UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE COMPUTAÇÃO

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DANIEL ADLER LEVACOV

Analisaê: um ambiente analítico para o sistema de caronas universitário Caronaê

RIO DE JANEIRO

2025

DANIEL ADLER LEVACOV

Analisaê: um ambiente analítico para o sistema de caronas universitário Caronaê

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Maria Luiza Machado Campos

Co-orientadora: Profa. Vivian dos Santos Silva

RIO DE JANEIRO

2025

CIP - Catalogação na Publicação

L655a

Levacov, Daniel Adler
Analisaê: um ambiente analítico para o sistema de caronas universitário Caronaê / Daniel Adler Levacov. -- Rio de Janeiro, 2025.
130 f.

Orientadora: Maria Luiza Machado Campos.
Coorientadora: Vivian dos Santos Silva.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2025.

1. Data warehouse. 2. Modelagem dimensional. 3. ETL. 4. Business intelligence. 5. Caronaê. I. Campos, Maria Luiza Machado, orient. II. Silva, Vivian dos Santos, coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

DANIEL ADLER LEVACOV

Analisaê: um ambiente analítico para o sistema de caronas universitário Caronaê

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 10 de dezembro de 2025.

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
MARIA LUIZA MACHADO CAMPOS
Data: 23/03/2026 11:56:33-0300
Verifique em https://validar.iti.gov.br

_____

Maria Luiza Machado Campos, Ph.D. (UFRJ)

gov.br Documento assinado digitalmente
VIVIAN DOS SANTOS SILVA
Data: 20/03/2026 17:16:25-0300
Verifique em https://validar.iti.gov.br

_____

Vivian dos Santos Silva, Ph.D. (UFRJ)

gov.br Documento assinado digitalmente
MARCIO DE ALMEIDA D AGOSTO
Data: 25/03/2026 12:40:41-0300
Verifique em https://validar.iti.gov.br

_____

Marcio de Almeida D'Agosto, D.Sc. (UFRJ)

gov.br Documento assinado digitalmente
SERGIO MANUEL SERRA DA CRUZ
Data: 25/03/2026 15:32:49-0300
Verifique em https://validar.iti.gov.br

_____

Sérgio Manuel Serra da Cruz, D.Sc. (UFRJ)

Às pessoas que entendem que nada é óbvio.

AGRADECIMENTOS

Queria agradecer primeiramente à minha mãe, Judith, pelos incontáveis conselhos, por sempre conseguir me motivar mesmo em meus momentos de menor ânimo. Conversar contigo sempre me faz perceber que os meus problemas não são assim tão complicados quanto parecem, e que por mais que a dificuldade faça parte da vida adulta, não significa que não dá para curtir o caminho.

Ao meu pai, Arnaldo, pela cobrança e preocupação constante com a minha carreira e meu rumo no mundo, sei que você se importa profundamente e quer mais que tudo que eu tenha sucesso e uma vida tranquila. Queria que você pudesse estar aqui para ver tudo em pessoa, mas sei que você está torcendo por mim de longe, mais do que ninguém.

À minha irmã, Mariana, por seu pragmatismo e a capacidade de me fazer perceber quando eu estou sendo dramático demais ou relaxado demais. Você genuinamente me ajuda a focar nos meus objetivos e eu espero que você saiba que eu tenho você como um grande exemplo de disciplina. Espero que sempre estejamos juntos para poder dizer *"Eu não bebo Schin, só bebo Skol"*.

Às minhas orientadoras Maria Luiza Machado Campos e Vivian dos Santos Silva, pela eterna paciência, e por terem o cuidado de garantir que esse projeto virasse a sua melhor versão possível. Vocês sempre me aconselharam com muito carinho e atenção, e me sinto muito grato pela dedicação de vocês a este trabalho.

Ao Thiago Taubman, meu primeiro amigo e meu irmão, por me fazer perceber que na maior parte do tempo as minhas paranoias são só isso: paranoias; por me aconselhar com calma e sabedoria daqui ou do outro lado do mundo, e por me arrancar risadas que rompem a barreira do espaço-tempo.

Ao Miguel Alvarenga, por ser meu parceiro em tudo na vida e estar comigo em todos os momentos independente de qualquer coisa. Estar com você para mim é estar em casa, e ainda é muito estranho não ter essa comunicação quase telepática todos os dias.

Ao Arthur Py, pelas horas de chamada nas quais você me aconselhou, sempre com muita calma e parceria. Você é minha alma gêmea intelectual e sem você eu não vivo. Estou contando os dias para sua chegada, e como eu já disse antes, o Japão é logo ali.

Ao Lucas Tanaka, por ser o meu Jake, sempre me trazendo para a realidade quando a minha cabeça começa a espiralar. Eu me sinto extremamente contente em ter me aproximado ainda mais de você nos últimos anos, e espero que você saiba que eu sou verdadeiramente um dos seus maiores fãs.

Ao Eric Tremsky, por sempre me ouvir com tanta atenção e estar genuinamente interessado em me ajudar a resolver qualquer problema que me deixa sem dormir. Seja do Quirguistão ou do Rio, você me lembra que a vida é uma só, e que se não for pra fazer da forma mais genuína, é melhor nem fazer.

Ao Joseph Chueke, por me incentivar a ver Vinland Saga, e por todos os conselhos de vida que, por mais que eu não entenda 100% das vezes, me ajudam a seguir em frente.

Ao Marcelo Campanelli, por ser um amigo de conversas profundas e de saídas agitadas, de coração enorme e ouvido sempre aberto. Obrigado por me mostrar Niterói e Itamonte, e por ser um parceiro para qualquer hora.

Ao Ronald Albert, por sempre ser um confidente leal, e me ajudar a elaborar meus pensamentos em conversa. Eu aprecio muito a sua forma única de ver o mundo, e gosto muito de ouvir o que você pensa sobre as coisas da vida.

A Miguel Herman, Miguel Kac, David Morelenbaum e Guilherme Rozenblat, a eterna banda, por sempre estarem ao meu lado. Estar com vocês é rir, jogar conversa fora, relaxar depois de uma semana difícil, relembrar o passado, discutir o presente e ponderar sobre o futuro. Por mais que cada um de nós tenha seu caminho distinto, sei que em vocês eu tenho companheiros para a vida.

À Tatiana Pines, por me proporcionar uma amizade leal, um ombro amigo para desabafar, e sempre me ajudar a refletir nos momentos de transição da vida, onde sinto que tudo está mudando e não sei muito bem para onde ir.

Ao Vitor Gutlerner, por ter escutado meus conselhos e entendido de onde eles vinham, e pela incrível força que você tem demonstrado de superar barreiras erguidas há muito tempo. Como eu já te disse, eu sempre vou estar aqui torcendo por você e te apoiando.

À Beatriz Vogt, por sempre demonstrar carinho do seu jeito tão sensível e por ser uma parceira de conselhos, risadas, viagens e loucuras. Ser da galera contigo é bom demais!

À Maria Eduarda Oswaldo Cruz, por sempre se preocupar comigo, mesmo quando não nos vemos muito, e sempre me aconselhar com sua calma característica. Eu te vejo e sei que você me vê.

Ao João Pinheiro, por ser um incrível parceiro de aventuras e bobeiras e por sempre me levantar profissionalmente. Saiba que qualquer oportunidade que eu tiver de te retribuir, eu irei.

À Natasha Costa, que me instiga a ser inquieto, que colore meu dia a dia com seu jeito único e fascinante de ser, de ver o mundo e interagir com ele. Você me inspira a tornar em realidade o potencial que eu sei que existe em mim, porém muitas vezes esqueço. Obrigado

pelos conselhos e (principalmente) pelas bobeiras, sem as quais eu seria tão triste quanto nosso querido Ale Sater.

Dude, sucking at something is the first
step towards being sorta good at something.

Jake the Dog

RESUMO

O presente trabalho tem como objetivo principal aprimorar a capacidade analítica do projeto Caronaê, um sistema universitário de caronas compartilhadas da Universidade Federal do Rio de Janeiro (UFRJ), por meio da concepção e implementação de um ambiente analítico dedicado. Inicialmente, o projeto utilizava consultas *Structured Query Language* (SQL) executadas diretamente no seu banco de dados transacional, o que se mostrava ineficiente para análises aprofundadas devido à alta normalização e à ausência de tratamento de dados. Como método, adota-se a metodologia de modelagem dimensional de Ralph Kimball para a construção de um *Data Warehouse* (DW), que utiliza o esquema estrela para otimizar o desempenho analítico. Este DW recebe os dados do ambiente operacional através de um processo de *Extract, Transform, Load* (ETL), desenvolvido em Python, que padroniza e transforma os dados, resolvendo inconsistências referenciais. Um painel gerencial interativo, criado na plataforma de *Self-Service BI* Tableau, se liga ao DW, permitindo a execução de operações *Online Analytical Processing* (OLAP) sobre os dados. Os resultados obtidos demonstram que a implementação do novo ambiente analítico proporciona uma visão mais estratégica e intuitiva dos dados, na forma de um painel gerencial com múltiplas visualizações por página e filtros interativos. Além disso, a nova infraestrutura garante a integridade dos dados e a eficiência do banco transacional, que não corre o risco de ser sobrecarregado com consultas analíticas complexas. O estudo conclui que o desenvolvimento de um ambiente analítico é uma resposta estratégica às necessidades de inteligência de negócio do Caronaê, oferecendo maior robustez, integridade e flexibilidade para a tomada de decisões sobre o projeto.

Palavras-chave: *data warehouse*; modelagem dimensional; ETL; *business intelligence*; Caronaê; Tableau.

ABSTRACT

This work aims to enhance the analytical capacity of the Caronaê project, a university carpooling system at the Federal University of Rio de Janeiro (UFRJ), through the design and implementation of a dedicated analytical environment. The project originally utilized Structured Query Language (SQL) queries executed directly on its transactional database, which proved inefficient for in-depth analysis due to high normalization and a lack of data treatment. The method employs Ralph Kimball's dimensional modeling methodology to construct a Data Warehouse (DW), which uses the star schema to optimize analytical performance. An Extract, Transform, Load (ETL) process, developed in Python, feeds the DW with data from the operational environment, standardizing and transforming the information while resolving referential inconsistencies. An interactive managerial dashboard, built on the Tableau Self-Service BI platform, connects directly to the DW, enabling the execution of Online Analytical Processing (OLAP) operations on the data. The results demonstrate that the implementation of the new analytical environment provides a more strategic and intuitive view of the data, manifested as a managerial panel featuring multiple visualizations per page and interactive filters. Moreover, the new infrastructure guarantees data integrity and maintains the efficiency of the transactional database by preventing it from overloading from complex analytical queries. This study concludes that developing a dedicated analytical environment is a strategic response to Caronaê's business intelligence needs, offering increased robustness, integrity, and flexibility for project decision-making.

Keywords: data warehouse; dimensional modeling; ETL; business intelligence; Caronaê; Tableau.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

SGBD – Sistema Gerenciador de Banco de Dados

DW – *Data Warehouse*

DM – *Data Mart*

BI – *Business Intelligence*

ETL – *Extract, Transform, Load*

OLTP – *Online Transaction Processing*

OLAP – *Online Analytical Processing*

SQL – *Structured Query Language*

UFRJ – Universidade Federal do Rio de Janeiro

SUMÁRIO

1 INTRODUÇÃO

Ir e voltar da universidade, algo para muitos corriqueiro, pode ser uma tarefa árdua para alguns. Na UFRJ, em específico no campus Cidade Universitária, distante dos principais bairros da cidade do Rio de Janeiro, essa questão configura um sério problema. Enquanto alguns membros da comunidade acadêmica realizam seus trajetos diários de carro, a maioria sem outros passageiros, outros dependem de um sistema de transporte público que, no Rio de Janeiro, é ainda bastante complicado e penoso. Nesse contexto, um grupo de jovens viu nessa dificuldade uma oportunidade. Como descrito por Meyer (2019), o projeto Caronaê[1][2] surge em 2014, fundado por alunos da graduação deste campus, como um facilitador para o transporte da comunidade acadêmica. Essa percepção de uma lacuna social e de mobilidade impulsionou a ideia de um sistema que pudesse conectar essas pessoas, permitindo que aqueles que dependiam do transporte público pudessem pegar carona com quem já estava se deslocando na mesma direção.

Em 2016, o aplicativo Caronaê é lançado. Através dele, membros da comunidade acadêmica poderiam oferecer e pegar caronas de forma segura e eficiente. Só no primeiro ano, foram mais de 46 mil caronas criadas e mais de 12 mil usuários cadastrados, facilitando a jornada de alunos, professores e funcionários da Cidade Universitária da UFRJ. Esse amplo uso gerou um grande volume de dados relevantes sobre o uso do aplicativo, que, embora valiosos, impuseram um desafio crítico: a dificuldade de realizar análises complexas de forma eficiente e segura.

Diante dessa lacuna, o presente trabalho busca estabelecer uma arquitetura de dados robusta e eficiente para o projeto através da criação de um ambiente analítico moderno. Para alcançar esse propósito, definem-se os seguintes objetivos:

- Criação de um data warehouse, um banco de dados que utiliza modelagem dimensional e é pensado para a frente analítica;
- Implementação de um processo de ETL (Extração, Transformação e Carga) para popular o data warehouse com os dados do ambiente operacional, incluindo a limpeza e tratamento dos dados para identificar e corrigir possíveis erros referenciais e padrões que fujam às regras de negócio;

[1] https://caronae.org
[2] https://www.instagram.com/redecaronae

- Desenvolver um painel gerencial que permita a realização de operações OLAP (*Online Analytical Processing*) sobre os dados do Caronaê, aproveitando a facilidade analítica característica do data warehouse, de forma a dar suporte à tomada de decisão da equipe do projeto.

Dessa forma, o trabalho não apenas aborda limitações técnicas de desempenho, mas estabelece uma base sólida para a continuidade e escalabilidade da frente de dados do Caronaê. A seguir, detalham-se os fatores técnicos e contextuais que motivaram esta abordagem.

1.1 MOTIVAÇÃO

Desde seu lançamento, o aplicativo do Caronaê tinha seus dados armazenados em um banco de dados relacional. Esse banco, de natureza transacional, servia de apoio ao funcionamento do aplicativo e, ocasionalmente, era explorado para algumas análises básicas sobre seu uso. A frente de dados do projeto estava mais empenhada no aspecto operacional, dado o sucesso do aplicativo, do que em realizar análises mais detalhadas sobre os dados.

Em 2020, após alguns anos de funcionamento, pelo advento da pandemia de COVID-19 e a consequente falta de viagens para a universidade, o aplicativo foi retirado do ar. O projeto então se reformulou, trazendo em 2021 novos membros para evoluir o aplicativo e também para analisar os dados dos quase quatro anos em que esteve funcionando, complementando ainda a documentação do projeto como um todo. Com isso, um estudo foi conduzido, trazendo pela primeira vez recortes analíticos mais elaborados para aqueles dados.

Esta primeira solução analítica, desenvolvida em 2021 pelo autor deste trabalho, consistia em realizar consultas SQL (*Structured Query Language*) diretamente no banco de dados transacional, hospedado localmente no PostgreSQL[3], e utilizar o Microsoft Power BI[4] para gerar visualizações. Embora essa abordagem tenha sido funcional para as análises iniciais, ela apresenta limitações inerentes significativas. Bancos de dados transacionais (componentes principais de um sistema OLTP - *Online Transaction Processing*) são otimizados para o processamento eficiente de transações de negócio. Eles são projetados para operações de escrita rápidas e para manter a integridade dos dados através de rigorosas regras de normalização, o que pode resultar em uma grande quantidade de tabelas interconectadas.

[3] https://www.postgresql.org
[4] https://www.microsoft.com/pt-br/power-platform/products/power-bi

No entanto, essa mesma normalização, que é crucial para as operações do dia a dia, torna o banco transacional pouco adequado para fins de *Business Intelligence* (BI). Consultas analíticas, que frequentemente envolvem agregações complexas e a união de múltiplas tabelas (JOINs), consomem muitos recursos e podem piorar significativamente o desempenho das operações em tempo real do sistema. A estrutura altamente normalizada exige mais operações de JOIN para obter uma visão completa dos dados, tornando as consultas mais lentas e difíceis de executar. Além disso, os dados brutos de um ambiente transacional podem não estar limpos, padronizados ou consistentes, e a ausência de uma etapa de transformação dedicada pode resultar em relatórios com informações de baixa qualidade, comprometendo a confiabilidade das análises.

Ademais, alguns problemas identificados na modelagem do banco transacional do Caronaê evidenciam a necessidade de mudança e padronização. Alguns atributos, por exemplo, referenciam conceitos de outras tabelas sem a utilização de chaves estrangeiras, comprometendo a integridade referencial dos dados. Isso faz com que sejam necessárias lógicas complexas nas consultas ou no processo de extração, aumentando a probabilidade de inconsistências. Além disso, foram identificados outros problemas, como colunas com nomes pouco descritivos ou até mesmo incorretos, e falhas de modelagem que configuram barreiras para a escalabilidade do projeto.

Evidencia-se, então, a necessidade da criação de um banco de dados e um ambiente analítico mais próprio para análises de negócio, apoiado por um processo de extração, transformação e carga (ETL, do inglês - *Extract, Transform, Load*) robusto.

1.2 METODOLOGIA

A solução analítica proposta no presente trabalho será constituída, primeiramente, por um modelo fundamentado nos princípios da modelagem dimensional, capitaneada na década de 90 por Ralph Kimball. O data warehouse criado seguirá a metodologia do esquema estrela, consolidada por Kimball (1998, 2002). Essa metodologia descreve modelos com uma ou mais tabelas fato centrais, modelando eventos cruciais para o processo de negócio, e diversas tabelas dimensão que as rodeiam, com informações descritivas sobre diferentes aspectos do negócio, que caracterizam os eventos das tabelas fato. A construção do data warehouse será inspirada nas normas e processos descritos por Barbieri (2011).

A alimentação deste data warehouse será feita a partir de um processo de ETL dos dados do banco transacional, durante o qual os dados serão tratados para a garantia de

qualidade e conformidade com as regras de negócio e transformados para se encaixarem no esquema estrela supracitado.

Além disso, será desenvolvido um painel gerencial em uma plataforma de Self-Service BI, para que a equipe Caronaê possa tomar decisões estratégicas utilizando como insumo análises holísticas dos dados do aplicativo.

Este trabalho, além da introdução, encontra-se estruturado da seguinte forma: no capítulo 2, será detalhada a fundamentação teórica necessária para a execução desta empreitada. O capítulo 3 contextualizará a história do projeto e irá delinear o funcionamento do aplicativo que esteve no ar até 2020 e o modelo do banco transacional. No capítulo 4, será examinada a solução analítica desenvolvida pelo autor deste trabalho em 2021, que se baseava em consultas executadas diretamente no banco transacional. No capítulo 5, será detalhada a nova solução analítica. Nesta etapa, serão delineados o modelo do data warehouse e o processo de ETL e será apresentado o painel gerencial interativo. Por fim, no capítulo 6, serão sintetizadas as considerações finais e serão discutidas as conclusões obtidas.

2 EMBASAMENTO TEÓRICO

Nesta seção é estabelecida a base teórica para o trabalho, cobrindo os conceitos principais empregados na construção do data warehouse, do processo de ETL e do painel gerencial.

2.1 DADOS NO AMBIENTE TRANSACIONAL

O ambiente transacional de uma instituição é o local onde os dados operacionais são gerados e processados. Sua função primordial é registrar e gerir as atividades diárias do negócio em tempo real, como transações de vendas, atualização de estoque ou registros de clientes.

Para atingir a eficiência e a confiabilidade necessárias para essas operações, a estrutura de dados desses sistemas é baseada no modelo relacional, um paradigma que se tornou o padrão na indústria desde a sua formalização na década de 1970.

2.1.1 Modelo Relacional

O modelo relacional é um modelo de dados para gerenciamento de bancos de dados baseado na teoria matemática de conjuntos. Proposto por Edgar F. Codd (1970) em 1970, sua principal característica é a organização dos dados em tabelas bidimensionais, que Codd denominou de "relações". Cada tabela é composta por colunas, que representam os atributos (ou campos), e linhas, que representam os registros (ou tuplas).

A força do modelo relacional reside na sua capacidade de estabelecer relacionamentos entre as tabelas através de chaves. Uma chave primária é um atributo (ou conjunto de atributos) que identifica de forma única cada registro em uma tabela. Uma chave estrangeira, por sua vez, é um atributo em uma tabela que se refere à chave primária de outra tabela, criando um elo entre elas. Essa interconexão permite a representação de estruturas complexas do mundo real de forma organizada e coesa, garantindo a integridade referencial dos dados.

A adoção do modelo relacional trouxe diversas vantagens, como a redução da redundância de dados, a garantia de integridade e consistência e a capacidade de realizar consultas complexas e flexíveis usando a linguagem SQL (*Structured Query Language*). Essas características o tornaram o alicerce fundamental para o desenvolvimento de sistemas transacionais, que priorizam a precisão e a confiabilidade em suas operações.

2.1.2 Normalização

Para que se adentre nas estruturas que permitirão o ambiente analítico, primeiro é importante entender um conceito fundamental introduzido por Codd (1970, 1971) como parte de seu modelo relacional: a normalização. A normalização de banco de dados, segundo Codd, é o processo de estruturar um banco de dados relacional de acordo com uma série das chamadas formas normais, a fim de reduzir a redundância e melhorar a integridade dos dados.

A normalização envolve a organização das colunas (atributos) e tabelas (relações) de um banco de dados para garantir que suas dependências sejam devidamente aplicadas por restrições de integridade. Isso é feito aplicando-se algumas regras formais, seja por um processo de síntese (criando um novo design de banco de dados) ou decomposição (melhorando um design de banco de dados existente).

O resultado do processo de normalização é um modelo fragmentado em um grande número de tabelas que não necessariamente reflete a visão do usuário sobre os dados, mas garante consistência e eficiência.

2.2 DADOS NO AMBIENTE ANALÍTICO

Com o avanço da tecnologia e a crescente complexidade dos negócios na década de 80, criou-se a necessidade de uma nova forma de gerenciar e utilizar a informação. O ambiente transacional, otimizado para a velocidade e a integridade das operações diárias, mostrou-se inadequado para a análise estratégica, que exige vastas coleções de dados históricos e a capacidade de realizar consultas complexas.

Diante desse desafio, desenvolveram-se arquiteturas e metodologias dedicadas a transformar dados brutos em inteligência acionável. Esta seção explora o surgimento desse novo paradigma, a evolução do *Business Intelligence*, e a criação dos conceitos de data warehouses e data marts, modelos de bancos especificamente projetados para a análise de dados. Além disso, é introduzido o modelo dimensional de modelagem de bancos de dados, assim como suas principais abordagens e funcionalidades.

2.2.1 *Business Intelligence* (BI)

No fim da década de 80, uma ideia seminal foi introduzida propriamente ao mundo: a inteligência de negócio, ou *business intelligence*. Popularizado a partir de 1989 pelo Gartner Group, o termo *Business Intelligence* (BI) foi definido à época por seu analista Howard

Dresner como "conceitos e métodos para melhorar a tomada de decisões empresariais usando sistemas de suporte baseados em fatos" (POWER, 2007 *apud* DRESNER 1989).

O termo já havia sido utilizado anteriormente, como por Hans-Peter Luhn, pesquisador da IBM que, em 1958, publicou o artigo *A Business Intelligence System*, em que propunha que sistemas automatizados poderiam ser capazes de analisar e distribuir informações de acordo com os interesses dos tomadores de decisão dentro das organizações, como descreve Stevens (2018).

Porém, foi apenas na virada da década de 80 para 90 que o conceito de BI pôde ser aplicado em larga escala, impulsionado pela maturidade dos bancos de dados relacionais e o desenvolvimento de novas arquiteturas e ferramentas para análise de dados.

2.2.2 Data Warehouse e Data Marts

À medida que as empresas acumulavam vastos volumes de dados em seus sistemas operacionais, a necessidade de extrair inteligência de negócio para análise e tomada de decisão estratégica tornou-se mais urgente. Foi nesse contexto que o conceito de data warehouse (DW) ganhou força.

Em 1988, Devlin e Murphy publicaram um artigo introduzindo o termo *"business data warehouse"*, uma nova forma de modelar dados especialmente pensada para análise. Posteriormente, Bill Inmon (2002) popularizou o conceito de data warehouse como um repositório centralizado de dados, reunindo informações correntes e históricas de diversas fontes operacionais. Seu projeto era especificamente otimizado para a análise e a criação de relatórios, diferenciando-se da natureza transacional dos sistemas do dia-a-dia.

