que podem ser considerados externos à experiência de locomoção em si, desenvolve uma percepção sobre esse sistema que tende a se tornar ainda mais negativa, quando ele é submetido a um padrão de movimentação do veículo que possa ser identificado, de forma geral, como irregular ou desagradável.

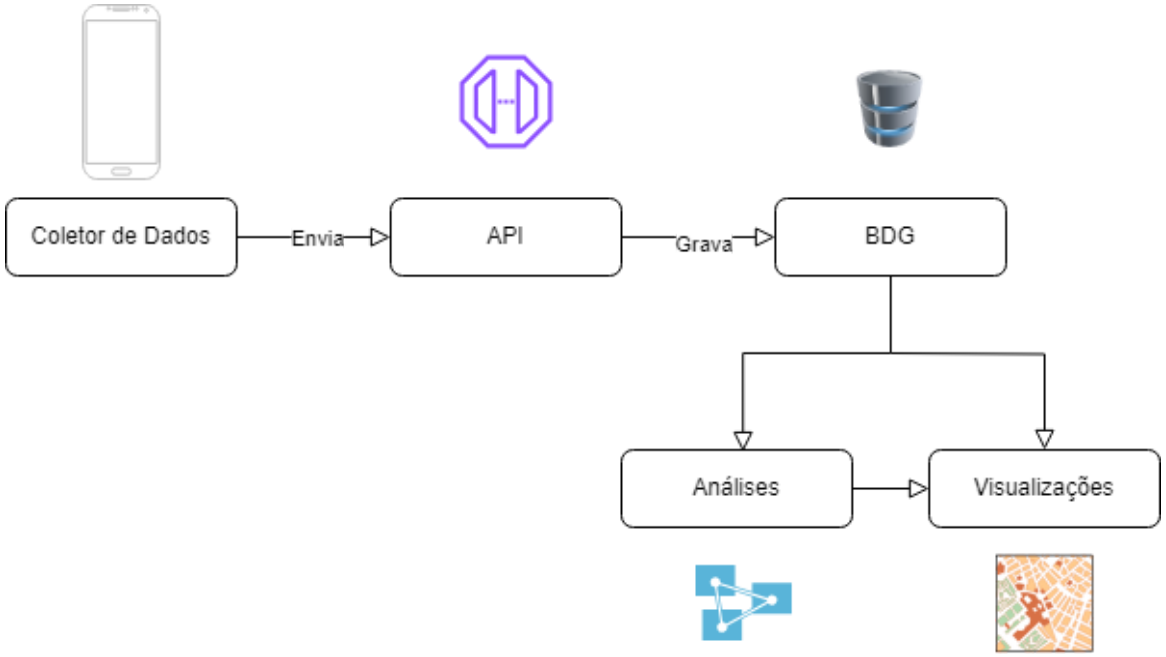
O objetivo do trabalho apresentado, baseado em um trabalho anterior, desenvolvido também sob a orientação do professor Clodoveu (De-Stefano Santos & Davis Jr, 2018), é avaliar o nível de conforto ao longo de trajetos em veículos de transporte coletivo em Belo Horizonte.

A estrutura do trabalho pode ser descrita da seguinte forma: Os dados são coletados com o uso de um aplicativo mobile. Esse tipo de coleta se encaixa no que é chamado de crowdsensing passivo. Esses dados são enviados para um endpoint de uma API, onde recebem um pré-tratamento e são armazenados em um banco de dados geográficos. Por fim, são realizadas análises e produzidas representações visuais dos fenômenos de interesse, especificamente, aqueles relacionados ao nível de conforto. A figura 1 representa graficamente com um diagrama o que foi apresentado.

Os dados selecionados são relativos à aceleração, à rotação e à localização do dispositivo durante o trajeto. A aceleração instantânea pode ser vista como uma medida representativa da percepção do nível de desconforto, pois se relaciona com a saída de um estado de inércia.

Figura 1

Diagrama da estrutura do que foi produzido no trabalho



2. Referencial

2.1. Referencial Teórico

2.1.1. Mobile crowdsensing

Um dos conceitos centrais para o trabalho desenvolvido, *mobile crowdsensing* pode ser caracterizado como o compartilhamento de dados e a extração de informações coletivamente por indivíduos com uso de dispositivos equipados com sensores de ambiente para medir e mapear fenômenos de interesse comum (Ganti et al., 2011). Uma categoria mais específica, proposta no trabalho de Mateveli et al. (2015), é a de *crowdsensing* passivo, em que o usuário não precisa interagir diretamente com o sistema para que ocorra a captura de informação por sensores nos dispositivos móveis.