Um data warehouse pode ser subdividido em partes menores e mais especializadas, chamados data marts. Um data mart é um subconjunto de dados de um data warehouse que é orientado a um assunto ou departamento de negócio específico (como vendas, marketing, finanças) (INMON, 2002). Ele serve a um grupo de usuários mais restrito e focado, oferecendo uma visão mais concentrada dos dados que são relevantes para suas necessidades. Essa especialização permite que os data marts sejam mais ágeis e respondam mais rapidamente às demandas departamentais.

Data marts se conectam entre si através de dimensões compartilhadas, ou *conformed dimensions* (KIMBALL, *et al*. 2002).

2.2.3 O Modelo Dimensional

A grande revolução da década de 90 no mundo dos dados foi a popularização do modelo dimensional de data warehouses. Introduzido por Ralph Kimball, o mesmo o descreveu da seguinte forma: “um modelo dimensional contém as mesmas informações que um modelo Entidade-Relacionamento, mas empacota os dados em um formato simétrico cujos objetivos de design são a compreensibilidade pelo usuário, o desempenho da consulta e a resiliência a mudanças.” (KIMBALL, *et al*. 1998, p. 9, tradução nossa).

Este modelo se baseia na ideia de dividir os dados entre fatos e dimensões. As tabelas de fato contêm as métricas numéricas de um processo de negócio, ou seja, aquilo que se quer medir. Por exemplo, "quantidade vendida", "valor da venda", "lucro". Cada linha em uma tabela de fatos representa um evento ou uma medição específica e é conectada às dimensões através de chaves.

As tabelas de dimensões fornecem o contexto descritivo para os fatos. Elas contêm atributos que permitem "fatiar” os dados. Por exemplo, em uma dimensão de "Produto", pode-se ter informações como nome do produto, categoria, marca, etc. Assim é possível analisar as métricas na granularidade desejada.

Essa modelagem mostrou-se ideal para data warehouses por otimizar o desempenho de análises através da desnormalização e ter uma estrutura mais intuitiva para os usuários de negócio. Como descrito por Barbieri (2011), há duas operações primárias que evidenciam a utilidade dessa modelagem: o *slicing* (fatiamento) e o *dicing* (criação de subcubo).

Slicing refere-se à operação de selecionar um único valor para uma das dimensões, criando uma "fatia" dos dados. Por exemplo, analisar "Vendas para o mês de dezembro". Aqui, a dimensão "Tempo" é "fatiada" para o valor "dezembro". *Dicing* envolve a seleção de múltiplos valores de duas ou mais dimensões, resultando em um "subcubo" de dados. Por exemplo, analisar "Vendas de produtos da categoria 'Eletrônicos' no Sudeste do Brasil durante o último trimestre". Aqui, as dimensões "Produto" (categoria), "Localização" (região) e "Tempo" (trimestre) são "picadas" para filtrar os dados.

Além do *slicing* e *dicing*, a modelagem dimensional permite que os usuários explorem os dados através de outras operações OLAP fundamentais, facilitando a exploração em diferentes níveis de granularidade e em diversas fontes. Barbieri (2011) lista essas operações:

- ***Drill-Down*** **(Detalhamento) e** ***Roll-Up*** **(Agregação):** O drill-down permite mover-se de um nível de agregação mais alto para um nível de detalhe mais baixo em uma dimensão, e o roll-up faz o inverso.

- ***Drill-Across* (Atravessamento):** Permite combinar e comparar fatos de diferentes tabelas de fatos que compartilham dimensões conformadas.
- ***Drill-Through* (Navegação ao Detalhe):** Permite ir do nível de sumarização mais baixo do data mart/data warehouse para os dados transacionais que o geraram.
- ***Pivot* (Rotação):** Altera a orientação da exibição dos dados, trocando as posições das dimensões nas linhas e colunas de uma tabela ou gráfico.

Essas operações permitem que os usuários naveguem e interroguem os dados de forma flexível e profunda, transformando vastos repositórios de informações em conclusões acionáveis para o negócio.

2.2.3.1 Esquema Estrela x Esquema Floco de Neve

Duas abordagens principais para projetos de data warehouse e data mart utilizando o modelo dimensional se destacam, representando filosofias de projeto distintas: o Esquema Estrela, defendido primariamente por Ralph Kimball e o Esquema Floco de Neve, especialmente apoiado por Bill Inmon. Embora ambos sejam modelos dimensionais, eles se enraízam em arquiteturas de construção de DW fundamentalmente opostas, o que influencia diretamente o tipo de modelagem adotado.

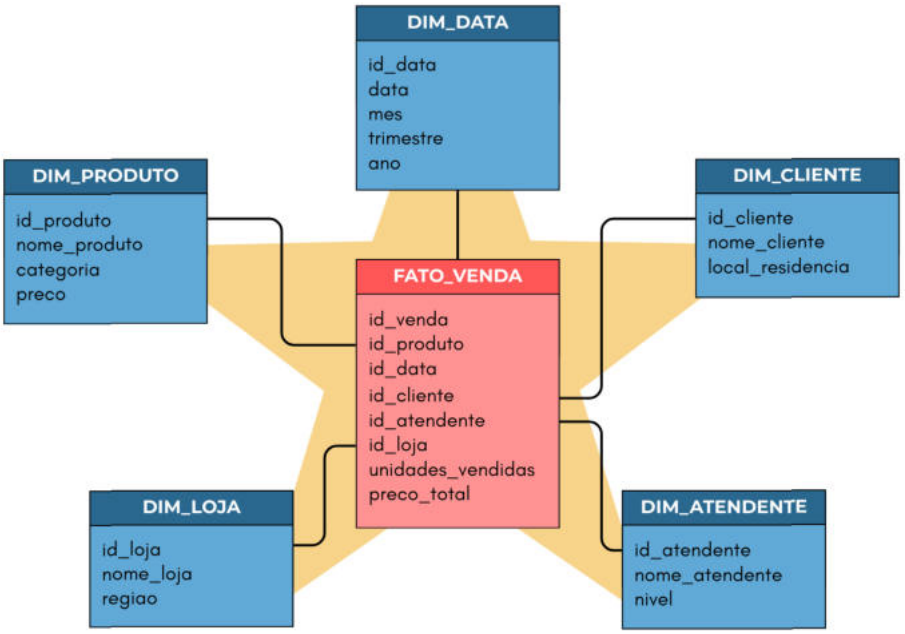
O Esquema Estrela recebe esse nome por se assemelhar visualmente a uma estrela, geralmente com uma tabela fato central cercada por várias tabelas de dimensão (KIMBALL, *et al*. 1998). Sua forma pode ser vista na figura 1, na qual está representado um modelo de banco de dados de vendas, com o fato sendo a venda em si, e com as dimensões data, produto, cliente, atendente e loja.

Nesse esquema, as tabelas de dimensão são desnormalizadas. Isso significa que elas contêm todos os atributos relevantes para aquela dimensão em uma única tabela, evitando a separação de atributos hierárquicos em tabelas distintas. Além disso, as tabelas de dimensão se conectam diretamente à tabela de fatos por meio de chaves estrangeiras, sem relações entre elas.

Por ser desnormalizado, o esquema exige menos JOINs entre tabelas para recuperar dados para a maioria das consultas analíticas. Isso resulta em um desempenho de consulta muito mais rápido, o que é crucial para ferramentas de BI e análise OLAP, que realizam consultas complexas com frequência. Naturalmente, isso aumenta a redundância de dados, porém essa concessão é importante para o funcionamento eficiente do sistema.

Essa modelagem de dados está intrinsecamente ligada à filosofia de arquitetura *bottom-up* proposta por Kimball. Nessa abordagem, os data marts, projetados para processos de negócio específicos, são as construções fundamentais. O data warehouse corporativo é, então, a integração desses data marts por meio de dimensões conformadas — dimensões com definições e chaves consistentes que permitem a união e a análise conjunta dos dados.

Figura 1: Exemplo do Esquema Estrela de Ralph Kimball, para um banco de dados de vendas



Fonte: elaboração própria

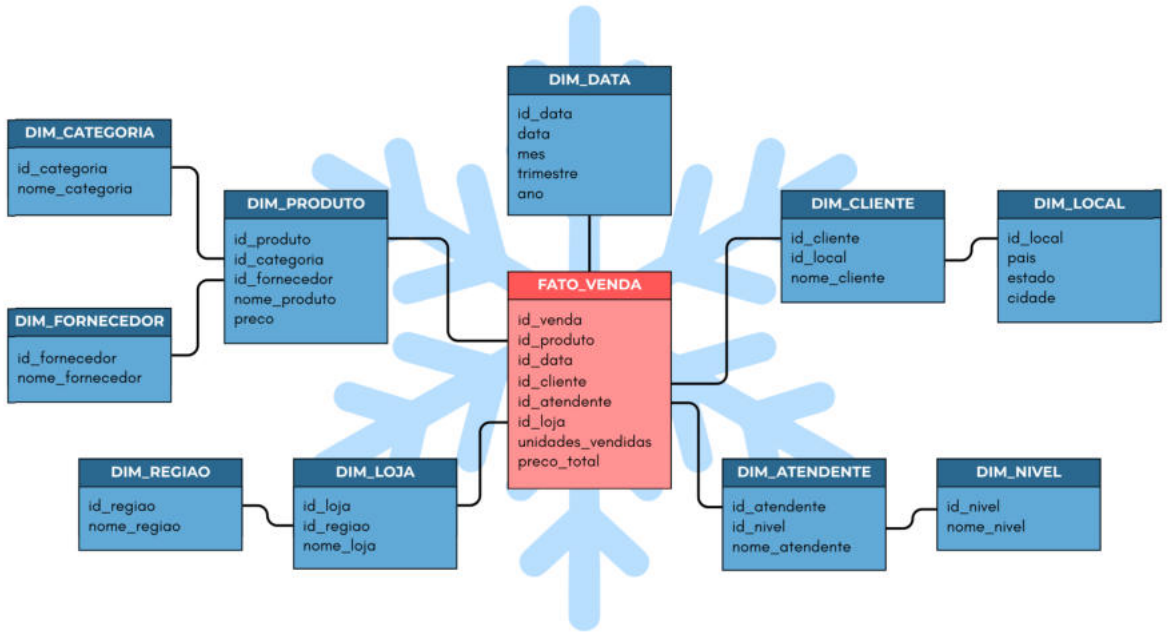
Em contrapartida, no Esquema Floco de Neve, segundo Barbieri (2011), as tabelas de dimensão são normalizadas, criando uma estrutura mais hierárquica e reduzindo a redundância. Visualmente, ele se assemelha a um floco de neve, com as tabelas de dimensão "ramificando-se" em subdimensões, como visto na figura 2, que representa um modelo em esquema floco de neve para o mesmo banco de dados representado na figura 1. Isso significa que atributos hierárquicos ou categorias dentro de uma dimensão são separados em tabelas de subdimensão. As tabelas de dimensão se conectam, então, indiretamente à tabela fato através de outras tabelas de dimensão. Na figura 2, percebe-se que há subdimensões de categoria, fornecedor, região, nível e local, ligadas às cinco dimensões que estão conectadas diretamente à tabela fato. As subdimensões contém informações que na figura 1 estão nas dimensões às quais estão conectadas.

A normalização das dimensões reduz a duplicação de dados, economizando espaço de armazenamento e melhorando a integridade dos dados, já que há menos lugares para ocorrerem inconsistências. Essa estrutura pode se provar útil para modelar hierarquias de dimensão muito complexas ou dados que se beneficiam da normalização para clareza ou manutenção.

O esquema, porém, exige mais JOINs para recuperar dados, pois as informações de uma dimensão podem estar espalhadas por várias tabelas. Isso pode levar a um desempenho de consulta mais lento. Além disso, ele é mais complexo de entender e projetar do que o esquema estrela, tanto para desenvolvedores quanto para usuários de negócio.

Essa modelagem de dados, com a priorização da normalização, se alinha à filosofia de arquitetura *top-down* proposta por Bill Inmon. Nessa abordagem, um data warehouse corporativo centralizado e altamente normalizado atua como a "fonte única da verdade" para toda a organização. Os data marts são, então, derivados desse DW corporativo para atender a necessidades específicas dos usuários de negócio.

Figura 2: Exemplo do Esquema Floco de Neve de Bill Inmon, para o mesmo banco de dados de vendas



Fonte: elaboração própria

Além disso, como pontua Barbieri (2011), os esquemas diferem na abordagem de criação de data marts. O Esquema Estrela de Kimball propõe uma abordagem *bottom-up*, onde os data marts, desenvolvidos para processos de negócio específicos, são as construções fundamentais. O data warehouse corporativo é, então, a integração desses data marts através

de dimensões conformadas — ou seja, dimensões com definições e chaves consistentes que permitem que os data marts sejam unidos e analisados em conjunto.

Já Inmon propõe com seu Esquema Floco de Neve uma abordagem *top-down*, na qual um data warehouse corporativo centralizado e altamente normalizado é a "fonte única da verdade". Os data marts são, então, derivados desse DW corporativo.

2.2.3.2 Dimensões Sucata (*Junk*)

Dentro do arcabouço teórico da modelagem dimensional, existem alguns tipos especiais de dimensões, como por exemplo dimensões que evoluem com o tempo, dimensões muitos-para-muitos, minidimensões, etc. Dessas dimensões especiais, serão detalhadas apenas duas, as que de fato serão utilizadas no trabalho: a dimensão sucata e a dimensão degenerada.

O conceito de dimensões sucata ou lixo (*junk*) na modelagem dimensional é empregado para agrupar e gerenciar eficientemente campos com características específicas, como aqueles de baixa cardinalidade, valores binários, tags, ou atributos de texto esparsos.

Esses campos não se encaixam em nenhuma das dimensões reconhecíveis, e a criação de uma dimensão individual para cada um não se justifica. Por isso, são agrupados em uma única dimensão, mesmo que não haja relação semântica entre eles.

Embora esses campos possam parecer secundários e não apresentem forte correlação entre si na tabela fato, Barbieri (2011) relembra que é crucial mantê-los como dimensões, pois são frequentemente usados como filtros em análises.

2.2.3.3 Dimensões Degeneradas

Uma dimensão degenerada, segundo Barbieri (2011), é um atributo identificador de um processo de negócio de nível superior (como o número de um pedido, transação ou nota fiscal) que é incluído diretamente na tabela fato, sem a necessidade de criar uma tabela de dimensão separada. Esse conceito é aplicado quando a granularidade da tabela fato está no nível dos itens individuais ou eventos do processo.

O principal papel da dimensão degenerada é servir como um integrador ou elemento de agrupamento dentro da tabela fato, permitindo que todos os itens pertencentes àquele identificador de nível superior sejam reunidos e analisados em conjunto. Adicionalmente, ela desempenha o papel de manter a referência a atributos de aspecto transacional. Ao incluir o identificador diretamente no fato, a dimensão degenerada atua como um ponto de acesso para

buscas detalhadas nos sistemas OLTP de origem, sendo frequentemente utilizada para a operação de *drill-through*.

2.2.3.4 Chaves Substitutas (*Surrogate Keys*) em Modelagem Dimensional

Segundo Barbieri (2011), o uso de chaves substitutas (*surrogate keys*) — também conhecidas como chaves artificiais — é altamente recomendável nos projetos de data warehouses e data marts. Essas chaves são campos sequenciais gerados e administrados pelo ambiente analítico, sem qualquer valor semântico de negócio (como nome ou número de matrícula).

A adoção de chaves substitutas é preferível às chaves naturais (com sentido de negócio) devido à maior estabilidade que oferecem. Chaves naturais podem apresentar problemas de unicidade e instabilidade. Além disso, entidades ou eventos podem não possuir chaves naturais para identificação.

As chaves substitutas servem primariamente como elos entre as tabelas fato e dimensão, enquanto os usuários interagem com campos semânticos para suas análises.

2.3 DO TRANSACIONAL AO ANALÍTICO

A transição de um ambiente puramente operacional para um focado em inteligência de negócio não é apenas uma mudança de ferramentas, mas uma transformação de paradigma. Ela representa o movimento de sistemas otimizados para registrar dados (transacionais) para sistemas otimizados para analisar dados (analíticos). Esta seção detalha as diferenças fundamentais entre esses dois mundos, explicando como os processos e tecnologias, como o OLTP e o OLAP, se complementam e como o processo de ETL atua como a ponte essencial para mover os dados de um ambiente para o outro de forma fluida e segura.

2.3.1 OLTP x OLAP

Pela distinção natural entre as finalidades dos bancos de dados, formalizaram-se duas categorias principais de operações: OLTP (*Online Transaction Processing*) e OLAP (*Online Analytical Processing*).

Os bancos de dados operacionais, adequados para registrar rapidamente as informações do dia-a-dia do negócio – como transações de vendas ou atualizações de estoque – são a base de para a execução de operações OLTP (REDDY, *et al.* 2010). Seu propósito

principal é o processamento eficiente de transações, garantindo a atomicidade, consistência, isolamento e durabilidade dos dados, e permitindo operações de escrita e leitura rápidas. Por essa razão, esses bancos de dados, em geral, seguem rigorosas regras de normalização. Essa normalização (que pode levar a dezenas ou até centenas de tabelas interconectadas) é fundamental para diminuir a redundância de dados e garantir a integridade das informações, minimizando inconsistências e erros nas operações de escrita.

Em contraste, os data warehouses e data marts são otimizados para a realização de operações OLAP, operações analíticas muito mais complexas que as operacionais. Essas operações são descritas em detalhe mais adiante. Esses DWs e DMs são mais focados na análise e consulta de dados compostos por valores sumarizados, calculados e integrados de várias fontes.

Diferentemente dos bancos transacionais, os data warehouses e data marts têm a liberdade de serem menos normalizados. Essa desnormalização intencional visa aumentar significativamente a performance das consultas, pois a redução do número de tabelas e a simplificação das estruturas diminuem a necessidade de operações complexas como JOINs.

Como caracteriza Barbieri: “O termo OLAP (*On-Line Analytical Processing*), hoje muito difundido, traduzido para processamento analítico on-line, representa essa característica de trabalhar os dados com operadores dimensionais, possibilitando uma forma múltipla e combinada de análise [...]” (BARBIERI, 2011, p. 109).

Outro ponto crucial é a forma como os dados são carregados e atualizados nesses bancos. Nos bancos transacionais, a atualização é contínua e ocorre campo a campo, refletindo as transações individuais à medida que acontecem em tempo real. Isso exige que o sistema gerencie um grande volume de operações de escrita (inserção, atualização, exclusão) de forma rápida e com garantia de integridade. Já nos data warehouses e data marts, os dados são geralmente carregados em lotes e praticamente não são atualizados após o carregamento inicial, focando na persistência histórica. Isso significa que os riscos de inconsistência de dados, que poderiam ser introduzidos pela redundância inerente à desnormalização, são minimizados, pois as operações de escrita (atualizações e exclusões) são raras e controladas.

2.3.2 ETL

Para que os dados das diversas fontes operacionais pudessem ser consolidados e preparados para análise no data warehouse, um processo crucial foi desenvolvido e formalizado: o ETL . Este processo de três etapas é o coração da construção e manutenção de um DW:

Extração (Extract): Refere-se à coleta de dados de diversas fontes operacionais, que podem estar em formatos e estruturas variadas.

Transformação (Transform): É a etapa onde os dados extraídos são limpos, padronizados, validados, combinados e convertidos para o formato e a estrutura necessários para o data warehouse, garantindo a qualidade e a consistência das informações. Inclui a resolução de inconsistências, o tratamento de valores nulos e a aplicação de regras de negócio, entre outros.

Carga (Load): Consiste em carregar os dados transformados no data warehouse (ou data mart), geralmente de forma incremental, garantindo que o repositório analítico seja atualizado periodicamente com novas informações.

Segundo Barbieri (2011), as camadas de ETL complementam a arquitetura de data warehouses e data marts. Este processo é essencial para preencher a lacuna entre os bancos de dados transacionais e as necessidades analíticas, permitindo a consolidação de informações para inteligência de negócio.

2.3.3 Novas Necessidades Analíticas: Self-Service BI

Com a virada do milênio e a explosão do *Big Data*, os dados aumentaram drasticamente em volume, variedade e velocidade, três características típicas deste tipo de dados, desafiando as arquiteturas de DW convencionais. Esse cenário gerou uma demanda ainda maior por ferramentas capazes de analisar dados em larga escala e de forma mais ágil.

Nesse contexto, as ferramentas de *Self-Service BI* emergiram como uma forma de democratizar o acesso à análise de dados para usuários de negócio. Plataformas como o Tableau[5] e o Microsoft Power BI[6] se destacaram por sua interface intuitiva e capacidade de visualização avançada, permitindo que analistas e gerentes explorem dados de forma autônoma (daí o nome, auto-serviço, se traduzido para o português), sem a necessidade de um profundo conhecimento em SQL ou da intervenção constante da equipe de TI.

Estabelecida a base teórica para o trabalho, a próxima seção detalha o que é, como surgiu e qual foi a trajetória do projeto Caronaê.

[5] https://www.tableau.com/pt-br
[6] https://www.microsoft.com/pt-br/power-platform/products/power-bi

3 O PROJETO CARONAÊ

Esta seção descreve a história do projeto, bem como detalha o funcionamento de seu aplicativo, que esteve ativo entre 2016 e 2020. Além disso, são apresentados os principais processos do negócio e uma visão geral da modelagem do banco transacional, em uso desde a criação do aplicativo.

3.1 A HISTÓRIA DO PROJETO

Como descreve Meyer (2019), o Projeto Caronaê foi idealizado em 2014 por estudantes de graduação dos cursos de Engenharia Ambiental e Arquitetura e Urbanismo no campus da Cidade Universitária da Universidade Federal do Rio de Janeiro (UFRJ). Esses estudantes identificaram um dos problemas na forma como o transporte de e para a universidade ocorria. Eles perceberam que muitas pessoas, sejam estudantes, professores ou funcionários administrativos, faziam diariamente o trajeto de carro sozinhas, sem passageiros, enquanto outros membros da comunidade acadêmica dependiam do transporte público.

Essa dinâmica, agravada pelo fato de que o campus da Cidade Universitária (o maior da UFRJ) está distante dos principais centros residenciais da cidade, levou-os a uma ideia: e se houvesse uma maneira de conectar as pessoas da comunidade acadêmica, permitindo que aqueles que precisavam utilizar o transporte público tivessem a oportunidade de pegar carona com alguém que já estivesse indo na mesma direção? Assim, eles começaram a trabalhar na ideia com base em alguns objetivos-chave:

- Facilitar o transporte de e para a universidade (inicialmente focando apenas no campus principal, Cidade Universitária);
- Oferecer um sistema eficaz de compartilhamento de carros, com uma experiência do usuário intuitiva e um modo rápido e fácil de conectar as pessoas;
- Garantir que todos os usuários fossem de fato membros da comunidade acadêmica, através de um mecanismo de verificação;
- Reduzir o trânsito intenso dentro e ao redor do campus;
- Aproximar pessoas de diferentes segmentos da universidade.

Além disso, alguns ideais foram estabelecidos para guiar o projeto em uma frente teórica:

- Conscientizar sobre o uso de caronas e suas possibilidades;
- Contribuir para a discussão sobre transportes sustentáveis;

- Expandir o projeto para além dos limites da universidade.

Os idealizadores também decidiram que o aplicativo deveria promover caronas solidárias, sem qualquer forma de pagamento envolvida, a menos que o passageiro voluntariamente desejasse colaborar com algum valor para o combustível, por exemplo, caso a carona se tornasse periódica.

O grupo então inscreveu o projeto no "Concurso Soluções Sustentáveis" organizado pelo Fundo Verde da UFRJ, e o projeto venceu na categoria Mobilidade Urbana.

Em 2015, iniciou-se a implementação do projeto, coordenada pelo professor Ronaldo Balassiano, do Programa de Engenharia de Transportes da COPPE/UFRJ. Formou-se uma equipe multidisciplinar composta por estudantes das Engenharias de Computação, Civil, Materiais, Ambiental e Transportes, além de alunos de Comunicação Social, Arquitetura e Urbanismo e Ciência da Computação.

Essa abordagem multidisciplinar permitiu que houvesse uma ampla diversidade de habilidades e experiências na equipe, resultando em entregas como aplicativos para Android e iOS, uma página web, pontos físicos de carona, desenvolvimento de identidade visual, campanhas de marketing e publicações acadêmicas.

Em 2016, o aplicativo finalmente foi lançado, inicialmente atendendo apenas o campus da Cidade Universitária, na Ilha do Fundão. Em seu primeiro ano, mais de 46 mil caronas foram criadas no aplicativo. Com isso, o modelo de carona solidária foi difundido, com os motoristas pedindo normalmente apenas uma pequena taxa para ajudar a pagar o combustível.