2.1.2. Volunteered Geographic Information (VGI)

A produção de conteúdo online pelos indivíduos, mais especificamente de dados geográficos na última década, é uma tendência que foi denominada *Volunteered Geographic Information*. Cidadãos podem gerar informação útil sobre o ambiente em que vivem, agindo como sensores (Mateveli et al., 2015).

2.1.3. Domain-Driven Design (DDD)

DDD é um conjunto de princípios que podem ser utilizados na etapa de projeto de um sistema de software (Valente, 2020). Defende-se que o projeto deve ser orientado pelo domínio da aplicação, e não por tecnologias, frameworks ou estruturas da linguagem a ser utilizada, e isso deve ser refletido na arquitetura do sistema desenvolvido. Para isso, os desenvolvedores precisam ter conhecimento dos conceitos específicos do domínio e padronizar as referências a termos e operações.
Os tipos de objetos presentes no sistema podem ser: Entidades, caracterizadas como objetos que possuem identidades únicas; Objetos de valor, que são caracterizados pelo valor de seus atributos, e não possuem identidades únicas; Serviços, que implementam operações entre outros objetos e não possuem estado; Agregados, coleções de entidades e objetos de valor; e Repositórios, usados para recuperar referências a objetos identificáveis, geralmente de bancos de dados.

2.1.4. Arquitetura Hexagonal

Conceito proposto por Alistair Cockburn, a arquitetura hexagonal se propõe a possibilitar o desenvolvimento de sistemas cujas classes do domínio sejam desacopladas das tecnologias e recursos externos utilizados. Conforme Valente (2020), um sistema construído sob os princípios da arquitetura hexagonal é capaz de oferecer maior reusabilidade de código, testabilidade e coesão, e está menos suscetível a fatores relacionados à dependência de tecnologias e acoplamento.
A comunicação entre as classes do domínio e as tecnologias e recursos externos é mediada por adaptadores, que implementam a tradução dos parâmetros das requisições entre as camadas.

2.1.5. Object-Relational Mapping (ORM)

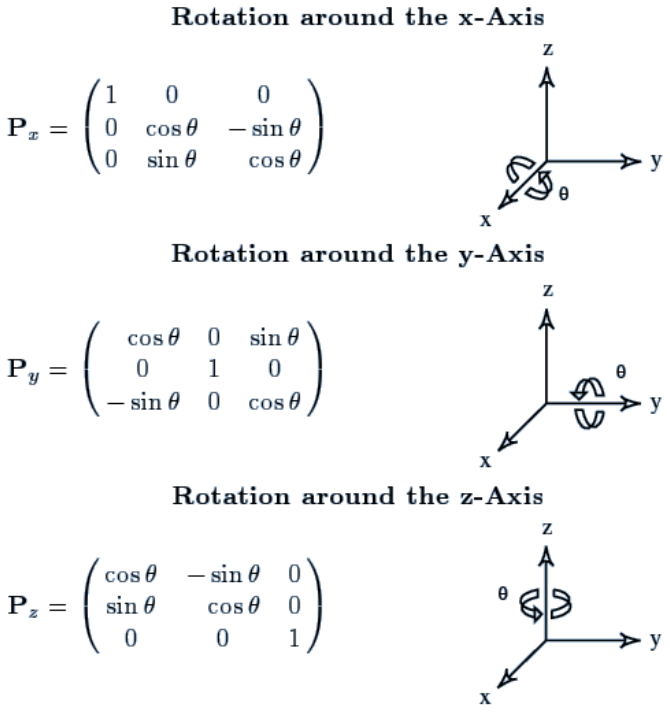
*ORM*s se apresentam como uma possível solução para os conflitos gerados pelas diferenças entre os paradigmas seguidos por bancos de dados relacionais e linguagens de programação orientadas a objetos, quando esses são selecionados como tecnologias a serem utilizadas em uma aplicação (Torres et al., 2017). Um *ORM* implementa interesses de persistência de dados em uma camada isolada, fornecendo abstrações para que os dados possam ser referenciados internamente como objetos, e não tabelas, no ambiente da linguagem de programação.