Em 2017, ao fazer parceria com docentes do Departamento de Ciência da Computação, a equipe do Caronaê ampliou seus esforços para disseminar a ideia do sistema de compartilhamento de caronas criando a Rede Caronaê. Ao mesmo tempo, buscou formalizar a iniciativa como uma ação de extensão, oficializando o projeto de extensão sob o mesmo nome no sistema acadêmico da UFRJ, em 2018, sob a coordenação da profa. Maria Luiza Machado Campos. A ideia era transformar o projeto em um empreendimento de código aberto, permitindo que qualquer pessoa estudasse, modificasse ou distribuísse o código, com a simples condição de mantê-lo aberto e creditar os desenvolvedores. Isso permitiria que programadores ao redor do mundo pudessem contribuir com melhorias, beneficiando todos os

usuários do sistema. Para viabilizar essa iniciativa, o código-fonte do aplicativo foi disponibilizado no GitHub[7] sob a licença GNU General Public License v3.0.

O mesmo aplicativo seria usado pela UFRJ e por outras instituições, que poderiam replicar o projeto em seus espaços, incentivando o uso de sistemas de compartilhamento de carros no mesmo modelo do Caronaê.

Em 2018, o projeto foi também expandido para o campus da Praia Vermelha, ainda na cidade do Rio de Janeiro, e em 2019 para o campus de Macaé, na região Norte do estado do Rio de Janeiro. Isto permitiria que mais membros da comunidade acadêmica da UFRJ participassem do sistema de compartilhamento de carros.

Em 2020, devido à pandemia, o aplicativo foi retirado do ar, pois ninguém estava se deslocando para os campi. Essa pausa, aliada à redução da equipe, levou o projeto a um estado de "hibernação".

Em 2021, o Caronaê firmou uma parceria com o Instituto Brasileiro de Transporte Sustentável (IBTS) para apoiar análises de dados relativos ao projeto e conduzir um novo estudo na área de emissões. Com isso, surgiu a frente de dados do Caronaê. Em novembro deste mesmo ano, foi finalizado um relatório de dados de uso do aplicativo desenvolvido pelo autor deste trabalho. Ainda neste ano, o autor deste trabalho apresentou o projeto no Festival do Conhecimento UFRJ.

Além disso, o Caronaê contribuiu para o desenvolvimento do Plano de Gestão de Logística Sustentável da Universidade (PLS-UFRJ), ajudando a pensar em como garantir que o transporte para e dentro dos campi seja sustentável e eficiente, aproveitando projetos de transporte já existentes, como o próprio Caronaê e o projeto de bicicletas Integra.

No mesmo ano, o projeto foi aprovado no edital de Projetos Especiais do Parque Tecnológico da UFRJ, recebendo financiamento para criar um novo aplicativo voltado ao público do Parque. Isso permitiu crescimento nas áreas de gestão, comunicação e desenvolvimento. No entanto, uma versão específica para o Parque não se mostrou viável, seja por conta do período em que as empresas estiveram funcionando remotamente e posteriormente de forma híbrida, mas também pela diversidade de questões colocadas para que seus funcionários se envolvessem na iniciativa. De qualquer modo, o reconhecimento do valor do projeto por parte do Parque e o financiamento recebido foram importantes para que o código geral do sistema e análises dos dados evoluíssem com a participação de alunos bolsistas dedicados ao projeto.

[7] https://github.com/caronae

Em 2022, o autor deste trabalho, com o apoio de membros do projeto, finalizou o relatório de dados sobre emissões. Nele, foi estimado o volume aproximado de emissões de gases de efeito estufa evitadas pelo compartilhamento de carros através do Caronaê durante o período em que o aplicativo esteve ativo.

Ainda em 2022, deu-se continuidade ao esforço de evolução do aplicativo para o Caronaê, com melhorias e novas funcionalidades. Um dos motivos por trás desta frente foi o objetivo de criar um único aplicativo que funcionasse tanto para iOS quanto para Android, pois até então existiam duas versões, uma para cada sistema operacional móvel.

O autor deste trabalho fez apresentações sobre seus estudos analíticos na SIAc (Semana de Integração Acadêmica da UFRJ) em 2022 e 2023, recebendo menção honrosa nas duas ocasiões. Além disso, em 2025, o autor fez uma apresentação sobre o Caronaê no estande da UFRJ no Rio Innovation Week, junto do professor Sérgio Serra. Os dois falaram sobre a história do projeto, seu momento atual e ideias para seu futuro.

3.2 O APLICATIVO

O aplicativo Caronaê consistia de duas funções principais: buscar caronas e oferecê-las. Como caronista (quem pega carona), era possível procurar caronas para ou a partir de algum campus da universidade com base em três fatores: ponto de partida, destino e data/horário. Se houvesse alguma disponível, o usuário poderia pedir para participar. Como motorista (quem oferece a carona), era possível oferecer viagens em uma data e horário específicos e aceitar ou recusar solicitações de passageiros. O aplicativo também possuía um chat integrado, permitindo que os usuários se comunicassem sem precisar entrar em outro aplicativo.

Figura 3: Tela inicial do antigo aplicativo Caronaê

Fonte: Gomes (2019)

Ao se cadastrar, o usuário acessava o aplicativo via Intranet UFRJ, como visto na figura 3. Então ele informava alguns dados, como nome, vínculo acadêmico (se é estudante de graduação, de doutorado, docente, etc.), telefone, e-mail, entre outros. Essas informações eram então armazenadas no banco de dados transacional do Caronaê, hospedado no SGBD PostgreSQL. A partir disso, o usuário já poderia interagir com o sistema de compartilhamento de caronas.

Se o usuário possuísse um carro e quisesse utilizá-lo para oferecer caronas, ele também deveria fornecer informações sobre o veículo, como marca, modelo, cor e placa. Os motoristas podiam criar tanto caronas únicas quanto recorrentes.

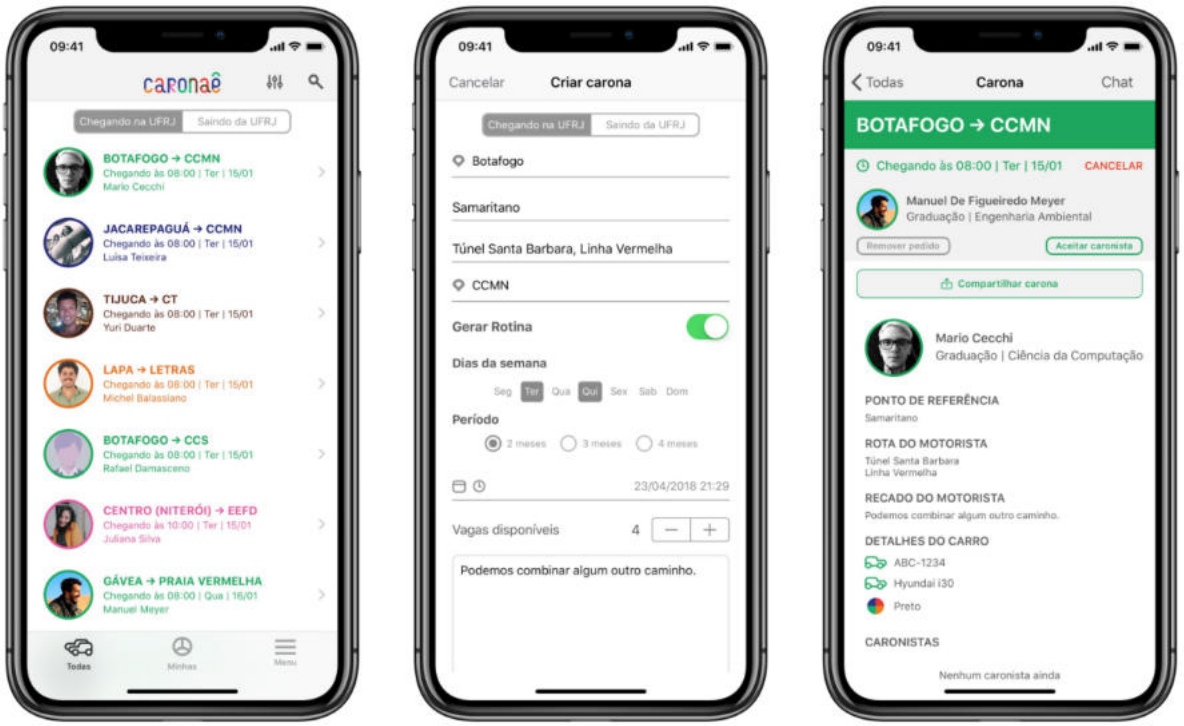
Uma carona única ocorre apenas uma vez, enquanto uma rotina consiste em várias viagens que compartilham exatamente as mesmas informações, exceto a data. Dessa forma, um motorista poderia agendar caronas para, por exemplo, todas as segundas e quartas-feiras às 8h, partindo de Botafogo e indo até o CCMN (pólo na Ilha do Fundão) até o final do semestre.

Algumas informações eram necessárias ao se criar uma carona, seja única ou recorrente, como se a viagem estava indo para o campus da universidade ou saindo dele, o

bairro de origem/destino, o polo universitário de destino/origem e o número de vagas disponíveis no carro. Se a carona fosse recorrente, eram necessárias informações extras: os dias da semana em que as caronas ocorreriam e a data final da rotina.

Na figura 4, é possível ver algumas das telas principais do antigo aplicativo, como a de busca por carona, criação de carona, e detalhes sobre a carona.

Figura 4: Telas de busca por carona, criação de carona, e de detalhes sobre a carona no antigo aplicativo Caronaê



Fonte: Gomes (2019)

3.3 AMBIENTE DE DADOS OPERACIONAL

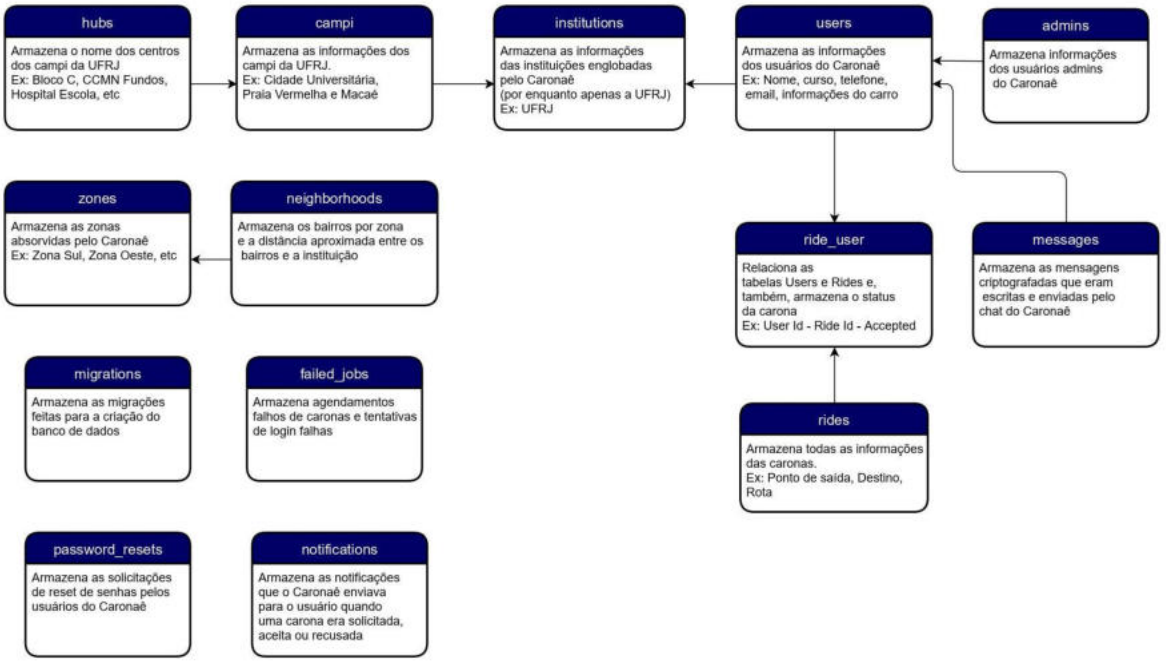
Atualmente, o projeto Caronaê tem apenas um banco de dados transacional, hospedado no PostgreSQL. Desde a desativação do aplicativo em 2020, esse banco está estático, sem novas informações sendo inseridas. Nesta subseção será detalhada a modelagem deste banco.

Através de engenharia reversa, foram gerados dois modelos lógicos. Na figura 5, é possível ver um modelo lógico simplificado, apenas com as tabelas[8] e suas descrições, e

[8] Os nomes das tabelas e colunas estão em inglês, pois a equipe que criou o banco de dados entendia que o projeto aspirava ultrapassar fronteiras, podendo potencialmente servir como base para projetos similares em universidades de outros países.

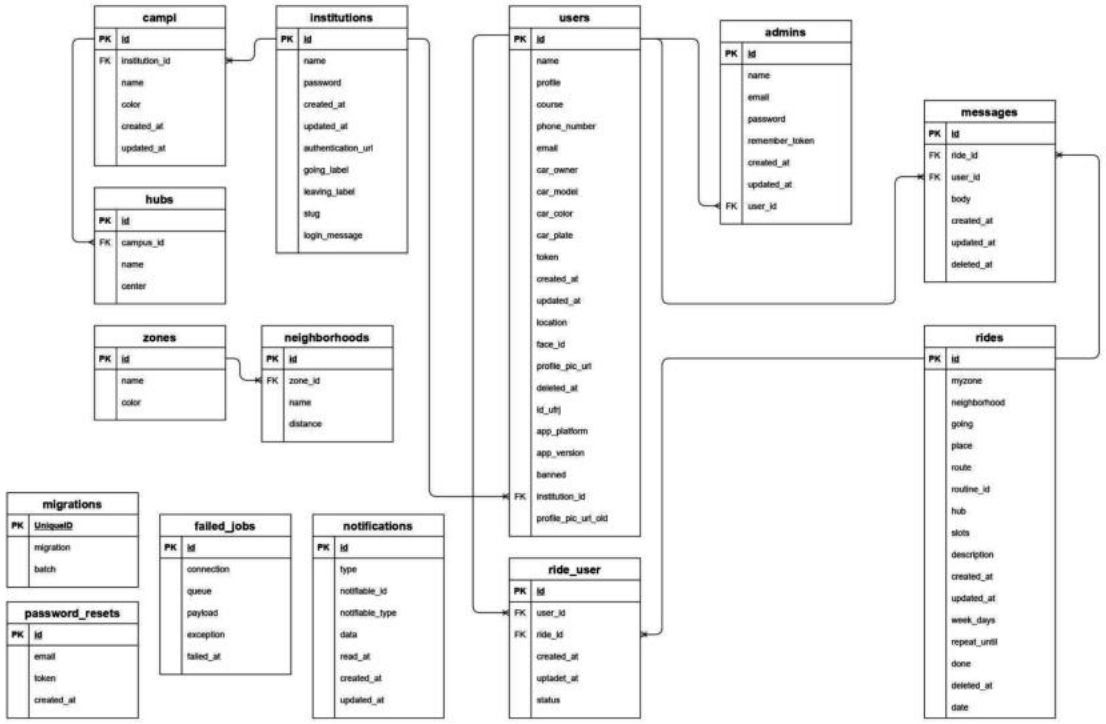
relações entre elas representadas por setas. Já na figura 6, há um modelo lógico mais detalhado, com os atributos de cada relação, identificação das chaves e setas apontando de um atributo chave para outro, ilustrando as conexões entre as tabelas com mais especificidade.

Figura 5: Modelo lógico simplificado do banco de dados transacional do Caronaê



Fonte: elaboração própria

Figura 6: Modelo lógico detalhado do banco de dados transacional do Caronaê



Fonte: elaboração própria

3.3.1 Descrição das Tabelas do Banco Transacional

A seguir são descritas brevemente as tabelas dessa modelagem transacional. Há, nessa modelagem, três tabelas centrais: **`users`**, **`rides`** e **`ride_user`**.

A tabela **`users`** armazena as informações sobre os usuários do aplicativo, como nome, vínculo acadêmico (se o usuário é da graduação, mestrado, servidor, etc.), se tem carro ou não (e as informações do carro, se tiver), etc. A tabela **`rides`** armazena as informações sobre as caronas criadas pelo aplicativo, como bairro de origem/destino, pólo universitário de origem/destino, descrição, etc. A tabela **`ride_user`** serve para relacionar as caronas com os usuários. Os registros nessa tabela podem ser de dois tipos: criação de carona ou pedido de entrada em carona.

A tabela **`ride_user`** tem três colunas principais: `ride_id` (id da carona), `user_id` (id do usuário) e `status`. Quando o registro é de criação de carona, o `user_id` sempre será o id do motorista (pessoa que criou a carona) e o *status* sempre será "*driver*". Quando é um registro de pedido de entrada na carona, o `user_id` será o id do usuário solicitante, e o *status* refletirá a situação do pedido: "*accepted*" se ele foi aceito, "*refused*" se

foi recusado, "*pending*" se não foi respondido e "*quit*" se o usuário desistiu de entrar na carona.

Além dessas, há cinco tabelas auxiliares ligadas mais diretamente às três do núcleo central. A tabela **neighborhoods** armazena as informações sobre bairros e suas respectivas distâncias até a Ilha do Fundão. Aqui constam bairros de diversas cidades. Não há informação disponível para aferir se houve um conjunto de bairros armazenados previamente ou se a tabela foi inicializada vazia. Entretanto, há indícios de que ao longo do funcionamento do aplicativo, bairros inseridos no campo de bairro das caronas que não constavam nessa tabela eram adicionados a ela. Há nessa tabela um campo `zone_id` relacionando cada bairro a sua respectiva zona da tabela `zones`. A tabela **zones** armazena as zonas de operação do aplicativo, dentro das quais estão os bairros da tabela neighborhoods. Assim como na tabela neighborhoods, não há informação disponível sobre como foi definido o escopo de zonas abrangidas, porém há indícios de que simplesmente é determinado pelos bairros presentes em neighborhoods.

A tabela **institutions** armazena informações sobre as instituições em que o Caronaê opera. Até o presente momento, a tabela contém apenas um registro correspondente à UFRJ, mas ela representa a possibilidade de expansão do projeto. A tabela **campi** armazena os *campi* onde o Caronaê funciona, relacionando-os a suas instituições na tabela institutions pelo campo `institution_id`. A tabela **hubs** armazena os pólos dos campi onde o Caronaê opera, junto com seus centros. Por exemplo, o centro "CCMN" na Cidade Universitária tem dois pólos: "CCMN: Frente" e "CCMN: Fundos". Os pólos se relacionam com os *campi* da tabela **campi** pelo campo `campus_id`.

Há também mais seis tabelas que estão mais ligadas ao lado operacional do aplicativo e não diretamente ao domínio das caronas. A tabela **admins** armazena informações dos usuários com permissão de administrador, ou seja, os membros do *staff* do Caronaê. Ela se liga à tabela **users** por meio do campo `user_id`. A tabela **messages** armazena as mensagens enviadas entre usuários, devidamente criptografadas. Ela se relaciona com a tabela users através do campo `user_id` e com a tabela **rides** (para estabelecer o contexto da carona sobre a qual a mensagem foi enviada) através do campo `ride_id`. A tabela **notifications** armazena as informações sobre as notificações enviadas para os usuários através do aplicativo. A tabela **migrations** armazena dados sobre as migrações feitas para a criação e atualização do banco de dados operacional. A tabela **failed_jobs** armazena

dados sobre agendamentos falhos de caronas e tentativas de login falhas. Por fim, a tabela **`password_resets`** armazena as solicitações de reset de senha feitas pelos usuários.

3.3.2 Descrição do Processo Caronaê

A seguir são descritos os processos que ocorriam através do aplicativo Caronaê, na versão associada aos dados analisados, do ponto de vista da alimentação do banco de dados transacional.

Quando um novo usuário fazia seu cadastro, ele/ela inseria:

- Nome
- Vínculo acadêmico (Graduação, Mestrado…)
- Curso
- Número de telefone
- E-mail
- Se tinha carro ou não (se sim inserir as três informações abaixo)
    - Modelo do carro
    - Cor do carro
    - Placa do carro
- Foto de perfil (opcional)
- CPF

e essas informações eram armazenadas na tabela **`users`**.

Quando um usuário criava uma carona única como motorista (fora de uma rotina), ele/ela devia inserir as seguintes informações, que passavam a constar na tabela **`rides`** como um novo registro:

- Zona (da tabela **`zones`**)
- Bairro (da tabela **`neighborhoods`**)
- Se estava indo ou voltando do campus
- Ponto de referência (opcional)
- Rota (opcional)
- Pólo do campus (da tabela **`hubs`**)
- Número de vagas
- Descrição da carona

Quando um usuário criava uma carona como motorista dentro de uma rotina, além das informações anteriores, ele/ela devia inserir os seguintes dados:

- Dias da semana
- Data final da rotina

A tabela **`ride_user`** armazenava pedidos referentes a uma carona específica, explicitando por qual usuário foi feito o pedido. Estes pedidos poderiam ser:

- A criação de uma carona (feita pelo motorista)
- Uma solicitação de um usuário para pegar uma carona e a subsequente resposta do motorista (se aceitou ou recusou o usuário como passageiro, ou se nunca respondeu à solicitação)

A criação da carona também era armazenada como um pedido na tabela **`ride_user`**, relacionando a carona ao usuário que a criou. O novo registro teria as seguintes colunas:

- id do motorista (da tabela **`users`**)
- id da carona (da tabela **`rides`**)
- Quando a carona foi criada
- Quando ela foi atualizada pela última vez
- *Status* do usuário que fez o pedido (no caso seria driver pois o pedido foi a criação da carona, algo que só pode vir do motorista)

Após a criação da carona, quando um usuário queria pegar a carona, sua solicitação era armazenada como pedido na tabela **`ride_user`**, onde uma nova linha era gerada com as seguintes colunas:

- id do usuário requerente (da tabela **`users`**)
- id da carona (da tabela **`rides`**)
- Quando a carona foi criada
- Quando ela foi atualizada pela última vez
- *Status* do usuário que fez o pedido
    - Antes do motorista responder ao pedido, o *status* ficava como “*pending*”

- Se o pedido fosse aceito ou recusado pelo motorista, a mesma linha era atualizada para refletir esse *status* ("*accepted*"/"*refused*")
- Se o usuário requerente desistisse da solicitação para entrar na carona, o *status* ficava como "*quit*"

Se o motorista por qualquer motivo quisesse excluir a carona, ele/ela poderia. Com isso, a linha do pedido com *status* "*driver*" relacionada à criação da carona na tabela **`ride_user`** era deletada, mas as linhas com os pedidos relacionados a ela não.

Depois da finalização da carona, uma notificação aparecia para o motorista para que ele/ela respondesse se a carona realmente havia ocorrido ou não. Essa informação era armazenada como booleano (*true* ou *false*) na coluna `done` da tabela **`rides`**.

3.3.3 Questões sobre a Modelagem do Banco Transacional

Com isso em mente, e analisando bem o modelo lógico detalhado da figura 6, foram identificados alguns problemas com essa modelagem.

Primeiramente, observa-se que há atributos em algumas tabelas que referenciam conceitos de outras tabelas, porém não utilizam chave estrangeira, como por exemplo o campo "`myzone`" da tabela `rides`. Este campo contém a zona do bairro de origem/destino da carona, porém o correto seria referenciar a tabela `zones` por meio de uma chave estrangeira. Há outros exemplos deste mesmo problema na modelagem.

Além disso, verifica-se que há colunas com nomes muito pouco descritivos, e alguns até mesmo errados. Por exemplo, a coluna `date` da tabela `rides` contém a data e hora programadas para a ocorrência da carona. O nome da coluna não transmite essa ideia, além de induzir o analista a achar que o campo contém somente uma data, e não um horário. Um exemplo de incorretude é a coluna `id_ufrj` da tabela users. Intuitivamente, o entendimento mais comum e direto seria o de que essa coluna contém o número do DRE (Divisão de Registro de Estudantes) do aluno, ou mesmo a matrícula do servidor no caso de usuários do corpo docente. Porém, analisando essa coluna e a quantidade de dígitos de seus valores numéricos percebe-se que o valor ali contido é o do CPF (Cadastro de Pessoa Física) do usuário. Logo, é evidente que o nome `id_ufrj` não caracteriza bem a coluna.

Outra questão evidente é a falta de adequação para uma potencial escalabilidade do projeto. Se é interessante haver uma tabela com registros das instituições abarcadas pelo

Caronaê, por que há uma coluna chamada `id_ufrj` na tabela `users`? Isso parece algo restritivo, dado que poderia potencialmente haver usuários de outras instituições.

O método empregado no momento da exclusão de caronas também configura um problema. Como descrito mais acima, no momento da exclusão da carona no aplicativo pelo usuário, o registro de criação da carona na tabela `ride_user` é excluído, porém os registros de seus pedidos não. Isso faz com que se perca a informação do motorista daquela carona, e haja pedidos de carona referenciando um id de carona inexistente. Por conta dessas questões referenciais, no ETL detalhado mais a frente, não são levadas em consideração as caronas excluídas.

4 PRIMEIRA SOLUÇÃO ANALÍTICA APOIADA PELO POWER BI

Em 2021, o autor deste trabalho ingressou na equipe do Caronaê como extensionista, reinaugurando a frente de dados do projeto. Sua primeira tarefa foi documentar o banco de dados transacional existente, criando o modelo lógico através de engenharia reversa, dicionário de dados, modelagem de processos BPMN, e mapeando inconsistências no banco.

Após essa etapa, foi conduzido um estudo focado nas análises de uso do aplicativo ao longo de seu funcionamento (entre 2016 e 2020). Para isso, empregou-se o uso de SQL para fazer as consultas no banco no PostgreSQL através do utilitário pgadmin, e da plataforma Microsoft Power BI para gerar visualizações sobre os dados extraídos do banco. O Google Planilhas foi usado para a criação de gráficos auxiliares.

Assim, foi desenvolvido um painel com diversos dados de uso do aplicativo, abrangendo algumas categorias diferentes. A seguir são apresentadas algumas dessas análises e as conclusões alcançadas a partir delas. Posteriormente, são sublinhadas as limitações de uma solução de *Business Intelligence* quando este tem como base um banco de dados transacional e não analítico.

4.1 GLOSSÁRIO

Considerou-se importante, à época, criar um pequeno glossário com alguns termos técnicos que foram empregados ao longo do desenvolvimento do painel e do relatório, para facilitar a compreensão de todos.

O primeiro termo descrito foi o de ***status***, que representa a situação de um usuário relativo a uma carona específica. Ele pode assumir os valores “motorista” **(**usuário que cria e dá a carona), “aceito” (usuário que pediu para entrar na carona e foi aceito pelo motorista´), “recusado” (usuário que pediu para entrar na carona e foi recusado pelo motorista), “pendente” (usuário que pediu para entrar na carona mas o motorista nunca respondeu ao pedido) ou “desistente” (usuário que pediu para entrar na carona mas depois desistiu).

Para a análise, fez-se necessário discernir as caronas que realmente aconteceram das caronas que foram criadas no aplicativo mas não aconteceram. Denominam-se as caronas que teriam realmente acontecido **caronas válidas.**

A primeira ideia para separar essas duas categorias foi utilizar a coluna `done` da tabela `rides`. Como descrito na seção 3, após a presumida conclusão de uma carona, uma notificação chegava ao motorista através do aplicativo, para este responder se a carona ocorreu ou não.

À época deste estudo, membros mais veteranos da equipe Caronaê instruíram o autor a não utilizar o conteúdo desta coluna como métrica para dizer se a carona realmente ocorreu, pois foi dito que os usuários não respondiam a essa notificação, ou seja, a métrica não representava a realidade.

Por fim, acabou se definindo que as caronas válidas seriam as caronas com pelo menos uma pessoa aceita ou pendente. Seriam caronas com probabilidade maior de terem realmente acontecido.

Consideram-se essas as caronas válidas e não apenas as que tiveram alguém aceito, porque, pelo entendimento da equipe Caronaê à época, muitas vezes o motorista não aceitava o pedido do usuário no aplicativo, mas o chamava para a carona através de outros meios. É importante reforçar que essa escolha serve apenas para a estimativa das quantidades apresentadas no relatório.

Além disso, definiu-se o conceito de **usuários ativos**. Foram chamados de usuários ativos aqueles que haviam criado uma carona, sido aceitos em uma, ou ficado com o pedido para entrar em carona pendente, pelo menos uma vez.

4.2 VISUALIZAÇÕES E ANÁLISES

As presentes análises tiveram como objetivo fazer um mapeamento geral do uso do aplicativo Caronaê em seu tempo de funcionamento, bem como tentar compreender os padrões de seus usuários e a distribuição de caronas por zonas e pólos universitários de cobertura. Previamente, não havia dados consolidados que cumprissem esses objetivos de forma compreensiva.

Essas análises seriam usadas posteriormente como insumo para a equipe do projeto, a fim de informar as decisões para o futuro do mesmo.

4.2.1 Números Gerais de Caronas

A figura 7 apresenta alguns números gerais sobre as caronas. A estimativa da época era que algo entre 6.222 e 14.689 caronas haviam ocorrido de fato. Esse número leva em

conta as caronas que tiveram alguém aceito como mínimo, e as que tiveram alguém aceito ou pendente como máximo.

Figura 7: Números gerais do estudo de 2021 sobre dados de uso do aplicativo

Fonte: elaboração própria

4.2.2 Visão Temporal de Caronas Criadas e Caronas Válidas Dadas

A figura 8 mostra as caronas criadas por período, e a figura 9 as caronas válidas ocorridas por período. Esses gráficos evidenciam muito bem alguns pontos sobre a história do uso do Caronaê, mas podem levar a uma interpretação limitada da realidade se avaliada apenas a tendência de queda ao longo dos anos sem uma contextualização mais aprofundada. Para isso, foram incluídos outros dois gráficos gerados no Google Planilhas, pois o Power BI não permite a visualização de colunas tão finas.

Figura 8: Caronas criadas por período - parte do estudo de 2021 sobre dados de uso do aplicativo

Fonte: elaboração própria

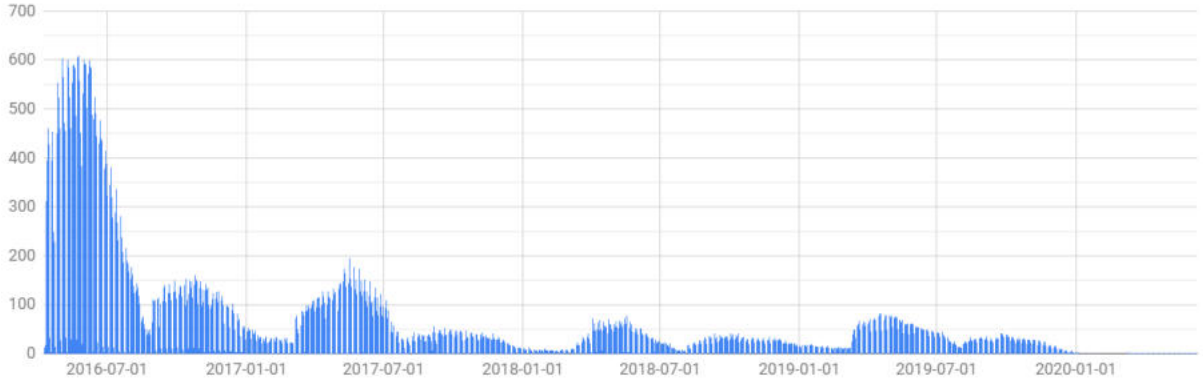
Figura 9: Caronas válidas ocorridas por período - parte do estudo de 2021 sobre dados de uso do aplicativo

Fonte: elaboração própria

A figura 10 representa também as caronas criadas, mas apresentando-as pela data programada para ocorrência. Este gráfico fornece uma visão muito mais dinâmica da criação

de caronas, mostrando como ela aumentava no início de cada período (março e agosto) e diminuía lentamente ao longo do período, para depois diminuir muito durante as férias.

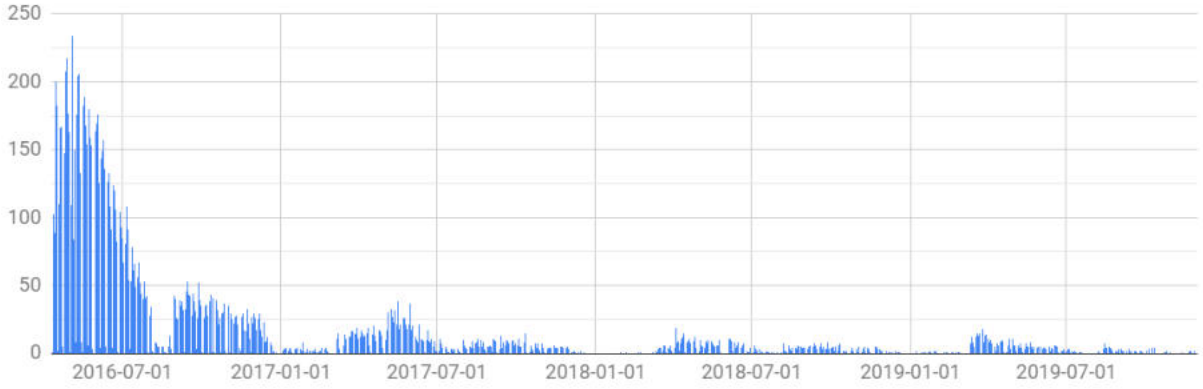
Figura 10: Caronas criadas por data programada para ocorrência - parte do estudo de 2021 sobre dados de uso do aplicativo



Fonte: elaboração própria

Já a figura 11 representa as caronas válidas pela data marcada para ocorrência. É possível observar os mesmos padrões da figura 10 aqui, mantendo essa dinamicidade do dia a dia.

Figura 11: Caronas válidas por data programada para ocorrência - parte do estudo de 2021 sobre dados de uso do aplicativo



Fonte: elaboração própria

As oscilações nos dados são, assim, contextualizadas. O grande número de caronas criadas e ocorridas em 2016.1 pode ser explicado pelo fato do Caronaê ter sido lançado nessa época, tendo grande campanha de divulgação e representando ainda uma novidade na UFRJ. Nos anos seguintes percebe-se uma queda, provavelmente auxiliada pela popularização de

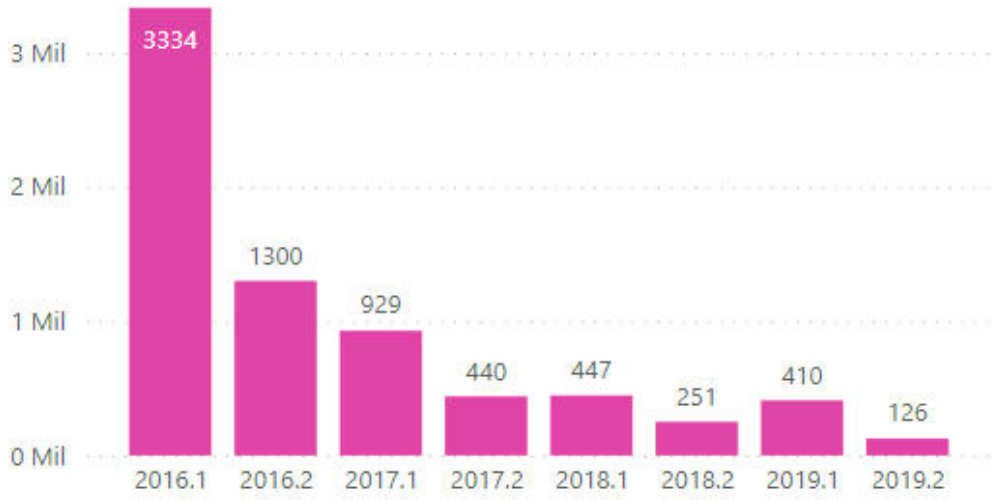
grupos de carona por Whatsapp e Telegram e a falta de campanhas de divulgação do Caronaê. No início de 2019 nota-se um leve aumento proveniente da expansão do aplicativo para o campus Macaé. Já em março de 2020, com a pandemia, o aplicativo saiu do ar, logo se vêem poucas caronas criadas apenas no início de 2020.1.

Levando em conta as figuras 8, 9, 10 e 11 e essa análise, percebe-se que vários fatores impactaram o uso do Caronaê ao longo dos anos. Essas mesmas tendências podem ser vistas nos outros gráficos do relatório que fazem uso dessa visão temporal.

4.2.3 Usuários Ativos

A figura 12 mostra a quantidade de usuários ativos por período no aplicativo. O total encontrado de usuários que estiveram ativos alguma vez ao longo de todo o período de funcionamento do aplicativo foi de 5.127.

Figura 12: Usuários ativos por período - parte do estudo de 2021 sobre dados de uso do aplicativo



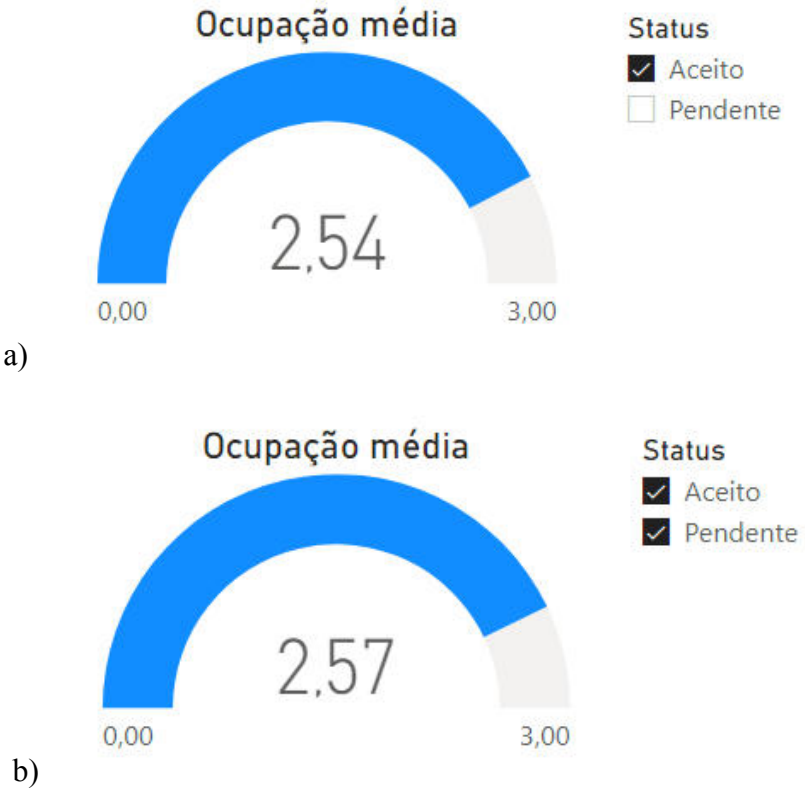
Fonte: elaboração própria

4.2.4 Ocupação Média

Os indicadores nas figuras 13a e 13b mostram a Ocupação Média das caronas, baseados em diferentes métricas. O primeiro leva em conta apenas a média de usuários aceitos como passageiros (+ 1, para que se inclua o motorista), dentro do universo de caronas que tiveram alguém aceito como passageiro. O segundo leva em conta a média de usuários aceitos como passageiros ou que ficaram com o pedido pendente (+ 1, para que se inclua o motorista), dentro do universo de caronas que tiveram alguém aceito como passageiro ou com

o pedido pendente. De qualquer maneira, nota-se que o número encontrado à época não oscila muito: ele varia entre 2,54 e 2,57.

Figura 13: Ocupação média das caronas do Caronaê - parte do estudo de 2021 sobre dados de uso do aplicativo

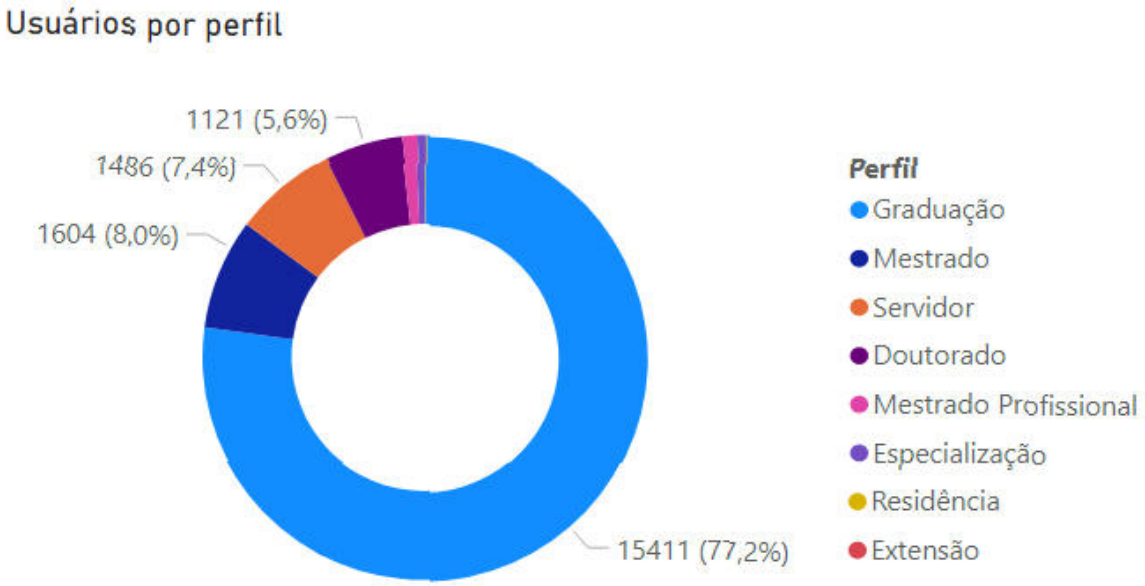


Fonte: elaboração própria

4.2.5 Distribuição de Usuários por Vínculo Acadêmico

Na figura 14, é possível visualizar a divisão dos usuários cadastrados no aplicativo por vínculo acadêmico (ou perfil). A grande maioria dos usuários são da graduação, mas também há uma quantidade expressiva do mestrado e doutorado e de servidores. O número total de usuários cadastrados no aplicativo é de 19.974.

Figura 14: Usuários por vínculo acadêmico - parte do estudo de 2021 sobre dados de uso do aplicativo

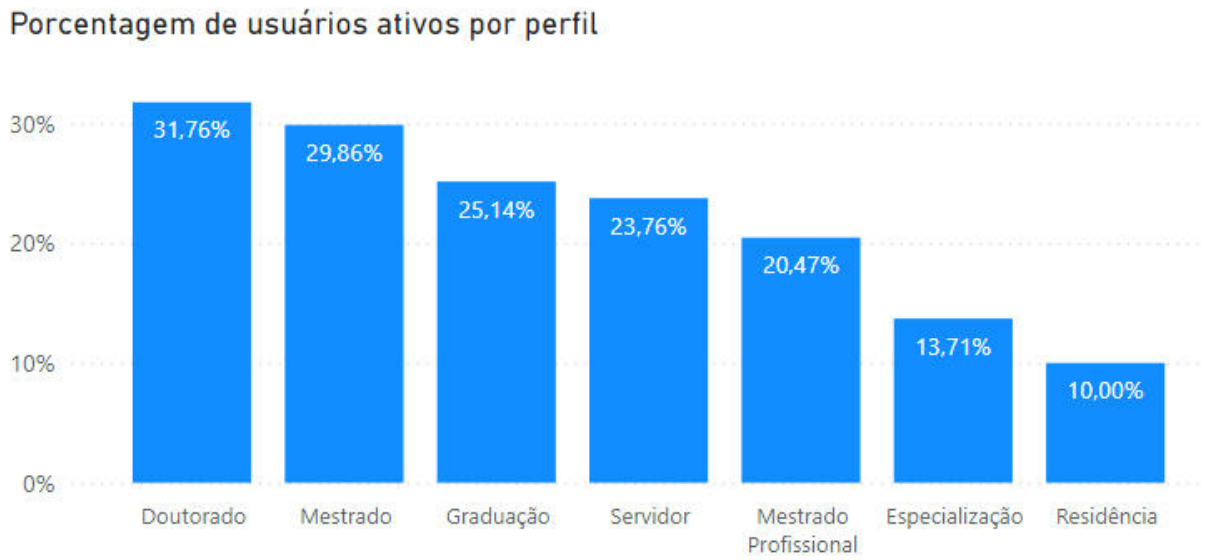


Fonte: elaboração própria

É possível ver na figura 15 a porcentagem de usuários ativos por vínculo acadêmico. A porcentagem geral de usuários ativos foi de 25,67%**.** É notável o engajamento dos usuários de mestrado e doutorado, com porcentagem mais alta que os outros perfis. Porém, ainda assim, observa-se que aproximadamente três a cada quatro usuários cadastrados nunca sequer participaram do sistema de busca e criação de caronas, muito menos participaram de uma carona através do aplicativo.

Obs: A categoria “Extensão” foi excluída deste gráfico pois o seu percentual de ativos estava em 66,67%, não por essa categoria ter muitos usuários ativos e sim porque existem apenas três usuários desta categoria, dos quais dois foram ativos. Logo, mostrá-la com um percentual tão maior que os outros poderia levar a conclusões errôneas.

Figura 15: Porcentagem de usuários ativos por vínculo acadêmico - parte do estudo de 2021 sobre dados de uso do aplicativo

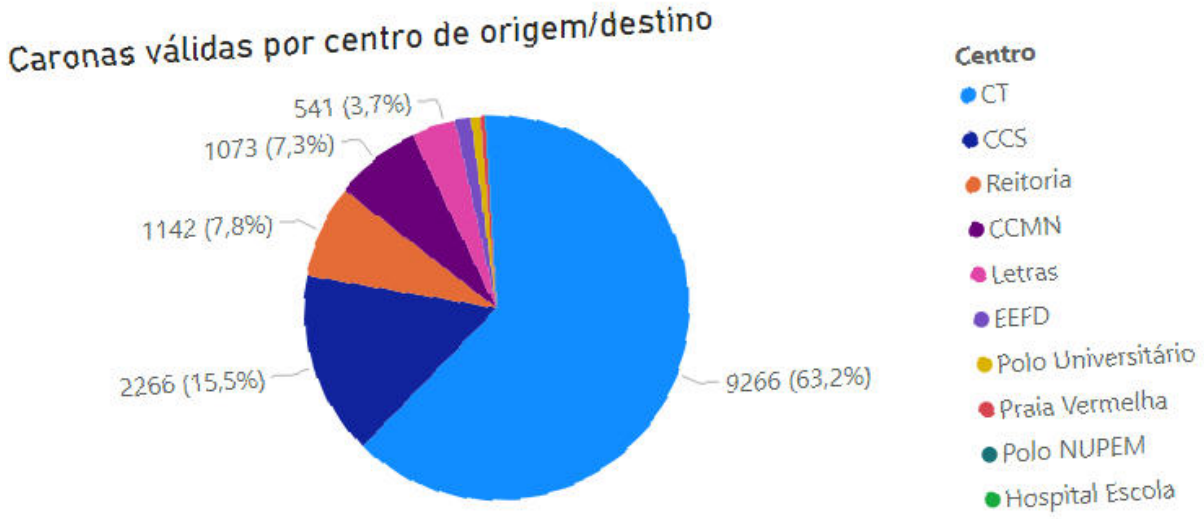


Fonte: elaboração própria

4.2.6 Distribuição de Caronas por Centros

A figura 16 apresenta a quantidade de caronas válidas ocorridas por centro de origem/destino. Observa-se que o CT é o centro de maior demanda. Além deste ser um dos maiores centros, é possível que isso seja decorrente do fato de o projeto ter surgido na Engenharia, que fica neste centro.

Figura 16: Caronas válidas por centro de origem/destino - parte do estudo de 2021 sobre dados de uso do aplicativo

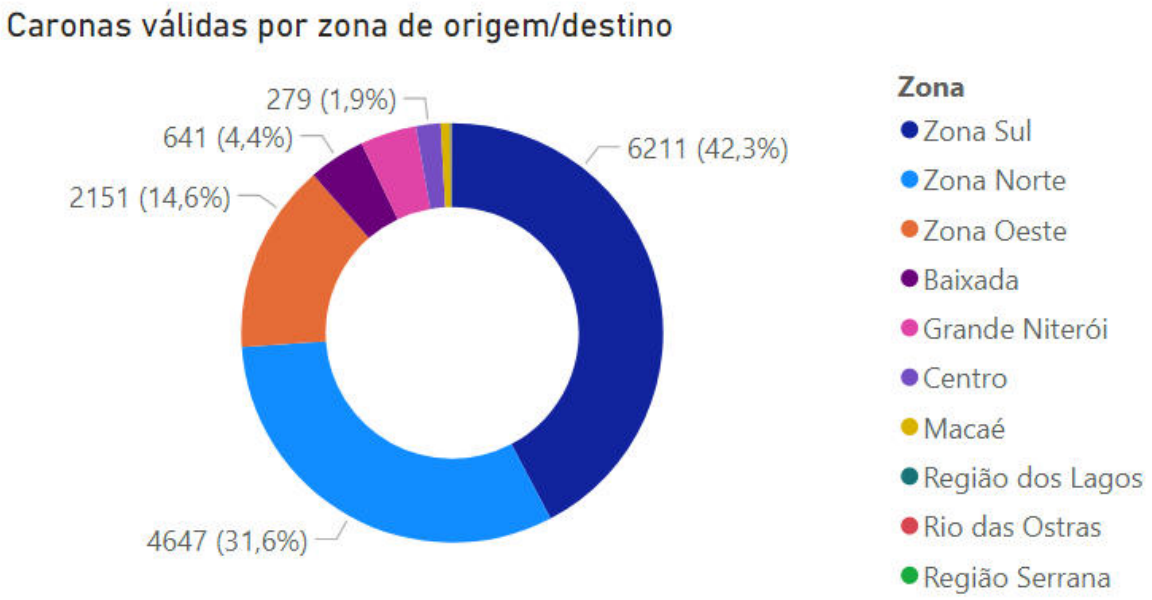


Fonte: elaboração própria

4.2.7 Distribuição de Caronas por Zonas

A figura 17 apresenta a quantidade de caronas válidas ocorridas, divididas por zona de origem/destino (dependendo se estão indo ou voltando da UFRJ). Percebe-se uma predominância da Zona Sul, Zona Norte, e Zona Oeste do Rio.