2.1.6. Matrizes de rotação

Uma matriz de rotação (ou de transformação) é uma matriz ortogonal que pode ser utilizada para realizar uma transformação das componentes de um vetor através de uma operação de multiplicação (Chow, 2013). Uma matriz de rotação básica, ou elementar, rotaciona um vetor em um ângulo θ em torno de um dos eixos x, y ou z, utilizando a regra da mão direita, em três dimensões, com a operação de multiplicação à esquerda. A figura 2 apresenta as três matrizes de rotação básica.

Figura 2

Representação das matrizes de rotação elementar



How to rewrite multivariate random functions as univariate to do cokriging. A flexible algorithm - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Definition-of-the-rotation-matrices-trough-the-axis-x-y-and-z-taken-from-Meyer-2000_fig3_237030100 [accessed 20 Jul 2024]

2.2. Trabalhos Relacionados

2.2.1. Street Bump

Um aplicativo desenvolvido na cidade de Boston, nos Estados Unidos, o Street Bump permite a coleta e o envio de dados relacionados às vias da cidade enquanto os usuários dirigem. São utilizados dados de localização por GPS e dos acelerômetros dos smartphones para identificar e mapear problemas nas vias (Carrera et al., 2013).

2.2.2. Versão anterior do trabalho

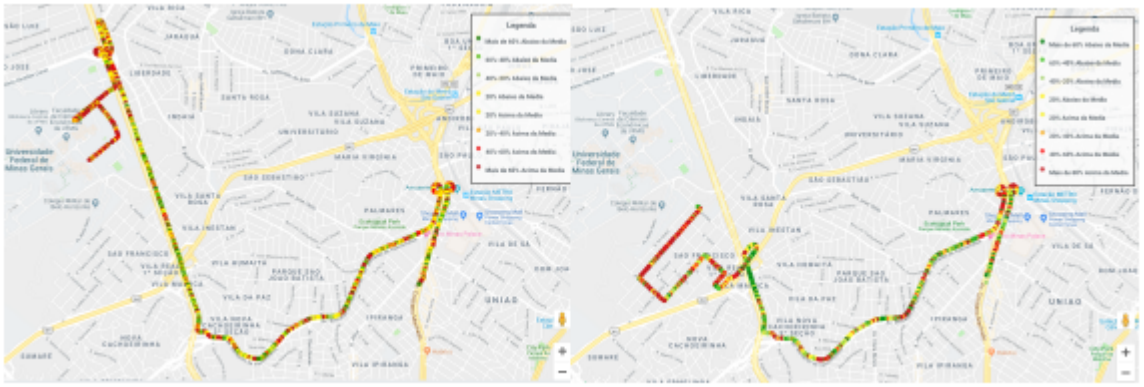
Coletor de dados desenvolvido por aluna em trabalho anterior (De-Stefano Santos & Davis Jr, 2018). Os dados de localização e dos sensores de aceleração e rotação de smartphones são coletados através de uma aplicação web enquanto os usuários transitam pela cidade em trajetos de ônibus, e os resultados são mapeados. Na figura 3 é mostrada uma das telas da interface gráfica do coletor.
Alguns métodos de análise dos dados foram desenvolvidos e aplicados, e os resultados foram então mapeados para o trajeto de duas linhas de ônibus da cidade, como mostra a figura 4. Três tipos de análise foram utilizados: intensidade do movimento, tipo do movimento e variação dos ângulos de rotação. Vale observar que a coleta era realizada sob o pressuposto de que o usuário tentaria manter o dispositivo na mesma posição durante todo o trajeto.

Figura 3

Captura da tela de monitoramento da interface de usuário do coletor

Figura 4

Visualização de trajetos por intensidade de movimento



(a) Viagem 50.

(b) Viagem 9550

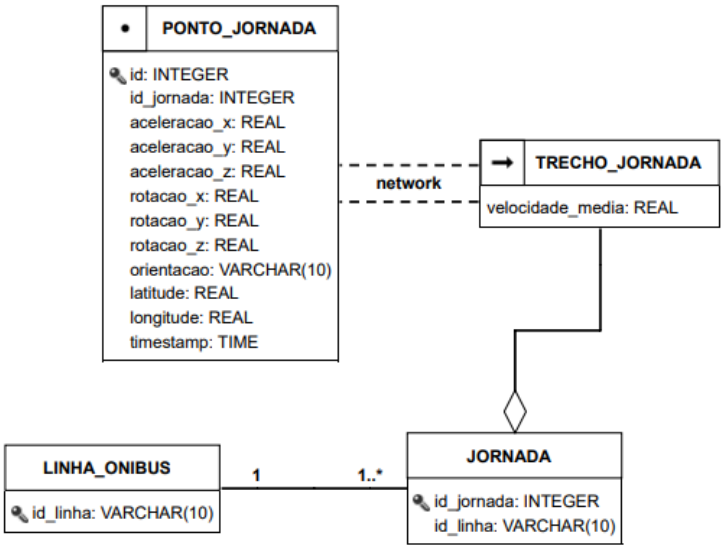
3. Contribuição

3.1. Modelagem e BDG

Existem quatro entidades: Linha de Ônibus, Jornada, Ponto de Jornada e Trecho de Jornada. Os pontos guardam uma referência para a jornada à qual eles pertencem, e as jornadas guardam uma referência para a linha de ônibus que foi percorrida. Isso é suficiente para que diversas configurações para análises sejam possíveis: analisar jornadas isoladas, comparar jornadas que tenham ocorrido em uma mesma linha de ônibus, agrupar todas as jornadas que se referem a uma mesma linha de ônibus, comparar jornadas ao longo dos horários em um dia. Um trecho de jornada é definido por dois pontos adjacentes e guarda a velocidade média entre eles. Essa informação agrega valor à análise exploratória, e é interessante para a visualização da jornada. Essa entidade não foi traduzida em uma tabela no banco, e as entradas são obtidas através de consultas. A figura 5 mostra um diagrama no modelo OMT-G do que foi apresentado.
Em relação ao período anterior deste trabalho, foram adicionados à entidade Ponto de Jornada os atributos de resultante da aceleração sobre os três eixos e a aceleração sobre cada eixo com uma referência fixa de sistema de coordenadas. Detalhes sobre como esses valores são calculados serão apresentados na seção 3.4.1.
Foi utilizado o gerenciador PostgreSQL em conjunto com a extensão PostGIS, que habilita a manipulação de valores geoespaciais, para o controle do banco.

Figura 5

Diagrama no modelo OMT-G do banco de dados geográficos



3.2. Coletor de Dados

Foi desenvolvido, na linguagem C++ e com a utilização do framework Qt, um aplicativo *mobile* para a coleta de dados de aceleração, rotação e localização de um dispositivo

durante trajetos em rotas urbanas, que pode ser encontrado neste repositório. Para obter as leituras dos sensores, foi utilizado o módulo *Qt Sensors*, e, para obter a localização do dispositivo, a API *Qt Positioning*, ambos parte do framework mencionado. Foi utilizado o mecanismo de *Signals & Slots* (Qt Company, 2024) para conectar eventos de novas leituras dos sensores e das coordenadas (*Signals*) a métodos que gravam seus valores (*Slots*).

Figura 6

Conexão dos eventos de novas leituras (signals) às funções de tratamento (slots)

```
_accelerometer = new QAccelerometer(this);
connect(_accelerometer, SIGNAL(readingChanged()), this, SLOT(registerAccelerometerReading()));

_rotationSensor = new QRotationSensor(this);
connect(_rotationSensor, SIGNAL(readingChanged()), this, SLOT(registerRotationReading()));

m_accessToPosition = false;
_source = QGeoPositionInfoSource::createDefaultSource(this);

if(_source) {
    connect(_source, &QGeoPositionInfoSource::positionUpdated, this, &MobilityData::registerGeolocation);
```

Ao final do trajeto, interrompendo a coleta, o usuário tem a possibilidade de permitir o envio dos dados.

Grande parte do sistema já estava implementado ao final do período da MSI I, restando alguns ajustes e refatorações para alcançar a versão final utilizada. Uma revisão completa sobre o processo de desenvolvimento dessa aplicação está presente no relatório final da MSI I. A seguir, serão feitos comentários sobre as mudanças feitas durante o curso da parte II da disciplina.

A coleta e inclusão dos dados do sensor de orientação não fazem mais parte do sistema. Esses dados foram considerados desnecessários para a análise posterior, tendo em vista que o processo de transformação dos eixos de aceleração para referências fixas utiliza apenas a rotação, e não a orientação, e não há necessidade de manter um controle desta durante a coleta para possivelmente alertar o usuário, já que a restrição de ele precisar manter o dispositivo na mesma posição durante todo o trajeto foi descartada.