Figura 17: Caronas válidas por zona de origem/destino - parte do estudo de 2021 sobre dados de uso do aplicativo

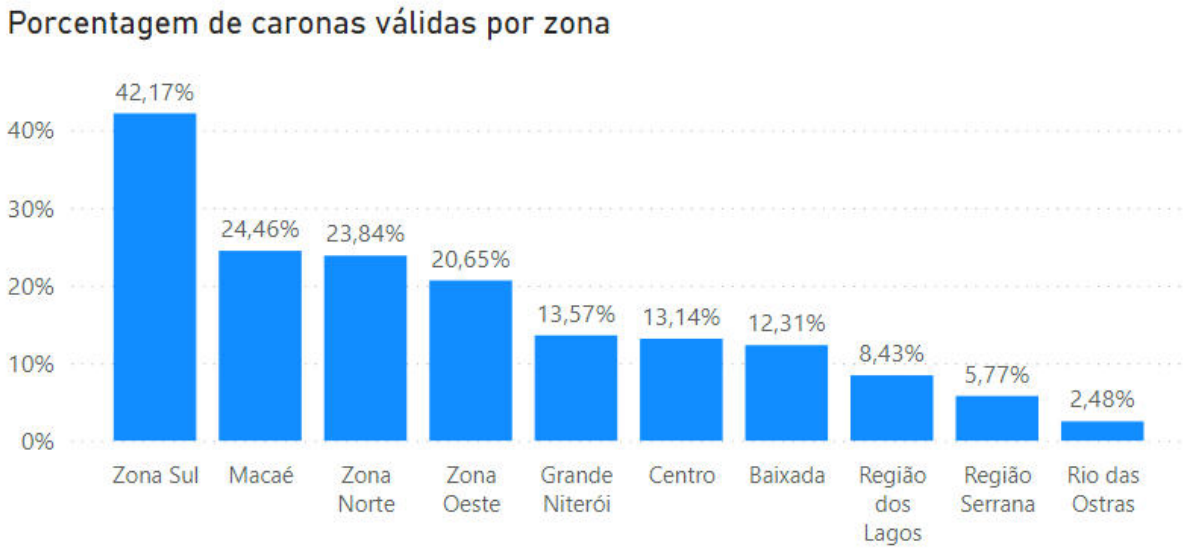


Fonte: elaboração própria

A figura 18 mostra a porcentagem de caronas válidas por zona. Ou seja, divide-se a quantidade de caronas válidas dadas de/para cada zona pela quantidade de caronas oferecidas de/para cada zona para chegar a essa taxa de conversão da zona.

Nota-se que a Zona Sul aparece com uma taxa de conversão muito mais alta que todas as outras zonas.

Figura 18: Porcentagem de caronas válidas por zona - parte do estudo de 2021 sobre dados de uso do aplicativo



Fonte: elaboração própria

Em suma, a primeira fase do projeto foi crucial para validar a possibilidade de extrair valor dos dados do Caronaê. A partir da documentação do banco de dados transacional e do uso de consultas SQL e da plataforma Power BI, o autor pôde realizar uma análise inicial abrangente. Os resultados proporcionaram um mapeamento geral do uso do aplicativo, identificando padrões de engajamento temporal, da distribuição de caronas por centros universitários e zonas do Rio de Janeiro, além dos perfis e comportamentos dos usuários.

Contudo, apesar das conclusões obtidas, o processo revelou as limitações de se fazer análises de BI diretamente de um banco de dados transacional. A alta normalização do banco, ideal para as transações do dia a dia, faz com que as consultas sejam menos eficientes, exigindo múltiplos JOINs para obter informações. Além disso, a falta de uma etapa de limpeza e transformação de dados automatizada faz com que os dados no qual o painel se baseia possam estar incompletos ou ser de baixa qualidade. As transformações necessárias tiveram que ser feitas manualmente, o que é muito mais moroso. Com base nesses desafios, a próxima etapa do projeto visa a construção de uma solução permanente e robusta, que supere esses obstáculos e permita uma análise mais profunda e confiável.

5 SEGUNDA SOLUÇÃO ANALÍTICA USANDO MODELAGEM DIMENSIONAL E TABLEAU

A seguir, a nova solução analítica produzida para este trabalho é abordada, utilizando-se modelagem dimensional para criar um ambiente analítico dedicado com base em um data warehouse, tirando proveito das vantagens que essa metodologia traz. É discutida também a implementação do painel analítico produzido no Tableau.

5.1 OBJETIVOS DA SEGUNDA SOLUÇÃO ANALÍTICA

Como discutido na seção 4, a primeira solução analítica desenvolvida para o Caronaê atendeu a algumas necessidades pontuais do projeto, como a de um mapeamento geral de seus grandes números históricos. Percebeu-se, porém, a necessidade de uma solução mais robusta de análise, que permitisse análises mais rápidas, eficientes e interseccionais por meio de um painel gerencial com alguma frequência de atualização. Dessa forma, torna-se possível para a equipe do projeto acompanhar as estatísticas de uso do aplicativo, e tomar suas decisões estratégicas informados pelos dados mais recentes.

A frequência de atualização proposta é a diária, garantindo que os relatórios sejam atualizados com os dados fechados do dia anterior, com o processo de ETL sendo executado idealmente de madrugada, para seguir as melhores práticas de engenharia de dados. Por mais que as transformações mais complexas não sejam executadas no servidor do banco operacional, o que minimiza a possibilidade de uma sobrecarga do mesmo, agendar o ETL para horários de menor volume transacional (ex: 1h às 5h da manhã) garante que a operação de SELECT no banco transacional ocorra quando houver a menor concorrência por recursos. Isso assegura que a experiência do usuário, mesmo durante o pico mais suave, não seja afetada.

Além disso, é interessante do ponto de vista da lógica de negócio que o relatório diário tenha os dados completos de um dia fechado. O agendamento noturno garante que todos os dados do dia anterior (até 23:59:59) sejam processados no ciclo noturno.

Assim, com um data warehouse hospedado em nuvem, bebendo da fonte do banco operacional também hospedado em nuvem através de um ETL diário, e com um painel analítico ligado diretamente ao data warehouse, mostrando seus dados de forma concisa e intuitiva, tem-se um ambiente analítico robusto e dinâmico.

5.2 TRATAMENTO E ALTERAÇÕES NO BANCO

Algumas mudanças tiveram que ser feitas no banco transacional, para que o processamento analítico funcionasse da melhor forma possível.

A primeira foi a exclusão de 33 pedidos na tabela `ride_user` referentes a caronas deletadas, todos com *status* pendente. Esses pedidos foram removidos pois foram criados após a exclusão de suas respectivas caronas.

A segunda foi atribuir a zona correta às caronas que tinham a zona "Outros" como sua zona de origem/destino. Essa zona foi criada para alocar bairros em zonas não catalogadas no banco de dados. Dessas caronas, porém, poucas de fato eram de/para bairros de uma zona não catalogada. A maioria delas tinham como bairro de origem/destino um bairro que fazia parte de uma zona catalogada no banco, e foram, então, associadas às suas zonas corretas. Além disso, foram criadas novas zonas para abranger os bairros que anteriormente estavam associados à zona "Outros".

Em seguida, verificou-se que havia usuários com o valor da coluna `car_owner` na tabela `rides` igual a *false* (ou seja, listados como não donos de carros), mas que tinham modelo de carro listado. Provavelmente esses usuários registraram o carro depois do momento de cadastro por, possivelmente, não terem ainda carro naquele momento e o terem adquirido posteriormente. O valor de `car_owner` foi mudado para *true* para todos os usuários deste grupo que tinham modelo e placa de carro válidos.

5.3 NOVA MODELAGEM DIMENSIONAL

Com base na metodologia proposta por Barbieri (2011), foi construído um modelo dimensional de data warehouse para o ambiente analítico.

5.3.1 Processo de Modelagem do Data Warehouse

Há uma série de passos a serem seguidos em um processo de modelagem dimensional. O primeiro deles é estabelecer o fato, o que se quer medir. No caso desta modelagem, há dois fatos, dois eventos centrais: a carona e o pedido.

Denomina-se carona qualquer corrida criada no aplicativo, seja ela uma futura corrida ou uma já ocorrida, ou seja, uma carona oferecida. Independente da corrida ter acontecido com ou sem passageiros, continua denominando-se carona. Independente da quantidade de

passageiros, uma corrida equivale a uma só carona. Toda carona tem necessariamente um usuário motorista atrelado a ela, que foi quem a criou e é quem a oferece.

Um pedido é um pedido de entrada em uma carona. Cada pedido está atrelado a apenas um usuário requerente e a apenas uma carona (na qual o usuário deseja ingressar). Todo pedido inicia-se como um pedido pendente e posteriormente pode ter seu *status* alterado para aceito, recusado, ou desistência (quando o próprio usuário requerente o retira), porém independente do *status*, continua sendo um pedido.

Assim sendo, uma carona pode ter diversos pedidos atrelados a ela. É importante, porém, notar que o número de pedidos aceitos não pode ultrapassar o total de vagas ofertadas na carona, algo que é definido no momento de sua criação pelo motorista.

O segundo passo importante é definir a granularidade dos fatos, o que um registro único na tabela fato representa. Para o fato carona, o nível de granularidade escolhido é o da própria carona, ou seja, uma corrida, uma viagem de carro. Uma parte dos dados referentes a ela são definidos no momento de sua criação, como quantidade de vagas ofertadas, locais de origem e destino, e outra parte é definida posteriormente, como se a carona foi marcada como ocorrida pelo motorista.

Para o fato pedido, o nível de granularidade é o de um pedido de um usuário para ingressar em uma carona. Seu registro inclui o dado de quem é o usuário requerente, a qual carona o pedido se refere, o *status* atual do pedido, além do momento de criação e de última atualização do mesmo.

O terceiro passo é a definição das dimensões que contextualizam o(s) fato(s), ou seja, que informações devem descrevê-lo(s). No caso do fato carona, é essencial estabelecer quem é o usuário motorista e criador da carona, a data e hora da criação da carona no aplicativo, a data e hora programadas para sua ocorrência, quais são seus locais de origem e destino e as informações da rotina da qual ela faz parte, se fizer parte de alguma. Uma rotina é um grupo de caronas recorrentes criado por um usuário, constituído por caronas que ocorrem toda semana nos mesmos dias da semana, sempre no mesmo horário, até uma data limite. Os dias de repetição da rotina, o horário das caronas e a data limite são definidas pelo usuário criador da rotina (consequentemente o motorista em todas as caronas da rotina) no momento de sua criação.

No caso do fato pedido, é necessário entender quem é o usuário requerente, a que carona o pedido se refere, data e hora da criação do pedido no aplicativo, a data e hora da sua última atualização e o seu *status* atual.

5.3.2 Seleção de Dados

Nem todos os dados de um ambiente transacional são úteis para um ambiente analítico. No presente trabalho, foi decidido incluir no data warehouse apenas os dados relativos às caronas e suas rotinas, aos pedidos e seus *status*, e às dimensões que os caracterizam: usuários, centros universitários e zonas. Dados relativos ao funcionamento interno do sistema ou do banco de dados não foram trazidos ao DW, pois são excessivamente técnicos e operacionais, e não estão relacionados à matéria-prima do projeto: as caronas. Assim, esses dados não têm valor para membros da equipe Caronaê que não sejam desenvolvedores.

Com isso, foi decidido não trazer para o data warehouse os dados das seguintes tabelas do banco transacional:

- **`admins`:** essa tabela contém dados sobre os usuários administradores do sistema, os membros da equipe Caronaê. Não são dados relevantes para as análises pretendidas para o data warehouse.
- **`failed_jobs`:** essa tabela contém dados sobre agendamentos falhos de caronas e tentativas de login falhas. Esses dados até poderiam ser interessantes do ponto de vista de melhorias técnicas do aplicativo, porém os dados aqui estão muito pouco estruturados e requereriam um estudo mais aprofundado dos detalhes técnicos das conexões entre banco e servidor. Além disso, há apenas 16 registros nessa tabela, o que é pouco para criar qualquer análise relevante.
- **`migrations`:** os dados nessa tabela são sobre migrações feitas para a criação e atualização do banco de dados operacional. É mais um registro da história das mudanças no banco, logo tem um caráter mais de metadado, e não é interessante para a análise no escopo deste trabalho.
- **`notifications`:** aqui há dados sobre as notificações enviadas para os usuários através do aplicativo. Todos os dados relevantes dessas notificações estão em outras tabelas, como a criação de um pedido, a resposta de um motorista a um pedido, etc.
- **`password_resets`:** as solicitações de reset de senha feitas pelos usuários são dados extremamente operacionais, não são interessantes em um nível analítico.

Todas as outras tabelas do banco transacional foram aproveitadas de alguma forma, como descrito na seção seguinte. É importante pontuar, porém, que nem todos os atributos do DW são utilizados nas análises do painel gerencial, pois o presente trabalho é uma primeira

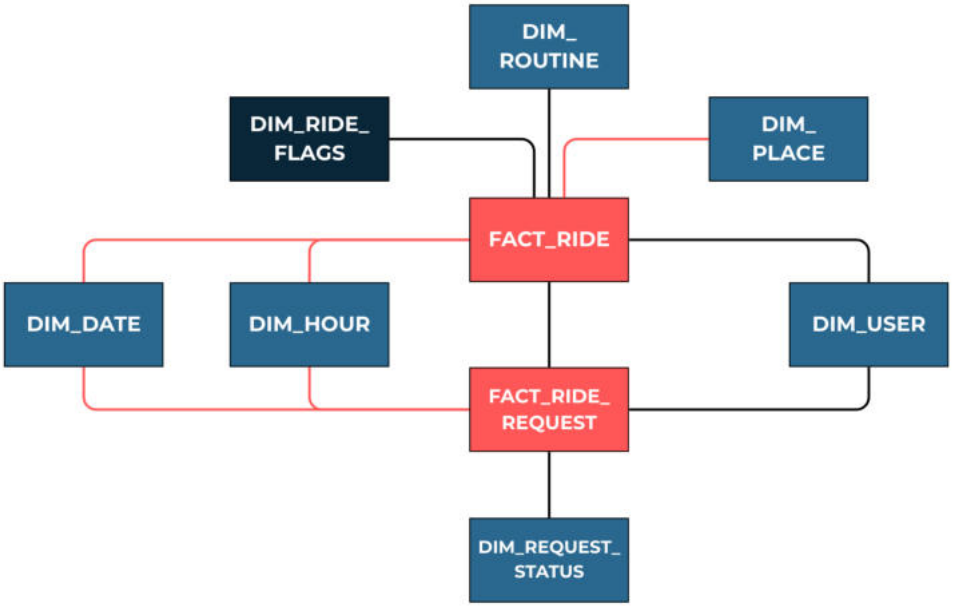
versão, e entende-se que com a evolução da ideia mais análises poderão ser criadas em cima do modelo aqui proposto.

Para estar em conformidade com a LGPD (Lei Geral de Proteção de Dados), dados sensíveis, como nome ou CPF dos usuários, não foram trazidos para o data warehouse. A supressão destas informações não tem impacto no resultado final, já que, para os objetivos pretendidos, tais dados não têm grande valor analítico, por não serem dados de natureza agregatória. Além disso, é ideal evitar tê-los no data warehouse pois isso anula a possibilidade de alguma análise do painel gerencial que venha a ser mostrada ao público geral contenha algum dado sensível.

5.3.3 Definição da Modelagem para o Data Warehouse

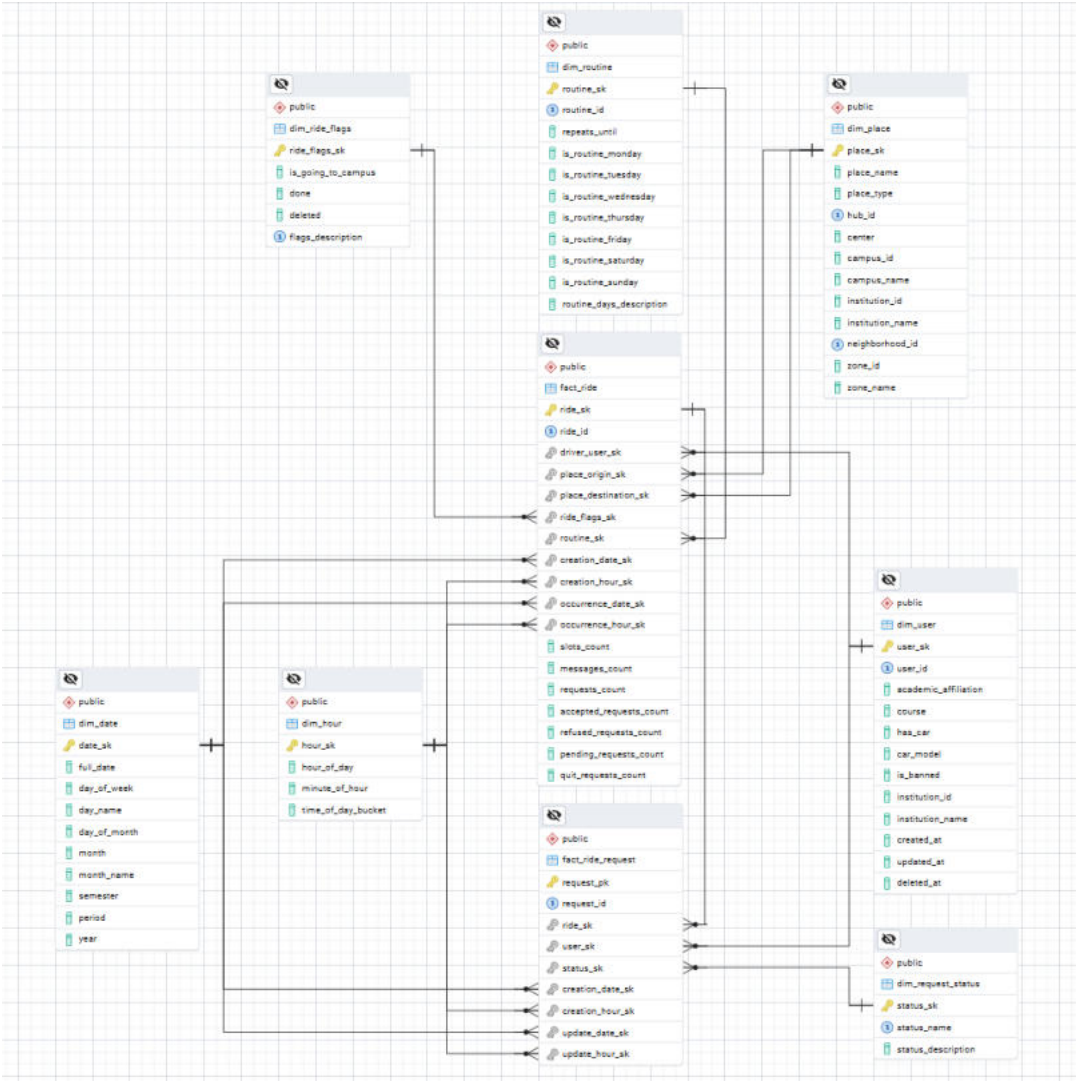
Novamente foram gerados dois modelos lógicos. Na figura 19 há um modelo lógico simplificado, apenas com as tabelas do data warehouse, e relações entre elas representadas por linhas. Os retângulos vermelhos representam tabelas fato e os azuis representam dimensões, sendo que o azul mais escuro representa dimensões sucata. As linhas vermelhas representam relações duplas (dois pares de chaves conectadas entre as tabelas). Já na figura 20, há um modelo lógico mais detalhado, com os atributos de cada relação, identificação das chaves e setas apontando de um atributo chave para outro, ilustrando as conexões entre as tabelas de forma mais clara.

Figura 19: Modelo lógico simplificado do novo ambiente analítico do Caronaê



Fonte: elaboração própria

Figura 20: Modelo lógico detalhado do novo ambiente analítico do Caronaê



Fonte: elaboração própria

Há, nessa modelagem, duas tabelas fato: `Fact_Ride` e `Fact_Ride_Request`. Elas se relacionam com as sete tabelas de dimensão para formar o modelo. Segue a definição das tabelas, seus atributos e suas relações:

`Dim_Date`: dimensão pré-armazenada com datas e atributos dessas datas. As datas pré-carregadas aqui vão de 01/04/2016 (alguns dias antes do início do funcionamento do aplicativo) até 31/12/2035 (10 anos no futuro). Seus atributos encontram-se descritos nos quadro 1.

Quadro 1: Atributos de `Dim_Date` e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`date_sk`	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
`full_date`	Data inteira	date	-	Não Nula
`day_of_week`	Dia da semana (número de 1 a 7)	integer	-	Não Nula
`day_name`	Dia da semana (nome)	character varying	-	Não Nula
`day_of_ month`	Dia do mês	integer	-	Não Nula
`month`	Mês	integer	-	Não Nula
`month_name`	Nome do mês	character varying	-	Não Nula
`semester`	Semestre do ano (1 ou 2)	integer	-	Não Nula
`period`	Período universitário (ANO.1 de janeiro a julho, ANO.2 de agosto a dezembro)			
`year`	Ano	integer	-	Não Nula

Fonte: elaboração própria

Essa tabela permite que sejam feitas análises por recortes temporais mais específicos que apenas ano, mês e dia. Foi decidido dividir a dimensão temporal em duas: `Dim_Date` e `Dim_Hour`, para que não haja uma tabela com milhões de linhas, e que para que seja possível fazer análises apenas no nível da data ou hora de forma mais flexível.

`Dim_Hour`: dimensão pré-armazenada com horas e atributos dessas horas. Seus atributos encontram-se descritos no quadro 2.:

Quadro 2: Atributos de `Dim_Hour` e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`hour_sk`	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
`hour_of_day`	A hora do dia	integer	-	Não Nula

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
minute_of_hour	O minuto da hora	integer	-	Não Nula
time_of_day_bucket	O momento do dia (manhã, tarde, noite ou madrugada)	character varying	-	Não Nula

Fonte: elaboração própria

Essa tabela permite que sejam feitos recortes temporais pelo momento do dia.

Dim_User: dimensão com dados sobre os usuários do aplicativo Caronaê. Seus atributos encontram-se descritos no quadro 3.

Quadro 3: Atributos de Dim_User e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
user_sk	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
user_id	Chave de negócio original da tabela users do banco transacional, para servir de referência	integer	-	Não Nula e Única
academic_affiliation	Vínculo acadêmico do usuário (Graduação, Mestrado, Doutorado, Servidor, etc.)	character varying	-	-
course	Curso ao qual o usuário pertence	character varying	-	-
has_car	Booleano que indica se o usuário tem ou não um carro cadastrado na plataforma	boolean	-	Não Nula
car_model	Modelo do carro do usuário (se tiver carro cadastrado)	character varying	-	-
is_banned	Booleano indicando se o usuário foi banido da plataforma	boolean	-	Não Nula
institution_id	Chave de negócio original da instituição a que o usuário pertence, da tabela institutions do banco transacional	integer	-	Não Nula

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`institution_name`	Nome da instituição a que o usuário pertence	character varying	-	-
`created_at`	Data e hora de cadastro do usuário no aplicativo	timestamp	-	-
`updated_at`	Data e hora em que o usuário usou o aplicativo pela última vez	timestamp	-	-
`deleted_at`	Data e hora da exclusão da conta do usuário no aplicativo (se o usuário tiver excluído sua conta)	timestamp	-	-

Fonte: elaboração própria

`Dim_Place`: dimensão com dados sobre locais, sejam eles bairros da cidade, com suas respectivas zonas, ou pólos da universidade, com seus respectivos centros, *campi* e instituições. Seus atributos encontram-se descritos no quadro 4.