Um pequeno ajuste na estrutura dos dados enviados também foi feito: a linha de ônibus e a lista de pontos agora são mais bem definidos no objeto JSON enviado via requisição para a API de recebimento. Antes, a estrutura era como a de um *array*, em que, na primeira posição, era inserida a linha de ônibus, e, nas posições seguintes, os pontos, um a um. Agora, a estrutura é como a de um mapa (ou dicionário, ou objeto), em que a linha de ônibus é referenciada com uma chave específica e um *array* dedicado para os pontos também. Isso deixa as operações de verificação da estrutura do objeto recebido e de acesso aos dados mais limpas do lado do servidor.

Por fim, foram adicionadas as funcionalidades de salvar localmente os dados de uma jornada após sua conclusão, e de recuperar e enviar esses dados, como forma de contornar a limitação de os testes terem sido realizados sem um servidor remoto. Dessa forma, ao final de uma coleta, se o servidor de testes não estivesse acessível, os dados não seriam perdidos.

3.3. API de recebimento dos dados

Foi desenvolvida, na linguagem Python, uma aplicação que configura um endpoint para o recebimento dos dados coletados anteriormente pelo aplicativo móvel, que pode ser encontrada neste repositório. Podemos dividir a estrutura dessa aplicação em 3 partes principais: recebimento, tratamento e armazenamento dos dados. Foi seguida, dentro do possível, a arquitetura hexagonal junto com o DDD (Valente, 2020). Existem, portanto, entidades, repositórios, serviços, e adaptadores, e as dependências são tratadas por meio de interfaces e com uso de injeção de dependência.
Para o recebimento dos dados, foi criada uma rota para requisições POST, e a principal tecnologia utilizada foi o framework Flask. O aplicativo mobile, ao final da coleta, envia os dados para o endpoint via uma requisição. Quando esta é detectada, é feita uma checagem da estrutura dos dados recebidos, e então o tratamento e o armazenamento dos dados são requisitados de camadas mais internas.
Na figura 7, é mostrado um trecho de código com a definição da rota e a chamada para um adaptador *HTTP*, que faz outras verificações e então chama uma classe de serviço, que será apresentada adiante.

Figura 7

Configuração de rota e função de tratamento para recebimento de dados

```python
@app.post('/receive-mobility-data')
async def receive_mobility_data():
    try:
        data = request.json
    except UnsupportedMediaType:
        return jsonify({'status': HTTPStatusCode.BAD_FORMAT.value})

    return http_adapter.receive_mobility_data(data)
```

Na camada do adaptador, verifica-se a presença de todos os dados necessários para o tratamento, e, em caso positivo, uma classe de serviço é chamada para realizar as transformações necessárias e requisitar a gravação dos dados no banco, o que é apresentado na figura 8.
A classe de serviço, como mencionado acima, adiciona novas informações, calculadas a partir dos dados originais, nomeadamente, a resultante da aceleração sobre os três eixos e os valores de aceleração em cada eixo fixo após aplicação das matrizes de transformação. Essa operação está representada na figura 9.
A implementação dessas operações foi extraída para uma classe separada, para manter a modularidade do código.
A classe de serviço foi implementada utilizando apenas a biblioteca padrão do Python, em uma tentativa de manter o domínio livre das especificidades de diferentes tecnologias, e a classe para a transformação dos dados utiliza a biblioteca NumPy, que acredita-se ser bastante estável e simples de utilizar para o nível de operação necessário, de forma que a camada de domínio não deveria ser considerada comprometida.

Figura 8

Implementação de método na camada de adaptadores para a verificação dos dados

```python
def receive_mobility_data(self, mobility_data) -> Response:
    try:
        bus_line = mobility_data['bus_line']
    except KeyError:
        return jsonify({'status': HTTPStatusCode.MISSING_BUS_LINE.value})
    try:
        points_data = mobility_data['points']
    except KeyError:
        return jsonify({'status': HTTPStatusCode.MISSING_POINTS.value})

    try:
        points = [self.point_from_json(point) for point in points_data]
    except KeyError:
        return jsonify({'status': HTTPStatusCode.BAD_FORMAT.value})

    self.__journey_service.register_journey(bus_line, points)
    return jsonify({'status': HTTPStatusCode.SUCCESS.value})
```

Figura 9

Implementação de método na camada de serviço para a requisição de gravação dos dados

```python
def register_journey(self, bus_line: str, journey_points: list[Point]) -> None:
    self.__bus_line_repository.create_if_not_exists(bus_line)
    journey_id = self.__journey_repository.create(bus_line)
    points_additional_data = self._get_points_additional_data(journey_points)

    self.__journey_points_repository.batch_create(journey_id, journey_points, points_additional_data)
```

Os repositórios foram implementados utilizando um pacote de ORM para o python, chamado Peewee. Os modelos para as tabelas, representados como classes, foram definidos com uma correspondência de um para um com as entidades da modelagem original, exceto pelo trecho de jornada, que não foi mapeado para uma tabela no banco. Para integrar o aspecto "Geo" à tabela de pontos da jornada, foi definida uma extensão da classe genérica de um campo (*Field*) da ORM: *GeometryField* é usada neste contexto para guardar as coordenadas dos pontos.

Figura 10

Extensão de um campo genérico para a inclusão de dados geoespaciais

```python
class GeometryField(Field):
    field_type = 'geometry'

    def db_value(self, value):
        return fn.ST_GeomFromText(*value)

    def python_value(self, value):
        return wkb.loads(value)
```

db_value é uma função que faz a tradução de um valor no Python para o que será inserido na coluna da tabela no banco, e *python_value* realiza processo inverso. A primeira é chamada atualmente de forma indireta em toda requisição de inserção ao banco, a partir da execução do método apresentado na figura 11.
Já a segunda não é utilizada até o momento, pois os dados não estão sendo consultados no contexto da aplicação, e sim no ambiente do gerenciador do banco de dados diretamente.

Figura 11

Implementação de método para a inserção de pontos na tabela do banco

```python
def batch_create(self, journey_id: int, points: list[Point], points_additional_data: list[PointAdditionalData]):
    data = [
        {
            'journey': journey_id,
            'timestamp': point.timestamp,
            'geometry': (f'POINT ({point.longitude} {point.latitude})', EPSG_WGS84),
            'acceleration_x': point.acceleration_x,
            'acceleration_y': point.acceleration_y,
            'acceleration_z': point.acceleration_z,
            'rotation_x': point.rotation_x,
            'rotation_y': point.rotation_y,
            'rotation_z': point.rotation_z,
            'transformed_x_acceleration': additional_data.transformed_x_acceleration,
            'transformed_y_acceleration': additional_data.transformed_y_acceleration,
            'transformed_z_acceleration': additional_data.transformed_z_acceleration,
            'resultant_acceleration': additional_data.resultant_acceleration
        }
        for point, additional_data in zip(points, points_additional_data)
    ]

    with db.atomic():
        JourneyPoint.insert_many(data).execute()
```

3.4. Análise dos Dados

3.4.1. Transformação dos dados de aceleração

É uma parte preliminar da análise, realizada ainda na etapa de recebimento dos dados e antes da gravação, pelo servidor. Os valores de aceleração retornados pelo módulo de acelerômetro utilizado têm eixos de referência fixos em relação ao celular, o que significa, por outro lado, que eles são móveis em relação à superfície. Em outras palavras, os eixos x, y e z são rotacionados junto com o celular durante a coleta, conforme este é movido, então, para não restringir os movimentos do usuário, optou-se por fazer uma transformação das componentes x, y e z da aceleração com o uso de matrizes de rotação, tendo como parâmetros os valores de aceleração e rotação coletados de cada ponto.
É feita uma rotação combinada sobre os 3 eixos, o que pode ser obtido ao multiplicar à esquerda cada matriz de rotação elementar pelo vetor de aceleração com as três componentes. Por opção, e pela propriedade associativa da multiplicação de matrizes, as três matrizes são multiplicadas antes, e o resultado pelo vetor. A escolha dessa operação se deve ao fato que consta na documentação oficial da classe *QRotationReading* que é utilizado o sistema de coordenadas cartesianas de mão direita, e, o dispositivo que indica rotações sobre os eixos x, y e z iguais a 0 está com a face para cima, paralelo à superfície,

o que permite interpretar o eixo z como apontando para cima, e facilita anular a influência da gravidade. A função apresentada na figura 12 contém parte da implementação do que foi apresentado nesta seção.

Figura 12

Implementação de método que retorna a aceleração com as componentes transformadas

```python
def get_transformed_acceleration_values(acceleration_values, rotation_values) -> tuple[float, float, float]:
    """
    Performs an active transformation of the `acceleration_values` for a fixed coordinate system,
    using the `rotation_values` based on the right-hand cartesian system
    """
    roll, pitch, yaw = rotation_values
    R_x, R_y, R_z = AccelerationDataHandler.get_rotation_matrices(roll, pitch, yaw)

    combined_rotation_matrix = R_z @ R_y @ R_x
    transformed_acceleration = combined_rotation_matrix @ acceleration_values

    return transformed_acceleration
```

A resultante da aceleração em cada ponto é calculada a partir desses valores, e adicionada aos dados. A implementação foi feita utilizando a biblioteca numpy, do python, como mencionado anteriormente.