Quadro 4: Atributos de `Dim_Place` e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`place_sk`	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
`place_name`	Nome do lugar, seja ele bairro ou pólo universitário	character varying	-	-
`place_type`	Tipo do lugar, pode ser “hub” (pólo universitário) ou “neighborhood” (bairro)	character varying	-	-
`hub_id`	Chave de negócio original da tabela `hubs` do banco transacional, para servir de referência	integer	-	Única
`center`	Centro em que o pólo fica	character varying	-	-
`campus_id`	Chave de negócio original do campus em que o pólo fica, da tabela `campi` do banco transacional	integer	-	-

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
campus_name	Nome do campus onde o pólo está localizado	character varying	-	-
institution_id	Chave de negócio original da instituição a que o pólo pertence, da tabela institutions do banco transacional	integer	-	-
institution_name	Nome da instituição a que o pólo pertence	character varying	-	-
neighborhood_id	Chave de negócio original da tabela neighborhoods do banco transacional, para servir de referência	integer	-	Única
zone_id	Chave de negócio original da tabela zones do banco transacional, para servir de referência	integer	-	-
zone_name	O nome da zona onde o bairro está localizado	character varying	-	-

Fonte: elaboração própria

Para os registros de pólo universitário (*hub*), as colunas relativas a bairro e zona estarão nulas. Já para os registros de bairro (*neighborhood*), as colunas relativas ao pólo, centro, campus e instituição estarão nulas.

Foi resolvido não criar dimensões como "dim_zone", "dim_campus" ou "dim_institution" e, ao invés disso, colocar esses dados nesta tabela para que fossem seguidas as regras de desnormalização do esquema estrela.

Dim_Request_Status: dimensão com os possíveis *status* de um pedido para entrar em uma carona. Seus atributos encontram-se descritos no quadro 5.

Quadro 5: Atributos de `Dim_Request_Status` e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`status_sk`	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
`status_name`	Nome do *status* (“*accepted”*, “*refused”*, “*pending”* ou “*quit”*)	character varying	-	Não Nula e Única
`status_description`	Descrição do *status*	character varying	-	-

Fonte: elaboração própria

`Dim_Routine`: dimensão com as rotinas (grupos de caronas recorrentes) e suas informações. Seus atributos encontram-se descritos no quadro 6.

Quadro 6: Atributos de `Dim_Routine` e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`routine_sk`	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
`routine_id`	Atributo identificador original da rotina na tabela `rides` do banco transacional, para servir de referência	boolean	-	Única
`repeats_until`	Data final da repetição da rotina	timestamp	-	-
`is_routine_monday`	Indica se a rotina se repete às segundas-feiras	boolean	-	Não Nula
`is_routine_tuesday`	Indica se a rotina se repete às terças-feiras	boolean	-	Não Nula
`is_routine_wednesday`	Indica se a rotina se repete às quartas-feiras	boolean	-	Não Nula
`is_routine_thursday`	Indica se a rotina se repete às quintas-feiras	boolean	-	Não Nula

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
is_routine_friday	Indica se a rotina se repete às sextas-feiras	boolean	-	Não Nula
is_routine_saturday	Indica se a rotina se repete aos sábados	boolean	-	Não Nula
is_routine_sunday	Indica se a rotina se repete aos domingos	boolean	-	Não Nula
routine_days_description	Descrição textual da combinação das informações das flags de repetição por dia de semana	character varying	-	-

Fonte: elaboração própria

Dim_Ride_Flags: dimensão sucata com informações booleanas sobre as caronas. Seus atributos encontram-se descritos no quadro 7.

Quadro 7: Atributos de Dim_Ride_Flags e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
ride_flags_sk	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
is_going_to_campus	Indica se a carona estava indo para o campus ou voltando dele	boolean	-	Não Nula
done	Indica se a carona foi finalizada	boolean	-	Não Nula
deleted	Indica se a carona foi deletada	boolean	-	Não Nula
flags_description	Descrição textual da combinação das informações das flags	character varying	-	Única

Fonte: elaboração própria

Esta é uma dimensão sucata referenciada pela tabela fato Fact_Ride.

Fact_Ride: tabela fato modelando o evento “carona”, com métricas sobre os pedidos de cada carona. Como estabelecido na seção 5.3.1, o evento carona consiste em uma única corrida, uma carona oferecida por um usuário motorista, independente da mesma ter

passageiros ou não e da quantidade de pedidos relacionados a ela. Seus atributos encontram-se descritos no quadro 8.

Quadro 8: Atributos de `Fact_Ride` e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`ride_sk`	Chave substituta que serve como chave primária e é o ponto de entrada para conexões com outras tabelas	integer	PK	Não Nula e Autoincremental
`driver_user_sk`	Chave estrangeira conectada à chave substituta de `Dim_User`, representando o usuário motorista da carona	integer	FK	Não Nula
`place_origin_sk`	Chave estrangeira conectada à chave substituta de `Dim_Place`, representando o lugar de origem da carona (que pode ser tanto um bairro quanto um pólo universitário)	integer	FK	Não Nula
`place_destination_sk`	Chave estrangeira conectada à chave substituta de `Dim_Place`, representando o lugar de destino da carona (que pode ser tanto um bairro quanto um pólo universitário)	integer	FK	Não Nula
`ride_flags_sk`	Chave estrangeira conectada à chave substituta de `Dim_Ride_Flags`, representando informações adicionais sobre a carona, como se a carona foi finalizada, se ela foi deletada e se ela estava indo ou voltando do campus	integer	FK	Não Nula
`routine_sk`	Chave estrangeira conectada à chave substituta de `Dim_Routine`, representando a rotina da qual a carona faz parte (quando a carona não faz parte de rotina este valor é 0, apontando para a linha de `Dim_Routine` que representa caronas sem rotina)	integer	FK	Não Nula

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`creation_date_sk`	Chave estrangeira conectada à chave substituta de `Dim_Date`, representando a data em que a carona foi criada no aplicativo	integer	FK	Não Nula
`creation_hour_sk`	Chave estrangeira conectada à chave substituta de `Dim_Hour`, representando a hora em que a carona foi criada no aplicativo	integer	FK	Não Nula
`occurrence_date_sk`	Chave estrangeira conectada à chave substituta de `Dim_Date`, representando a data programada para a ocorrência da carona	integer	FK	Não Nula
`occurrence_hour_sk`	Chave estrangeira conectada à chave substituta de `Dim_Hour`, representando a hora programada para a ocorrência da carona	integer	FK	Não Nula
`ride_id`	Dimensão degenerada contendo a chave de negócio original da tabela `rides` do banco transacional, para fins de rastreabilidade com o ambiente transacional	integer	-	Não Nula e Única
`slots_count`	Quantidade de vagas oferecidas na carona	integer	-	-
`messages_count`	Quantidade de mensagens relativas à carona	integer	-	DEFAULT 0
`requests_count`	Quantidade total de pedidos relativos à carona (já excluindo o pedido de criação da carona, de *status* “*driver”*)	integer	-	DEFAULT 0
`accepted_requests_count`	Quantidade de pedidos relativos à carona com *status* “*accepted*”	integer	-	DEFAULT 0
`refused_requests_count`	Quantidade de pedidos relativos à carona com *status* “*refused*”	integer	-	DEFAULT 0

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`pending_requests_count`	Quantidade de pedidos relativos à carona com *status* "*pending*"	integer	-	DEFAULT 0
`quit_requests_count`	Quantidade de pedidos relativos à carona com *status* "*quit*"	integer	-	DEFAULT 0

Fonte: elaboração própria

Essa é uma das duas tabelas fato, juntando as informações da carona por meio de chaves estrangeiras e agregando dados de pedidos relativos à carona.

Por mais que a tabela `Fact_Ride_Request` tenha dados sobre cada pedido, é interessante ter os dados agregados dos pedidos relativos a cada carona aqui para que se possa analisar a quantidade de pedidos no nível da carona de forma mais eficiente.

As dimensões `Dim_Date` e `Dim_Hour` desempenham cada uma dois papéis diferentes nessa tabela fato: o de indicar a data/hora de criação da carona, e o de indicar a data/hora programada para a ocorrência da carona.

A dimensão `Dim_Place` também desempenha dois papéis diferentes nessa tabela fato: o de indicar o local de origem da carona e o de indicar o local de destino da carona.

`Fact_Ride_Request`: tabela fato modelando o evento "pedido", com dados sobre os pedidos relativos a cada carona. Seus atributos encontram-se descritos no quadro 9.

Quadro 9: Atributos de `Fact_Ride_Request` e suas características

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`request_pk`	Chave primária para a tupla da tabela	integer	PK	Não Nula e Autoincremental
`ride_sk`	Chave estrangeira conectada à chave substituta de `Fact_Ride`, representando a carona à qual o pedido se refere	integer	FK	Não Nula
`user_sk`	Chave estrangeira conectada à chave substituta de `Dim_User`, representando o usuário que fez o pedido	integer	FK	Não Nula

ATRIBUTO	DESCRIÇÃO	TIPO	CHAVE	CONSTRAINTS
`status_sk`	Chave estrangeira conectada à chave substituta de `Dim_Request_Status`, representando o *status* do pedido	integer	FK	Não Nula
`creation_ date_sk`	Chave estrangeira conectada à chave substituta de `Dim_Date`, representando a data de criação do pedido	integer	FK	Não Nula
`creation_ hour_sk`	Chave estrangeira conectada à chave substituta de `Dim_Hour`, representando a hora de criação do pedido	integer	FK	Não Nula
`update_ date_sk`	Chave estrangeira conectada à chave substituta de `Dim_Date`, representando a data da última atualização do pedido	integer	FK	Não Nula
`update_ hour_sk`	Chave estrangeira conectada à chave substituta de `Dim_Hour`, representando a hora da última atualização do pedido	integer	FK	Não Nula
`request_id`	Dimensão degenerada contendo a chave de negócio original da tabela `ride_user` do banco transacional, para fins de rastreabilidade com o ambiente transacional	integer	-	Não Nula e Única

Fonte: elaboração própria

Essa é a segunda das duas tabelas fato, com informações no nível do pedido. Por meio de chaves estrangeiras, são trazidas informações de outras dimensões sobre o pedido.

As dimensões `Dim_Date` e `Dim_Hour` desempenham cada uma dois papéis diferentes nessa tabela fato: o de indicar a data/hora de criação do pedido, e o de indicar a data/hora de sua última atualização.

5.4 DESCRIÇÃO DO PROCESSO DE ETL

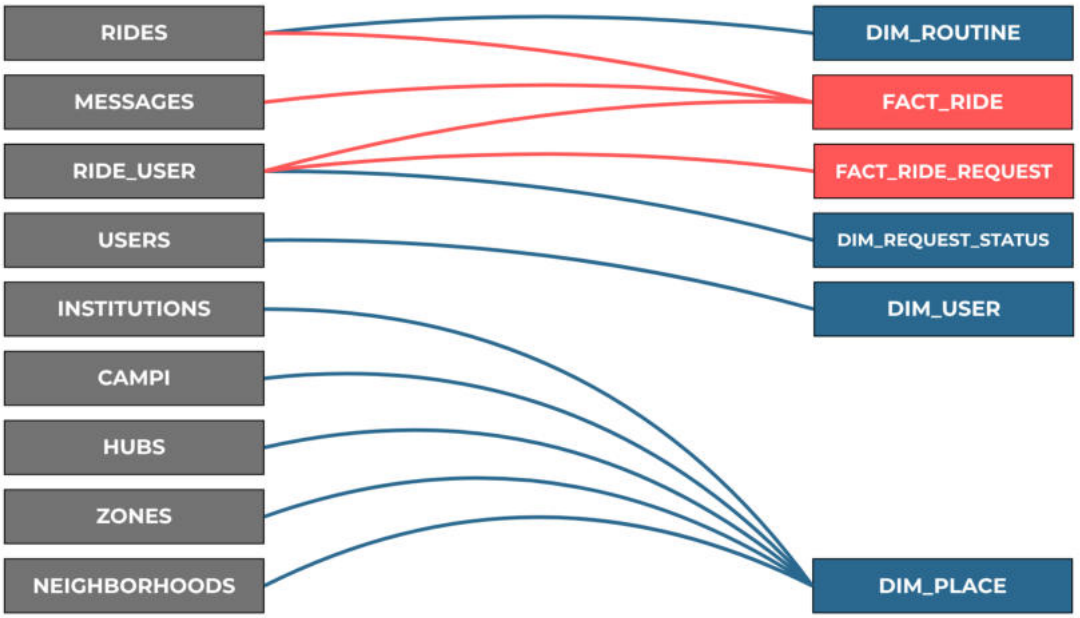
Nesta seção, o processo de ETL é detalhado, como definido na seção 2.3.2 e descrito por Barbieri (2011). Aqui define-se quais dados serão extraídos do banco transacional, quais transformações eles sofrerão e como serão carregados no banco dimensional. Este é um processo central para o ambiente analítico, e será executado diariamente, como descrito na

seção 5.1, garantindo que os dados do painel gerencial estejam sempre atualizados, limpos, padronizados e sem falhas referenciais. A carga será sempre incremental, para economizar tempo e recursos.

5.4.1 Planejamento do Processo de ETL

Na figura 21, é possível ver, em mais alto nível, de quais tabelas do banco transacional cada tabela do data warehouse recebe seus dados. À esquerda estão as tabelas do banco transacional, em cinza, e à direita estão as tabelas do DW, com as tabelas fato em vermelho e as dimensões em azul. As linhas vermelhas representam os dados que fluem para as tabelas fato, e as linhas azuis os dados que fluem para as dimensões.

Figura 21: Descrição da origem dos dados no banco transacional e seu destino no banco dimensional



Fonte: elaboração própria

No quadro 10, encontra-se uma explicação mais detalhada da tabela de origem, tabela de destino, e das transformações que cada dado sofreu. Não foram incluídos os dados que não foram utilizados no data warehouse, nem os dados das tabelas pré-populadas (`Dim_Date`, `Dim_Hour`, `Dim_Ride_Flags`) ou as chaves substitutas.

Quadro 10: Mapeamento entre dados de origem e destino

TABELA TRANSACIONAL	NOME DA COLUNA TRANSACIONAL	TRANSFORMAÇÃO SOFRIDA	TABELA DIMENSIONAL	NOME DA COLUNA DIMENSIONAL
users	id	Coluna renomeada	Dim_User	user_id
users	profile	Coluna renomeada	Dim_User	academic_affiliation
users	course	-	Dim_User	course
users	car_owner	Coluna renomeada	Dim_User	has_car
users	car_model	-	Dim_User	car_model
users	banned	Coluna renomeada	Dim_User	is_banned
users	institution_id	-	Dim_User	institution_id
institutions	name	A coluna no DW corresponde ao nome da instituição à qual o usuário está ligado, obtido através de uma junção das tabelas users e institutions	Dim_User	institution_name
users	created_at	-	Dim_User	created_at
users	updated_at	-	Dim_User	updated_at
users	deleted_at	-	Dim_User	deleted_at
hubs/ neighborhoods	name	A coluna no DW corresponde ao nome do lugar. Pode vir tanto da coluna name da tabela hubs quanto da coluna name da tabela neighborhoods, e é obtido através de uma concatenação dos dados dessas duas tabelas.	Dim_Place	place_name

TABELA TRANSACIONAL	NOME DA COLUNA TRANSACIONAL	TRANSFORMAÇÃO SOFRIDA	TABELA DIMENSIONAL	NOME DA COLUNA DIMENSIONAL
hubs/ neighborhoods	–	A coluna no DW corresponde ao tipo do lugar. Este valor é "*hub*" quando o lugar é um pólo universitário com dados oriundos da tabela hubs, e "*neighborhood*" quando é um bairro com dados oriundos da tabela neighborhoods. Antes da concatenação das duas tabelas, a coluna é criada com os valores como descrito acima.	Dim_Place	place_type
hubs	id	Coluna renomeada	Dim_Place	hub_id
hubs	center	-	Dim_Place	center
hubs	campus_id	-	Dim_Place	campus_id
campi	name	A coluna no DW corresponde ao nome do campus do qual o pólo (*hub*) faz parte, obtido através de uma junção das tabelas hubs e campi	Dim_Place	campus_name
campi	institution_ id	A coluna no DW corresponde ao id da instituição da qual o pólo (*hub*) faz parte, obtido através de uma junção das tabelas hubs e campi	Dim_Place	institution_id
institutions	name	A coluna no DW corresponde ao nome da instituição da qual o pólo (*hub*) faz parte, obtido através de uma junção das tabelas hubs, campi e institutions	Dim_Place	institution_ name
neighborhoods	id	Coluna renomeada	Dim_Place	neighborhood_ id

TABELA TRANSACIO-NAL	NOME DA COLUNA TRANSACIO-NAL	TRANSFORMAÇÃO SOFRIDA	TABELA DIMENSIONAL	NOME DA COLUNA DIMENSIONAL
neighborhoods	zone_id	-	Dim_Place	zone_id
zones	name	A coluna no DW corresponde ao nome da zona da qual o bairro faz parte, obtido através de uma junção das tabelas neighborhoods e zones	Dim_Place	zone_name
ride_user	status	Uma seleção dos valores únicos da coluna status da tabela ride_user, representando os *status* possíveis para um pedido, excluindo "*driver*"	Dim_Request_ Status	status_name
rides	routine_id	Seleção dos valores únicos da coluna routine_id da tabela rides	Dim_Routine	routine_id
rides	repeats_until	-	Dim_Routine	repeats_until
rides	week_days	A coluna no DW indica se a rotina se repete às segundas-feiras. É resultado de uma função que lê a coluna week_days e traduz seus dados em valores booleanos para cada dia da semana.	Dim_Routine	is_routine_ monday
rides	week_days	A coluna no DW indica se a rotina se repete às terças-feiras. É resultado de uma função que lê a coluna week_days e traduz seus dados em valores booleanos para cada dia da semana.	Dim_Routine	is_routine_ tuesday

TABELA TRANSACIO-NAL	NOME DA COLUNA TRANSACIO-NAL	TRANSFORMAÇÃO SOFRIDA	TABELA DIMENSIONAL	NOME DA COLUNA DIMENSIONAL
rides	week_days	A coluna no DW indica se a rotina se repete às quartas-feiras. É resultado de uma função que lê a coluna week_days e traduz seus dados em valores booleanos para cada dia da semana.	Dim_Routine	is_routine_ wednesday
rides	week_days	A coluna no DW indica se a rotina se repete às quintas-feiras. É resultado de uma função que lê a coluna week_days e traduz seus dados em valores booleanos para cada dia da semana.	Dim_Routine	is_routine_ thursday
rides	week_days	A coluna no DW indica se a rotina se repete às sextas-feiras. É resultado de uma função que lê a coluna week_days e traduz seus dados em valores booleanos para cada dia da semana.	Dim_Routine	is_routine_ friday
rides	week_days	A coluna no DW indica se a rotina se repete aos sábados. É resultado de uma função que lê a coluna week_days e traduz seus dados em valores booleanos para cada dia da semana.	Dim_Routine	is_routine_ saturday
rides	week_days	A coluna no DW indica se a rotina se repete aos domingos. É resultado de uma função que lê a coluna week_days e traduz seus dados em valores booleanos para cada dia da semana.	Dim_Routine	is_routine_ sunday

TABELA TRANSACIO-NAL	NOME DA COLUNA TRANSACIO-NAL	TRANSFORMAÇÃO SOFRIDA	TABELA DIMENSIONAL	NOME DA COLUNA DIMENSIONAL
`rides`	`week_days`	A coluna no DW contém um texto com os dias da semana em que a rotina se repete, obtido através de um mapeamento dos números que representam os dias da semana na coluna `week_days` da tabela `rides` para os nomes abreviados dos dias da semana (“Seg”, “Ter”, “Qua”, “Qui”, “Sex”, “Sab”, “Dom”).	`Dim_Routine`	`routine_days_description`
`rides`	`id`	Coluna renomeada	`Fact_Ride`	`ride_id`
`rides`	`slots`	Coluna renomeada	`Fact_Ride`	`slots_count`
`messsages`	-	A coluna no DW corresponde à quantidade de mensagens trocadas no chat da carona, obtida através de uma contagem da quantidade de registros por carona na tabela `messages` do banco transacional	`Fact_Ride`	`messages_count`
`ride_user`	-	A coluna no DW corresponde à quantidade de pedidos relativos à carona, obtida através de uma contagem da quantidade de registros por carona na tabela `ride_user` do banco transacional, excluindo os pedidos de *status* “*driver*”	`Fact_Ride`	`requests_count`

TABELA TRANSACIONAL	NOME DA COLUNA TRANSACIONAL	TRANSFORMAÇÃO SOFRIDA	TABELA DIMENSIONAL	NOME DA COLUNA DIMENSIONAL
`ride_user`	`status`	A coluna no DW corresponde à quantidade de pedidos com o *status* "*accepted*" relativos à carona, obtida através de uma contagem da quantidade de registros por carona na tabela `ride_user` do banco transacional onde o *status* é igual a "*accepted*"	`Fact_Ride`	`accepted_ requests_count`
`ride_user`	`status`	A coluna no DW corresponde à quantidade de pedidos com o *status* "*refused*" relativos à carona, obtida através de uma contagem da quantidade de registros por carona na tabela `ride_user` do banco transacional onde o *status* é igual a "*refused*"	`Fact_Ride`	`refused_ requests_count`
`ride_user`	`status`	A coluna no DW corresponde à quantidade de pedidos com o *status* "*pending*" relativos à carona, obtida através de uma contagem da quantidade de registros por carona na tabela `ride_user` do banco transacional onde o *status* é igual a "*pending*"	`Fact_Ride`	`pending_ requests_count`

TABELA TRANSACIO-NAL	NOME DA COLUNA TRANSACIO-NAL	TRANSFORMAÇÃO SOFRIDA	TABELA DIMENSIONAL	NOME DA COLUNA DIMENSIONAL
`ride_user`	`status`	A coluna no DW corresponde à quantidade de pedidos com o *status* "*quit*" relativos à carona, obtida através de uma contagem da quantidade de registros por carona na tabela `ride_user` do banco transacional onde o *status* é igual a "*quit*"	`Fact_Ride`	`quit_ requests_count`
`ride_user`	`id`	Coluna renomeada	`Fact_Ride_ Request`	`request_id`

Fonte: elaboração própria

5.4.1.1 População da Dimensão Sucata `Dim_Ride_Flags`

A tabela `Dim_Ride_Flags`, por constituir uma dimensão sucata, deve ser populada previamente com todas as combinações possíveis das *flags* nela presentes. Como esta dimensão possui 3 *flags* booleanas (além da chave substituta e do atributo de descrição das *flags*), há 2^3 (8) possíveis combinações de valores, logo a tabela terá 8 linhas.

5.4.2 Execução do Processo de ETL

A implementação do processo de ETL foi toda feita em Python, com o uso das seguintes bibliotecas:

- **pandas:** biblioteca de manipulação e análise de dados, utilizada para o tratamento dos dados e a transformação das consultas SQL textuais em dataframes, para que possam ser executadas.
- **psycopg2:** biblioteca adaptadora de base de dados PostgreSQL, utilizada para conectar ao banco de dados transacional e executar consultas SQL sobre ele.
- **datetime:** biblioteca com classes para manipulação de data e hora, utilizada para lidar com datas e horas nas dimensões temporais.

- **itertools:** biblioteca com funções para a criação de iteradores com loops eficientes, utilizada para popular a `Dim_Ride_Flags` com todas as combinações logicamente válidas de flags.

A seguir são descritas as etapas do fluxo de ETL relativas a cada tabela de destino no DW, construídas através de comandos SQL executados com psycopg2 e de funções Pandas. Nota-se que, pelo aspecto da carga incremental, sempre são extraídos apenas aqueles dados criados ou modificados após o momento da última extração. Os comandos SQL utilizados estão descritos no Apêndice A.