3.4.2. Análise Exploratória e Visualização

Após a coleta, transformação e gravação dos dados, é feita uma análise exploratória, com a utilização do sistema de informação geográfica QGIS. É possível fazer uma conexão com o banco de dados a partir da aplicação para desktop, e então realizar consultas e exibir os resultados em camadas visuais, com um mapa como plano de fundo.

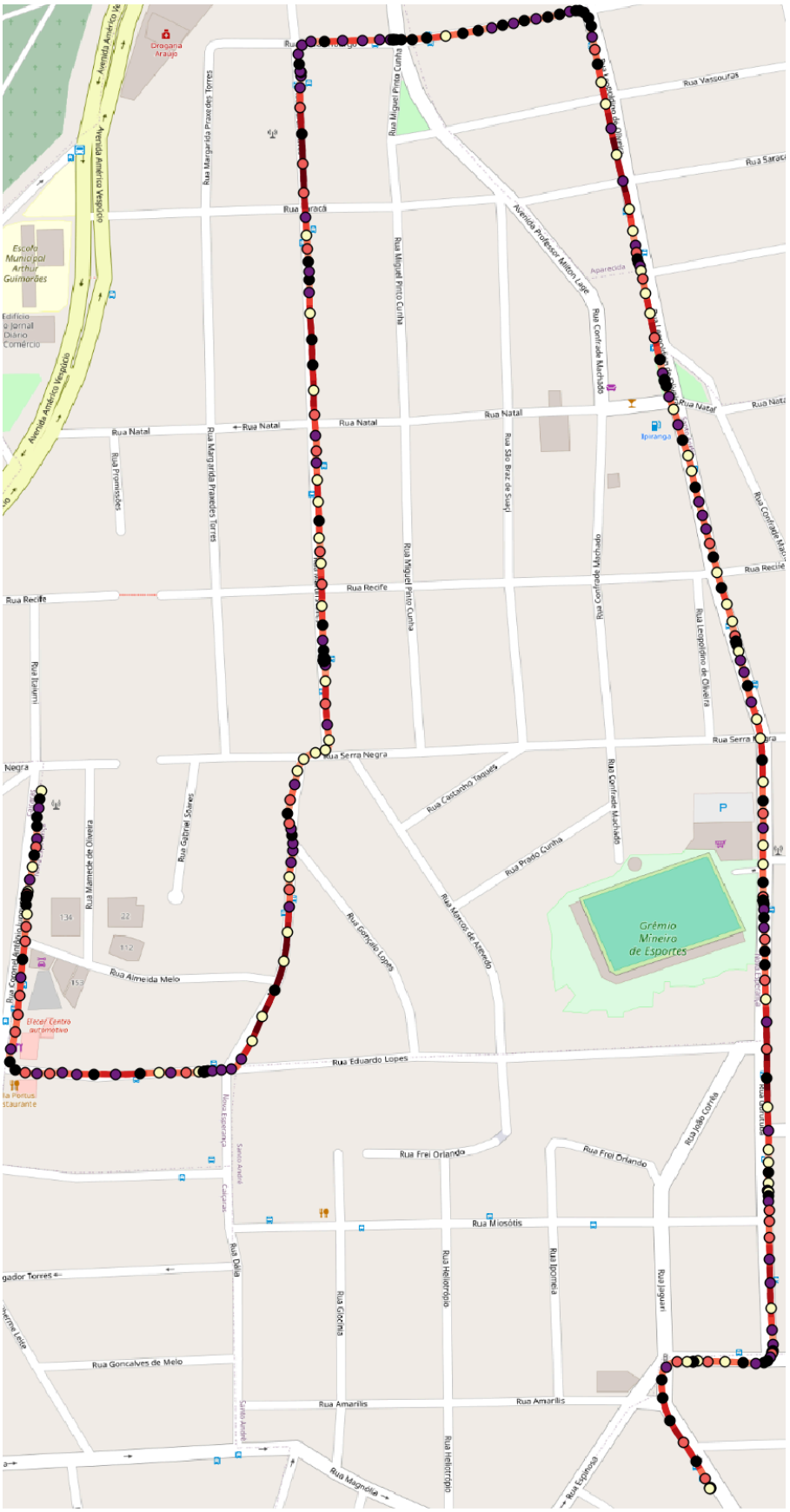
Parâmetros como a aceleração resultante em cada ponto e a velocidade média em cada trecho definido por pontos adjacentes foram manipulados a fim de gerar uma representação visual da jornada que apresentasse de forma fiel os fatores que podem influenciar o conforto dos passageiros.

A figura 13 mostra um trecho mapeado da rota 9404, em que a aceleração resultante em cada ponto foi classificada em função do desvio padrão, e codificada em uma escala de cores, assim como a velocidade média em cada trecho, que foi classificada em intervalos iguais. Para os pontos: quanto mais claro, maior a aceleração resultante. Para os trechos: velocidades médias mais altas são representadas em vermelho mais escuro.

É possível notar a irregularidade nas sequências de pontos e trechos. Há diversas ocorrências de pontos representados com a cor mais clara adjacentes a pontos representados com a cor mais escura. Nota-se também que há várias mudanças súbitas na aceleração resultante realizadas em altas velocidades médias, indicadas pelas sequências de trechos representados em vermelho mais escuro. Essas conclusões são coerentes com a percepção pessoal durante a coleta, no trajeto.

Figura 13

Representação visual de trecho da rota 9404



4. Conclusões e Trabalhos Futuros

É evidente que mais rotas poderiam ter sido cobertas e mais parâmetros considerados na análise. Entretanto, tendo em vista a restrição de tempo, o foco foi direcionado ao desenvolvimento de ferramentas com alta manutenibilidade e bem projetadas e documentadas. A adoção de práticas cobertas pela Arquitetura Hexagonal e pelo DDD, e a tentativa de produzir um código limpo e com um projeto inicial de testes automáticos, têm como objetivo permitir que os produtos deste trabalho possam ser compreendidos, utilizados e ampliados com maior facilidade por qualquer parte interessada na extensão deste trabalho.

Para trabalhos futuros, é possível citar a possibilidade de analisar os dados com um ferramental diferente, como o aprendizado de máquina, para a detecção de eventos específicos, como buracos e curvas muito acentuadas, mediante a obtenção de dados anotados. Também é útil que a API retorne algum output para o usuário, por meio de uma rota para requisições do tipo GET, o que deixaria a experiência de uso do coletor mais completa. Seria extremamente benéfico desenvolver um serviço para a execução da coleta em plano de fundo, para que não seja necessário manter a aplicação aberta o tempo todo.

Todos esses pontos foram mencionados, ainda na parte I, como atividades a serem cumpridas possivelmente durante o curso normal da disciplina de monografia. No entanto, novas perspectivas foram consideradas ao longo do projeto, e as decisões foram tomadas visando à continuidade.

5. Referências Bibliográficas

Cardoso, L. (2007). Transporte público, acessibilidade urbana e desigualdades socioespaciais na Região Metropolitana de Belo Horizonte.

De-Stefano Santos, P. I., & Davis Jr, C. A. (2018). Análise de Transporte Público utilizando Mobile Crowdsensing. Trabalho de Conclusão de Curso (Sistemas de Informação), UFMG.

Ganti, R. K., Ye, F., & Lei, H. (2011). Mobile crowdsensing: current state and future challenges. *IEEE communications Magazine*, *49*(11), 32-39.

Mateveli, G. V., Machado, N. G., Moro, M. M., & Davis Jr, C. A. (2015, October). Taxonomia e Desafios de Recomendação para Coleta de Dados Geográficos por Cidadãos. In *SBBD (Short Papers)* (pp. 105-110).

Valente, M. T. (2020). Domain-Driven Design (DDD): Um Resumo. In *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Independente. https://engsoftmoderna.info/artigos/ddd.html

Valente, M. T. (2020). O que é uma Arquitetura Hexagonal? In *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Independente. https://engsoftmoderna.info/artigos/arquitetura-hexagonal.html

Torres, A., Galante, R., Pimenta, M. S., & Martins, A. J. B. (2017). Twenty years of object-relational mapping: A survey on patterns, solutions, and their implications on application design. *information and software technology*, *82*, 1-18.

Chow, T. L. (2013). *Classical mechanics*. CRC press.

Carrera, F., Guerin, S., & Thorp, J. B. (2013). By the people, for the people: The crowdsourcing of" STREETBUMP": An automatic pothole mapping app. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *40*, 19-23.

Qt Company. (2024). Signals & Slots. Qt Documentation. https://doc.qt.io/qt-6/signalsandslots.html