1. Criação de `Dim_Date`
    1.1. Geração de lista com datas de 01/04/2016 até 31/12/2035
    1.2. Geração dos outros atributos da tabela baseados nos valores da lista
    1.3. Carga em `Dim_Date`

2. Criação de `Dim_Hour`
    2.1. Geração de lista com todas as horas e minutos do dia (00:00 até 23:59)
    2.2. Geração dos outros atributos da tabela baseados nos valores da lista
    2.3. Carga em `Dim_Hour`

3. Criação de `Dim_User`:
    3.1. Seleção de colunas relevantes de uma junção da tabela `users` com a tabela `institutions`
    3.2. Limpeza e padronização
    3.3. Carga em `Dim_User`

4. Criação de `Dim_Place`:
    4.1. Seleção de colunas relevantes de uma junção das tabelas `hubs`, `campi` e `institutions`
    4.2. Seleção de colunas relevantes de uma junção das tabelas `neighborhoods` e `zones`
    4.3. Concatenação dessas duas seleções com Pandas
    4.4. Limpeza e padronização

4.5. Carga em `Dim_Place`

5. Criação de `Dim_Request_Status`:
   5.1. Seleção de valores distintos da coluna `status` da tabela `ride_user`, excluindo o *status* "*driver*"
   5.2. Inserção de descrições sobre cada *status*
   5.3. Carga em `Dim_Request_Status`

6. Criação de `Dim_Routine`:
   6.1. Seleção de conjuntos distintos de valores de `routine_id`, `repeats_until` e `week_days` da tabela `rides`, em que `routine_id` e `repeats_until` não são nulos
   6.2. Transformação dos dados de `week_days` em 7 colunas de *flags* booleanas, cada uma com valor *true* se a rotina se repete naquele dia ou *false* se não se repete
   6.3. Criação de uma coluna com a descrição textual dos dias em que a rotina acontece
   6.4. Limpeza e padronização
   6.5. Carga em `Dim_Routine`

7. Criação de `Dim_Ride_Flags`:
   7.1. Definição das *flags* da dimensão sucata
   7.2. Iteração para gerar todas as combinações possíveis das *flags* booleanas e construir a descrição textual de cada registro
   7.3. Carga em `Dim_Ride_Flags`

8. Criação de `Fact_Ride`:
   8.1. Seleção de colunas relevantes de uma junção das tabelas `rides`, `ride_user` e `messages_count`
      8.1.1. De `ride_user` é trazido para `Fact_Ride` o id do motorista de cada carona com o nome `driver_id`

8.1.2. `messages_count` é uma tabela criada a partir de `messages`, com a quantidade de mensagens enviadas no chat de cada carona

8.2. Seleção de dados dos pedidos de cada carona de `ride_user`, com id da carona e *status* do pedido

8.3. Agregação dos dados dos pedidos (8.2) com Pandas, resultando em um *dataframe* com a quantidade de pedidos de cada *status* para cada carona

8.4. *Merge* com Pandas dos dados agregados de pedidos (8.3) com os dados das caronas (8.1)

8.5. Limpeza e padronização

8.6. Mapeamento de chaves substitutas nas dimensões `Dim_Date` e `Dim_Hour`, a partir das colunas `created_at` (chave indicando a data de criação da carona em `Dim_Date` e chave indicando a hora de criação da carona em `Dim_Hour`) e `date` (chave indicando a data programada para a ocorrência da carona em `Dim_Date` e chave indicando a hora programada para a ocorrência da carona em `Dim_Hour`) da tabela `rides` do banco transacional

8.7. Mapeamento de chave substituta na dimensão `Dim_Ride_Flags` que representa a combinação de flags booleanas relativa à carona, a partir das colunas `going`, `done` e `deleted_at` da tabela `rides` do banco transacional

8.8. Mapeamento de chave substituta na dimensão `Dim_Routine`, a partir da coluna `routine_id` da tabela `rides` do banco transacional. Para as caronas com `routine_id` nulo (que não fazem parte de rotina), mapeia-se a chave para `routine_sk` = 0, valor específico para caronas não rotineiras.

8.9. Mapeamento de chave substituta na dimensão `Dim_User`, a partir da coluna `driver_id` da seleção de dados de caronas (8.1).

8.10. Mapeamento de chave substituta na dimensão `Dim_Place`. Para `Dim_Place` inicialmente há uma chave para a chave substituta do bairro e uma chave para a chave substituta do pólo universitário da carona (lembrando que toda carona necessariamente tem um bairro e um pólo universitário atrelados a ela, um dos dois como ponto de origem e o outro como ponto de destino). Porém, como é mais interessante apontar qual é o ponto de origem e de destino da carona, utiliza-se a coluna `going` do banco transacional para

entender se a carona estava indo ou voltando da universidade, assim esclarecendo o ponto de origem e o de destino. Daí surgem as colunas `place_origin_sk` (apontando para o ponto de origem da carona em `Dim_Place`, seja ele bairro ou pólo universitário) e `place_destination_sk` (apontando para o ponto de destino da carona em `Dim_Place`, seja ele bairro ou pólo universitário).

8.11. Carga em `Fact_Ride`

9. Criação de `Fact_Ride_Request`:
   9.1. Seleção de colunas relevantes de `ride_user`, excluindo as linhas em que `status` = '*driver*'
   9.2. Limpeza e padronização
   9.3. Mapeamento de chaves substitutas nas dimensões `Dim_Date` e `Dim_Hour`, a partir das colunas `created_at` (chave indicando a data de criação do pedido em `Dim_Date` e chave indicando a hora de criação do pedido em `Dim_Hour`) e `updated_at` (chave indicando a data da última atualização do pedido em `Dim_Date` e chave indicando a hora da última atualização do pedido em `Dim_Hour`) da tabela `ride_user` do banco transacional.
   9.4. Mapeamento de chaves substitutas nas dimensões `Dim_User` e `Dim_Request_Status` e na tabela fato `Fact_Ride`
   9.5. Carga em `Fact_Ride_Request`

5.4.3 Estrutura do Repositório

Estão presentes na pasta raíz do repositório os seguintes scripts:

- **config.py:** contém as configurações necessárias para que a biblioteca psycopg2 se conecte ao banco transacional e ao data warehouse no PostgreSQL.
- **etl_main.py:** script principal que coordena o processo de ETL para o data warehouse, chamando as funções nos outros scripts. O processo de ETL é iniciado ao rodar este script.
- **sql_queries.py:** contém as queries de criação e de deleção das tabelas do data warehouse.

- **utils.py:** contém funções auxiliares.

Além disso, há um arquivo de texto: last_etl_run.txt, que guarda a data e hora em que o processo de ETL foi executado pela última vez.

Há, também, duas pastas: uma para os scripts de ETL das dimensões (dim_scripts), e outra para os scripts de ETL das tabelas fato (fact_scripts). Na pasta **dim_scripts**, estão presentes os seguintes scripts:

- dim_date_etl.py
- dim_hour_etl.py
- dim_place_etl.py
- dim_request_status_etl.py
- dim_ride_flags_etl.py
- dim_routine_etl.py
- dim_user_etl.py

Na pasta **fact_scripts**, estão presentes os seguintes scripts:

- fact_ride_etl.py
- fact_ride_request_etl.py

5.4.4 Descrição do Fluxo do Código

1. **Início:** o script **etl_main.py** é executado, chamando primeiramente a função **main_etl_process**, que coordena o processo de ETL como um todo. Se o atributo "carga_completa" dessa função for verdadeiro, será feita uma carga completa, com todos os dados relevantes do banco transacional. Se for falso, será feita uma carga incremental, trazendo apenas os dados que foram inseridos ou atualizados depois da data e hora em **last_etl_run.txt**. A opção padrão é ter essa variável como falsa, porém é interessante manter a opção da carga completa para situações de teste ou de carga problemática no DW.
2. **Limpeza:** se "carga_completa" for igual a *true*, o script apaga o arquivo **last_run_file.txt**, que armazena a data e hora da última execução, e utiliza uma data antiga padrão como referência, assim forçando uma carga completa.
3. **Conexão:** em seguida, ele estabelece conexões com os bancos de dados transacional e dimensional, através da função **connect_to_db** do script **utils.py** e das configurações para a conexão nos bancos no script **config.py**.

4. **Criação ou Limpeza de Tabelas do DW:** a função **create_or_reset_dw_tables** é responsável por criar todas as tabelas do data warehouse se elas não existirem, ou truncar as tabelas (apagar seus dados) se elas já existirem, executando as queries de criação e de truncamento das tabelas do script **sql_queries.py**. Ela só é executada se "carga_completa" for igual a *true*. Isso garante um ambiente limpo para cada execução completa do ETL.
5. **Inserção de Membros "Desconhecidos":** a função **insert_all_unknown_dim_members** insere um registro "Desconhecido" em todas as tabelas de dimensão, com valores padrão de desconhecido para cada atributo, sempre com a chave primária da dimensão (chave substituta que termina com "_sk") igual a -1. Isso é crucial para garantir a integridade referencial em caso de dados ausentes ou inconsistentes na tabela fato, garantindo que não existam chaves estrangeiras nulas.
6. **Carga das Dimensões:** o script executa o ETL para cada tabela de dimensão, chamando as funções principais dos scripts da pasta **dim_scripts**.
7. **Carga dos Fatos:** após carregar as dimensões, o script executa o ETL para as tabelas de fatos (`Fact_Ride` e `Fact_Ride_Request`), chamando as funções principais dos scripts da pasta **fact_scripts**. Se "carga_completa" for igual a *false*, ele processa apenas os dados que foram adicionados ou modificados desde a data e hora da última execução. Se "carga_completa" for *true*, ele processa tudo.
    7.1. Aqui, no ETL de `Fact_Ride`, é executada a função **derive_and_lookup_flags**, definida em **dim_ride_flags_etl.py**. Ela pega os dados relevantes de `rides` para descobrir qual `ride_flags_sk` de `Dim_Ride_Flags` tem o registro que corresponde à combinação correta de informações da carona. Como mencionado na seção 5.4.1.1, `Dim_Ride_Flags` contém todas as combinações possíveis de suas *flags*. As *flags* da dimensão sucata são:
        7.1.1. `done` (indica se a carona foi finalizada): informação extraída diretamente da coluna `done` da tabela `rides`.
        7.1.2. `is_going_to_campus` (indica se a carona estava indo ou voltando do campus): informação extraída diretamente da coluna `going` da tabela `rides`.
        7.1.3. `deleted` (indica se a carona foi deletada): informação extraída da coluna `deleted_at` da tabela `rides`, que contém a data em que a

carona foi deletada, se ela houver sido deletada, ou nulo, se ela não houver sido deletada. Logo, o valor de `deleted` é verdadeiro quando há uma data de deleção da carona em `deleted_at`, e falso quando há o valor nulo.

8. **Finalização:** No final do processo, a data e hora atuais são gravadas no arquivo last_etl_run.txt, marcando o ponto de partida para a próxima carga incremental. As conexões com os bancos de dados são fechadas para liberar recursos.

A implementação do rastreamento de mudanças nas dimensões, como definido por Barbieri (2011), está fora do escopo do trabalho.

5.5 NOVAS DEFINIÇÕES PARA AS ANÁLISES

Apesar do estudo de 2021 ter desconsiderado os valores da coluna `done` da tabela `rides` do banco transacional para entender quais caronas realmente aconteceram, um novo olhar sobre essa métrica revelou que ela pode, afinal de contas, ser útil. Foi aferido durante a produção do presente trabalho que as caronas com valor da coluna igual a *true* constituem 95,3% das caronas com pelo menos uma pessoa aceita.

Além disso, é notável a potencial supernotificação de caronas válidas com o uso da métrica anterior, dado que ela considera caronas apenas com pedidos pendentes, nunca respondidos pelo motorista, como caronas válidas.

Logo, a fim de utilizar uma métrica mais conservadora e de maior confiabilidade aparente, são consideradas válidas, neste novo estudo, caronas que têm valor verdadeiro na coluna `done`, e que tiveram pelo menos um usuário aceito.

Complementarmente, é definida como carona finalizada qualquer carona que tenha o valor verdadeiro na coluna `done` em sua linha da tabela `rides`.

Outra definição que foi considerada interessante de ser mudada é a de usuários ativos. No estudo de 2021, um usuário foi considerado ativo durante um período de tempo se ele, durante esse período, criou uma carona, ou pediu para entrar em uma carona e foi aceito ou ficou pendente. Porém, levando em consideração o fato de que os usuários que pediram para entrar em uma carona e foram recusados ou desistiram de fato interagiram com o sistema de busca de caronas, são considerados esses também usuários ativos, assim abrangendo todos

aqueles que criaram uma carona ou pediram para entrar em uma carona, independente da resposta do pedido.

Foi criada, então, uma nova definição para discutir os usuários que a princípio participaram de fato de caronas: usuários impactados. Esses são os usuários que deram caronas válidas ou foram aceitos em caronas válidas. Assim, há duas métricas para entender os usuários do aplicativo em dois níveis diferentes de engajamento.

5.6 VISUALIZAÇÕES E ANÁLISES ATRAVÉS DO TABLEAU

Para que seja possível analisar de forma visual e clara os dados do data warehouse, foi criado um painel gerencial interativo no Tableau, extraindo os dados diretamente dele. Utilizando como base as análises da seção 4, o painel construído tira proveito da interatividade inerente às plataformas de Self-Service BI ao juntar gráficos correlatos em uma página, e disponibilizar filtros para possibilitar análises interseccionais. Assim, o usuário desenvolve uma visão mais profunda dos dados, podendo fatiá-los de diferentes formas.

É importante acentuar algumas características do Tableau que o tornam superior ao Power BI — ferramenta escolhida para o estudo de 2021 — para o desenvolvimento do painel analítico.

O Power BI é parte integrante do ecossistema da Microsoft, o que lhe confere uma vantagem significativa quando integrado com outras ferramentas da suíte Microsoft, como o Excel. Sua arquitetura também se beneficia do motor de modelagem de dados do Power Pivot e da linguagem DAX.

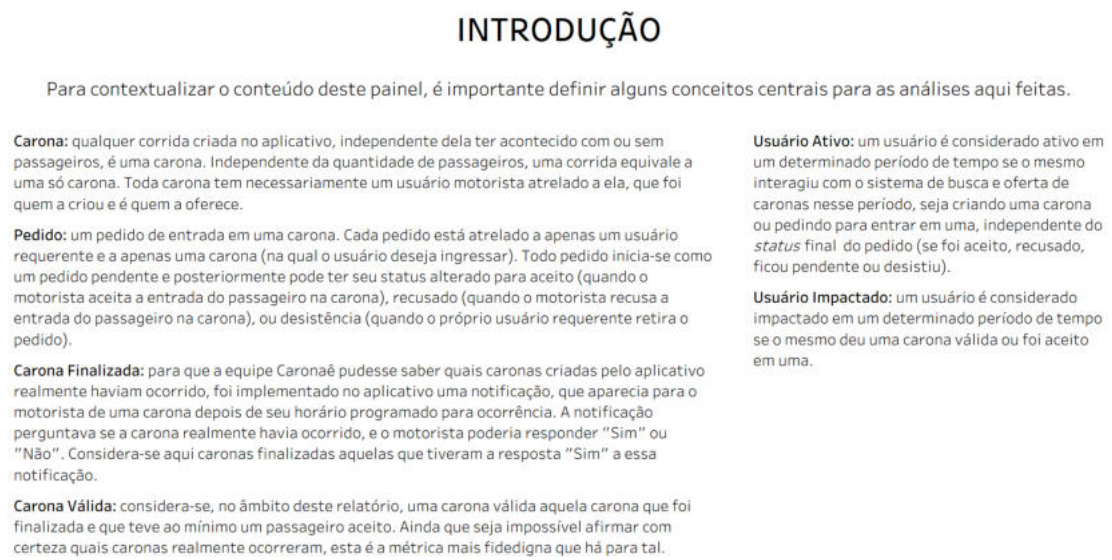
O Tableau, por sua vez, é uma ferramenta agnóstica em relação a fornecedores. Sua força reside em sua capacidade de conectar-se a uma vasta gama de fontes de dados de forma universal, desde arquivos locais até grandes bancos de dados em nuvem, sem estar vinculado a um ecossistema específico. Essa característica é a ideal para o presente trabalho, pois são utilizados dados provenientes do PostgreSQL. Além disso, a linguagem nativa de fórmulas do Tableau é mais intuitiva para usuários de SQL, em contraste com o DAX, utilizado nos produtos da Microsoft, que tem uma gramática mais própria.

O Tableau oferece, também, uma experiência de arrastar e soltar que torna a criação de gráficos e dashboards interativos uma tarefa extremamente intuitiva e ágil. Ademais, possui uma curva de aprendizado menos íngreme que a do Power BI, o que o torna a ferramenta preferida para a exploração de dados ad-hoc e a criação de painéis focados em narrativas visuais.

5.6.1 Introdução

A primeira página do painel gerencial (figura 22) é uma introdução, que define para o usuário conceitos importantes para as análises. Os conceitos definidos são os de carona, pedido, carona finalizada, carona válida, usuário ativo e usuário impactado, todos definidos também no texto do presente trabalho.

Figura 22: Página introdutória do painel gerencial feito no Tableau em 2025



Fonte: elaboração própria

5.6.2 Números Gerais de Caronas

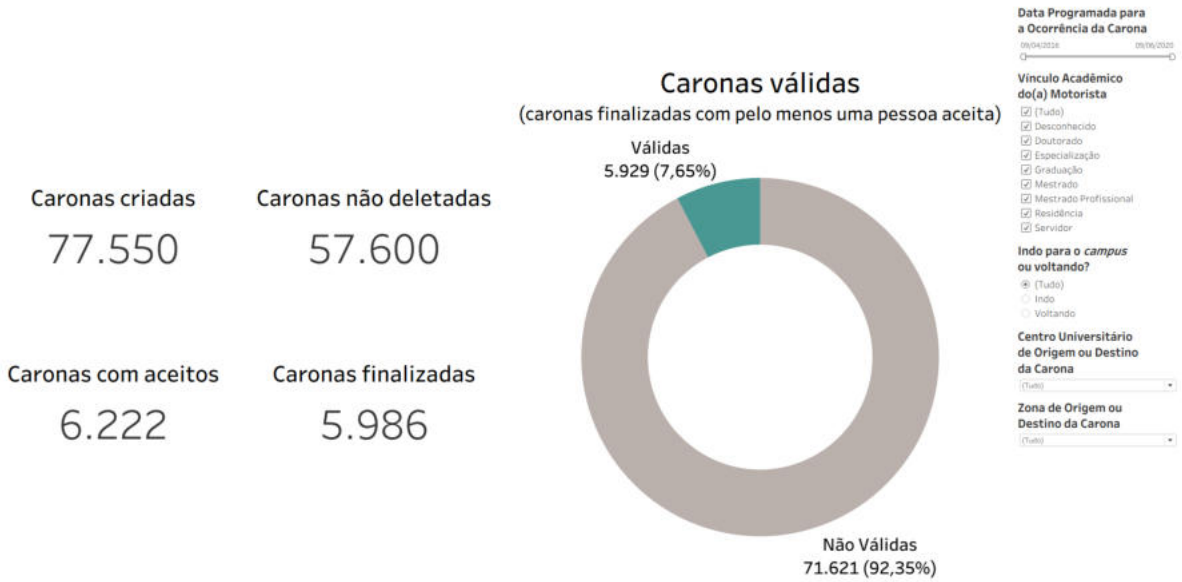
A página representada na figura 23 se assemelha bastante a sua equivalente do trabalho de 2021 (figura 7), porém troca o medidor por um gráfico de rosca, e adiciona alguns filtros, para que o usuário possa investigar esses números gerais de caronas sob diferentes lentes.

O primeiro filtro é um filtro de data, no qual o usuário escolhe um período de análise, e todas as caronas com data programada de ocorrência neste período são contempladas. O segundo filtro permite que o usuário veja os números pelo vínculo acadêmico do motorista, podendo escolher mais de um. O terceiro filtro permite a visualização dos números só considerando as caronas que estavam indo para o campus ou voltando dele. O quarto filtro dá ao usuário a possibilidade de escolher ver os números de caronas com um centro universitário

específico como ponto de origem/destino, e o quinto filtro faz o mesmo para as zonas compreendidas pelo projeto.

Por causa da abordagem mais conservadora, há agora, na figura 23, menos caronas consideradas válidas do que no estudo de 2021: um total de 5.927, ou 7,65% das caronas criadas.

Figura 23: Página de números gerais dos dados de caronas do painel gerencial feito no Tableau em 2025

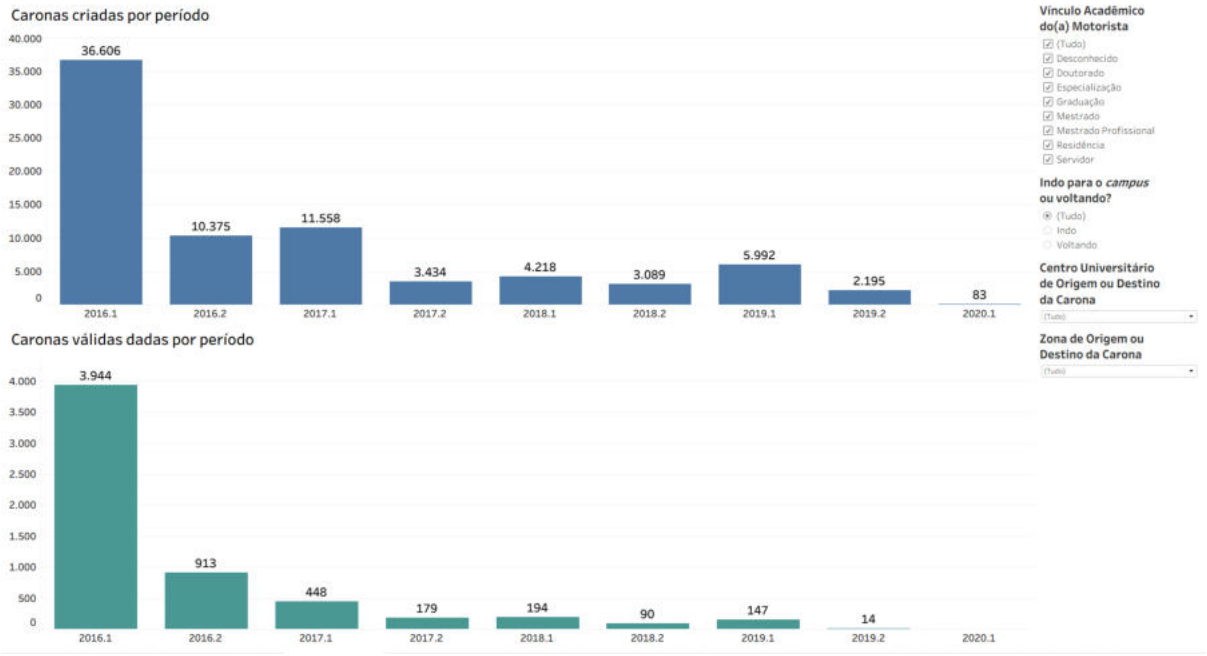


Fonte: elaboração própria

5.6.3 Visão Temporal de Caronas Criadas e Caronas Válidas Dadas

Para a visualização da figura 24, juntou-se os dois gráficos de caronas por período na mesma página e foram adicionados alguns filtros. O primeiro filtro permite que o usuário veja os números pelo vínculo acadêmico do motorista, podendo escolher mais de um. O segundo filtro permite a visualização dos números só considerando as caronas que estavam indo para o campus ou voltando dele. O terceiro filtro dá ao usuário a possibilidade de escolher ver os números de caronas com um centro universitário específico como ponto de origem/destino, e o quarto filtro faz o mesmo para as zonas compreendidas pelo projeto.

Figura 24: Página de caronas criadas por período do painel gerencial feito no Tableau em 2025



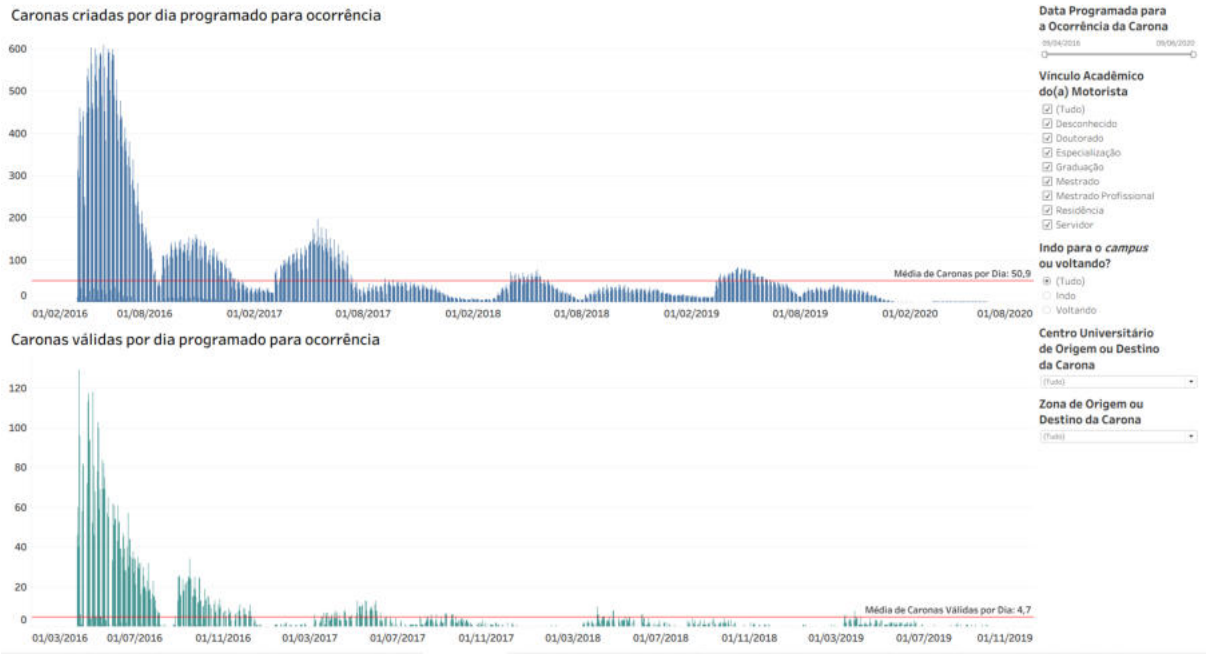
Fonte: elaboração própria

Para a criação dos gráficos da figura 25, que operam no nível do dia, não é necessário o auxílio do Google Planilhas como na primeira solução analítica (figuras 10 e 11), pois a flexibilidade do Tableau permite a criação de gráficos com colunas mais finas.

Esta página também conta com filtros. O primeiro filtro é um filtro de data, no qual o usuário escolhe um período de análise, e todas as caronas com data programada de ocorrência neste período são contempladas. O segundo filtro permite que o usuário veja os números pelo vínculo acadêmico do motorista, podendo escolher mais de um. O terceiro filtro permite a visualização dos números só considerando as caronas que estavam indo para o campus ou voltando dele. O quarto filtro dá ao usuário a possibilidade de escolher ver os números de caronas com um centro universitário específico como ponto de origem/destino, e o quinto filtro faz o mesmo para as zonas compreendidas pelo projeto.

Além disso, o Tableau permite a adição de uma linha de média, indicando quantas caronas (ou caronas válidas, para o gráfico de baixo), tiveram seu dia programado para ocorrência no período selecionado.

Figura 25: Página de caronas por dia programado para ocorrência do painel gerencial feito no Tableau em 2025



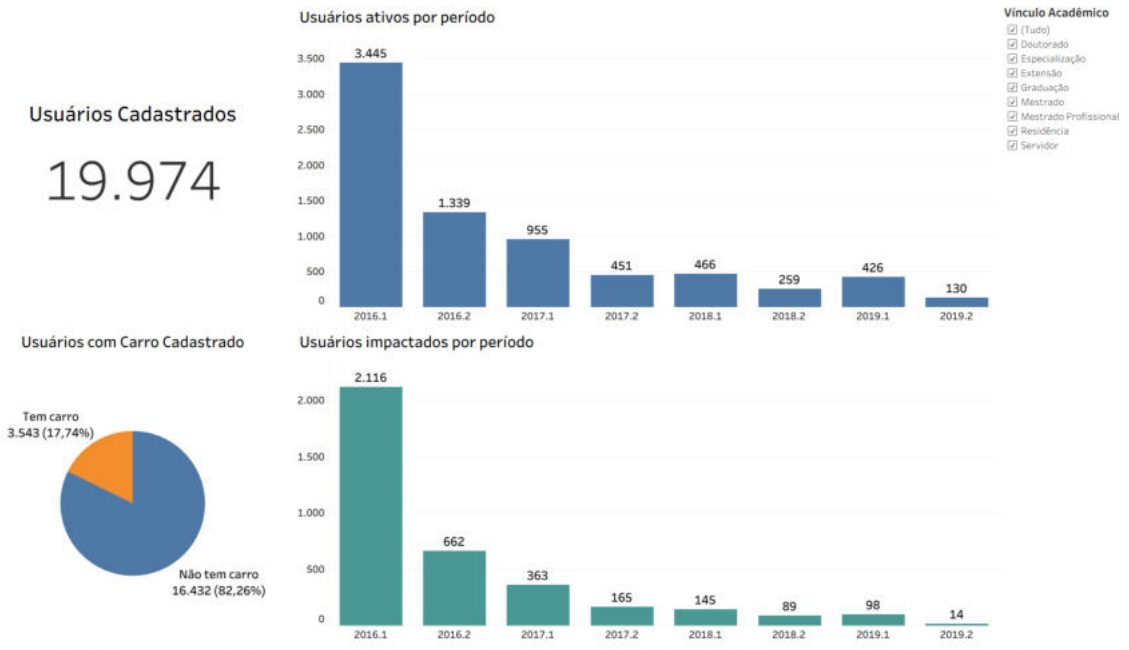
Fonte: elaboração própria

5.6.4 Usuários Ativos e Impactados

A figura 26 traz dois dados que não constavam no estudo de 2021: o de número total de usuários cadastrados e o de proporção de usuários com e sem carro. Além disso, há os dados de usuários ativos e impactados por período. Há a possibilidade de filtrar, nesta página, pelo vínculo acadêmico dos usuários, sendo possível escolher mais de um.

É possível ver as novas definições de usuários ativos e de usuários impactados na prática, com o novo gráfico de usuários impactados e com a diferença nos valores entre o gráfico de usuários ativos da figura 26 e sua visualização equivalente do estudo de 2021 (figura 12). Houve, no total, de acordo com as novas definições, 5.286 usuários ativos e 2.842 usuários impactados ao longo do período de funcionamento do aplicativo. A parcela de ativos corresponde a 26,5% do total de usuários cadastrados e a de usuários impactados corresponde a 14,2% desse mesmo total.

Figura 26: Página de usuários do painel gerencial feito no Tableau em 2025



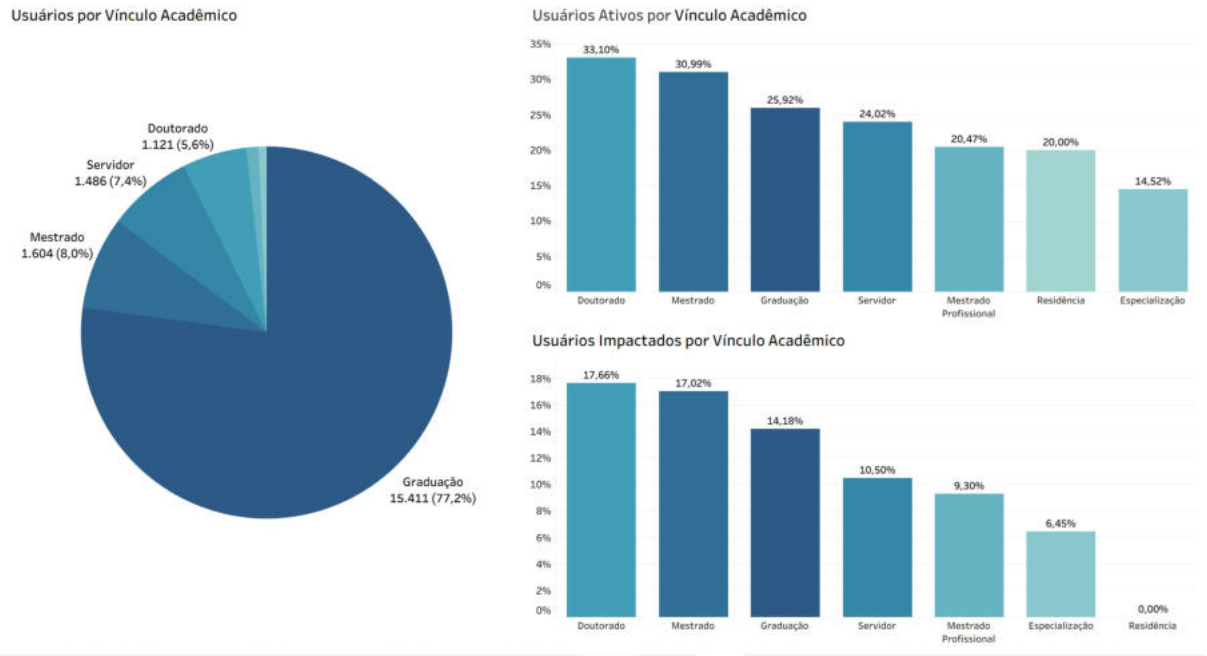
Fonte: elaboração própria

5.6.5 Distribuição de Usuários por Vínculo Acadêmico

A página ilustrada na figura 27 traz dados de usuários por vínculo acadêmico, usuários ativos por vínculo acadêmico, e o novo dado de usuários impactados por vínculo acadêmico. Nos gráficos de barras, é possível ver novamente a diferença entre as duas novas métricas de engajamento, aqui categorizadas por vínculo acadêmico do usuário.

Obs: a categoria “Extensão” foi excluída dos gráficos de usuários ativos e impactados por vínculo acadêmico pois o seu percentual nos dois gráficos era muito superior ao das outras categorias (66,67% no de usuários ativos e 33,33% no de usuários impactados), não por essa categoria ter muitos usuários ativos/impactados e sim porque existem apenas três usuários desta categoria, dos quais dois foram ativos e um foi impactado. Logo, mostrá-la com um percentual tão maior que os outros poderia levar a conclusões errôneas.

Figura 27: Página de análise por vínculo acadêmico do painel gerencial feito no Tableau em 2025



Fonte: elaboração própria

5.6.6 Ocupação Média

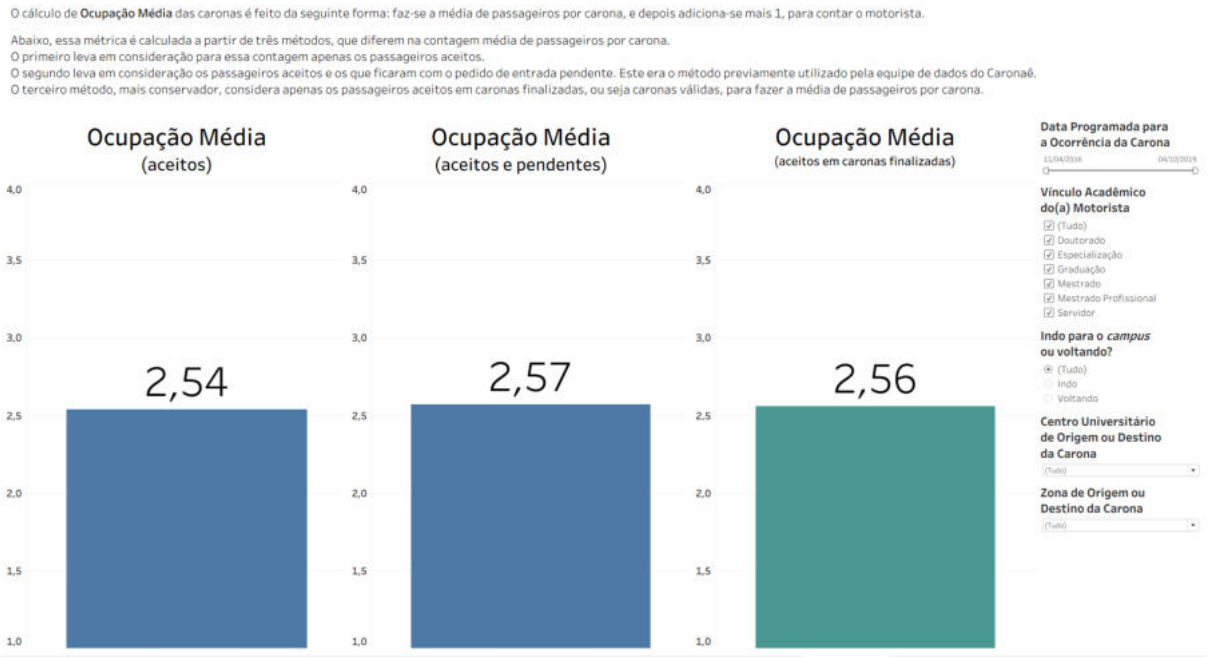
A figura 28 mostra a página que traz dados de ocupação média. Há, no topo da página, uma pequena explicação sobre o cálculo da ocupação média de acordo com os três métodos apresentados. Os dois primeiros — o que leva em consideração apenas os passageiros aceitos e o que leva em consideração aceitos e pendentes — já haviam sido utilizados no estudo de 2021. O terceiro, criado para o estudo atual, leva em consideração apenas passageiros aceitos em caronas finalizadas (na prática, apenas caronas válidas, pois uma carona finalizada com algum aceito é uma carona válida). Assim se mantém a perspectiva mais conservadora introduzida com o novo conceito de caronas válidas. Os medidores foram trocados por gráficos de barra, para que a diferença entre os valores fosse percebida com mais facilidade.

Há cinco filtros nesta página. O primeiro filtro é um filtro de data, no qual o usuário escolhe um período de análise, e todas as caronas com data programada de ocorrência neste período são contempladas. O segundo filtro permite que o usuário veja os números pelo vínculo acadêmico do motorista, podendo escolher mais de um. O terceiro filtro permite a visualização dos números só considerando as caronas que estavam indo para o campus ou voltando dele. O quarto filtro dá ao usuário a possibilidade de escolher ver os números de

caronas com um centro universitário específico como ponto de origem/destino, e o quinto filtro faz o mesmo para as zonas compreendidas pelo projeto.

É interessante notar que a ocupação média das caronas considerando a média de usuários aceitos como passageiros em caronas que foram finalizadas (+1 pela inclusão do motorista) na figura 28 transita na mesma faixa que as duas métricas anteriormente propostas.

Figura 28: Página de ocupação média do painel gerencial feito no Tableau em 2025



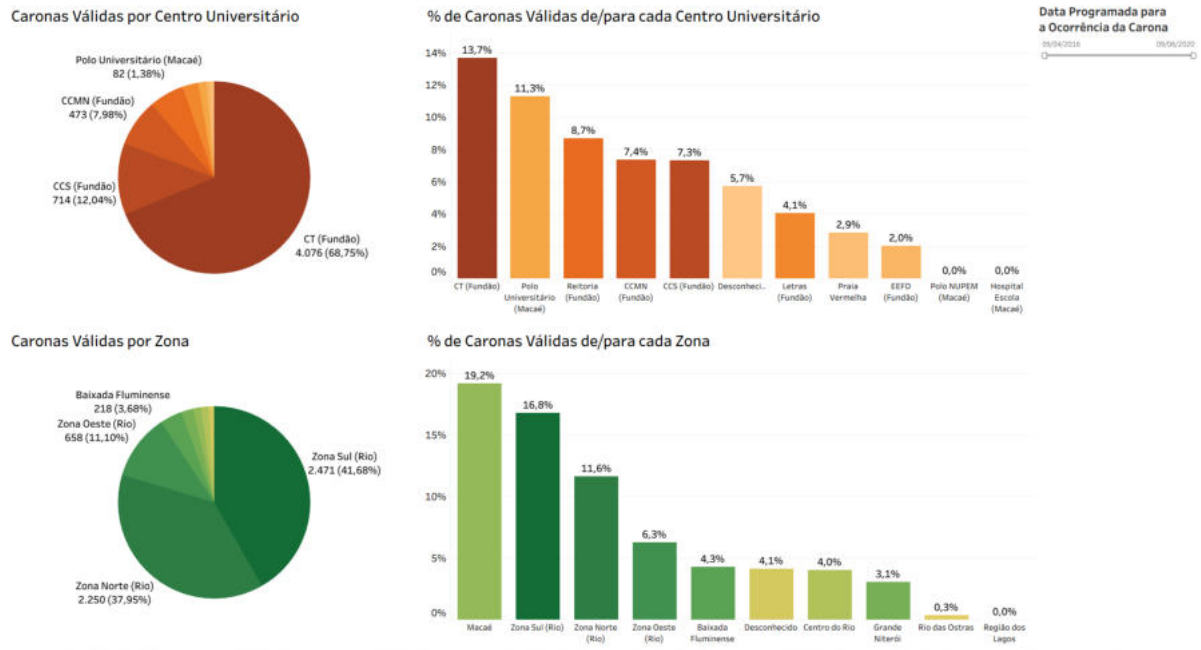
Fonte: elaboração própria

5.6.7 Distribuição de Caronas por Centros e Zonas

Na figura 29, é possível ver a página que junta os dados de centros universitários e de zonas compreendidas pelo projeto. Além dos gráficos de caronas válidas por centro e por zona, e de % de caronas válidas para cada zona, há também um novo gráfico de caronas válidas por centro. Há um filtro de data na página, no qual o usuário escolhe um período de análise, e todas as caronas com data programada de ocorrência neste período são contempladas.

Nos gráficos de pizza, apesar da mudança da métrica de caronas válidas e subsequente diminuição de seu número, a proporção de caronas válidas por centros e zonas não mudou muito. Já no gráfico de % de caronas válidas de/para cada zona, a nova definição de caronas válidas fez bastante diferença, fazendo Macaé superar a Zona Sul como a zona com maior porcentagem de caronas válidas.

Figura 29: Página de caronas válidas por centro e zona do painel gerencial feito no Tableau em 2025



Fonte: elaboração própria

5.7 DISCUSSÃO DAS DUAS SOLUÇÕES

Cabe, então, uma comparação das duas soluções aqui apresentadas: a solução de 2021, que extraía dados diretamente do banco transacional para o Power BI, e a atual, que se fundamenta sobre um data warehouse, tem um processo de ETL integrado e utiliza o Tableau como ferramenta de *Business Intelligence*.

Como descrito na seção 2.1.4, são muitas as vantagens da criação de um data warehouse para execução de operações OLAP: desde a preservação da eficiência das operações OLTP, ao desempenho analítico superior oriundo da desnormalização inerente do esquema estrela, até a definição da etapa de tratamento do ETL, em que os dados são limpos e padronizados para garantir sua integridade referencial e qualidade no DW.

Além disso, por mais que o banco transacional do Caronaê atualmente contenha dados de todo o histórico de funcionamento do aplicativo, a ideia desse tipo de banco é ser como um *snapshot* do negócio: ele não deve necessariamente conter a totalidade dos dados históricos, pois não é projetado para tal. Sua função é manter o dia-a-dia da operação funcionando de forma eficiente. Logo, quanto mais o negócio crescer, menos prático será manter todo o histórico no banco transacional, e esse histórico é necessário para o BI. A

natureza do data warehouse de ser um armazém para grandes volumes de dados históricos permite análises de tendências, comparações e previsões que são impossíveis de realizar em um banco de dados transacional.

Por fim, como descrito na seção 5.6, o Tableau representa algumas vantagens sobre o Power BI, desde a sua linguagem nativa de fórmulas que é mais intuitiva para usuários SQL do que o DAX (linguagem do Power BI), ao fato dele ser agnóstico a fornecedores, podendo se conectar com uma grande variedade de fontes de dados, até sua experiência de usuário *"drag and drop*", mais intuitiva e ágil que a de outras plataformas de Self-Service BI.

A experiência do usuário com o painel foi avaliada através de um experimento qualitativo com foco na usabilidade da versão final. O painel foi apresentado pronto para os usuários, todos membros ou ex-membros da equipe Caronaê. Eles não construíram seus próprios relatórios, porém tiveram a oportunidade de interagir com o painel navegando entre as páginas e utilizando os filtros de cada página. Em seguida são apresentados a avaliação qualitativa e seus principais resultados.

5.8 AVALIAÇÃO QUALITATIVA

Para que os usuários pudessem interagir com o painel gerencial, o mesmo foi disponibilizado através da plataforma Tableau Public[9] de compartilhamento e visualização de painéis de dados, parte da suíte de ferramentas do Tableau. Além disso, foi criado um formulário na plataforma Google Forms[10], com o intuito de coletar informações sobre a experiência dos usuários com o painel. As perguntas presentes no formulário estão disponíveis no Apêndice B.

Foram obtidas 7 respostas ao formulário. A maioria dos usuários que responderam ao formulário avaliaram sua experiência geral com o painel Analisaê como "Ótima", elogiando a interatividade e a facilidade de uso do painel. A maioria dos respondentes também classificou o painel como "intuitivo e fácil de usar" e "visualmente agradável". Todos os usuários definiram as informações mostradas no painel como "úteis e/ou interessantes". As respostas dos usuários foram disponibilizadas no Apêndice C.

No formulário foram destacados pontos positivos do painel, como:

- **Riqueza e Profundidade das Análises:** a ampla quantidade e variedade de métricas apresentadas foi altamente valorizada, em especial as análises interseccionais (ex:

[9] https://www.tableau.com/pt-br/community/public
[10] https://workspace.google.com/intl/pt-BR/products/forms

usuários ativos por nível acadêmico) e o detalhamento dos dados de caronas por locais (centros da UFRJ e zonas da cidade), permitindo um entendimento mais completo do público e uso do Caronaê.

- **Alta Interatividade e Flexibilidade:** a possibilidade de manipular os dados ativamente, com destaque para a facilidade de filtragem na tela de "Números gerais", permitiu aos usuários isolar contextos específicos para uma análise mais detalhada (ex: caronas de/para alguma zona ou centro).
- **Clareza e Organização da Informação:** a clareza das informações e a organização do conteúdo foram elogiadas, com o agrupamento em abas e a intuitividade para manipulação como fatores que facilitaram a navegação e a compreensão geral do painel.
- **Estética e Apresentação Visual:** a escolha de elementos visuais, em particular o uso de cores em gradiente, foi destacada por ser agradável e contribuir positivamente para a apresentação das análises.
- **Facilidade de Uso em Diversos Navegadores:** foi pontuado em específico o bom funcionamento do painel no Safari, navegador nativo dos produtos Apple.

Também foram mencionados alguns pontos de melhoria para o painel, como:

- **Usabilidade de Filtros e Interação com Elementos:** o filtro de data foi visto por usuário como “não prático” para o uso preciso, pois ele mostra todo o período de funcionamento do aplicativo em seu controle deslizante e pode ser difícil filtrar para períodos pequenos.
- **Visualização e Design do Painel:** o painel foi percebido como um pouco "frio e impessoal" devido à falta de elementos da identidade visual do projeto e à ausência de uma “visão de designer". Houve também um problema de layout em telas menores (15 polegadas), onde o Tableau se apresentou recortado, sugerindo a necessidade de otimização da fonte ou do design para diferentes resoluções. Isso pode ser explicado pela falta de opções de responsividade do Tableau. A ferramenta oferece as opções de janela fixa ou adaptável para a visualização. A janela fixa não se adapta a diferentes resoluções, podendo ficar pequena demais em uma tela grande ou grande demais em uma tela pequena. Já a janela adaptável tem um nível de responsividade muito limitado. Há a opção de criar layouts diferentes para diferentes dispositivos, porém não há como diferenciar entre desktops e laptops. A opção escolhida foi a janela

adaptável, porém ela pode prejudicar a visualização em telas menores como telas de laptops.

- **Falta de Dados e Métricas Específicas sobre Oferta/Motoristas:** um usuário sentiu falta de métricas que liguem o potencial de motoristas com a oferta real de caronas. Especificamente, pontuou que não é possível saber a proporção de usuários com carro cadastrado que realmente criaram caronas, ou fazer um comparativo claro entre a demanda de passageiros e a oferta de caronas.
- **Desempenho (Tempo de Carregamento):** o tempo de *loading* percebido ao mudar de guia ou ao selecionar uma opção de filtro foi destacado como um ponto negativo na experiência de uso. Isso pode ser explicado pela opção do autor do trabalho em utilizar filtros que apenas mostrassem valores relevantes. Isso significa que após a seleção de um valor e um filtro, os os outros filtros mostram apenas os valores que restam após a primeira filtragem. Isso é algo que deixa o painel mais lento, porém foi entendido pelo autor como algo importante para que, após uma primeira filtragem, não fossem mostrados valores em outros filtros que na realidade ao serem escolhidos em conjunto com o primeiro não mostrariam dado algum.
- **Incompletude da Informação sobre Caronas Não Válidas:** um usuário sentiu falta de um detalhamento sobre os motivos de caronas não serem válidas. Este questionou se um desses motivos poderia ser a ocorrência de falhas por questões técnicas do aplicativo. Infelizmente, porém, não há como saber os reais motivos de caronas do passado não serem válidas. Toda a informação disponível para inferir a validade de uma carona foi utilizada no presente trabalho.
- **Consistência Visual (Cores):** um usuário observou a falta de uso das cores pré-atribuídas às zonas, presentes no banco transacional, o que poderia ter sido utilizado para melhor reconhecimento e consistência visual no painel.

Em relação a erros ou problemas técnicos, foi mencionada novamente a lentidão do painel e sua baixa responsividade, especialmente para telas menores. Sobre funcionalidades que atualmente não estão presentes no painel que os usuários gostariam de ver, foram pontuadas:

- **Melhoria na Usabilidade do Filtro Temporal:** a principal demanda em relação ao filtro de data é torná-lo mais prático e preciso. A sugestão é implementar um filtro com opções mais claras de seleção (como dropdown de mês/ano/período) ou um campo de texto para inserção direta do período desejado.

- **Integração de Métricas de Sustentabilidade e Impacto Ambiental:** há um forte interesse em incluir cálculos de impacto ecológico, como a estimativa da redução de emissões de Gases de Efeito Estufa (GEE) ou CO_2 por carona.
- **Análise de Dados de Gênero e de Segurança:** os usuários desejam adicionar variáveis de análise relacionadas ao gênero para estudos de hábitos e segurança. Isso inclui analisar padrões como a probabilidade de aceitação entre gêneros e o uso de caronas entre mulheres, visando tornar a plataforma mais confortável e segura. Além disso, foi sugerida a inclusão de uma aba para visualizar problemas ou incidentes reportados nas caronas.
- **Avaliação de Rota:** uma sugestão técnica é a implementação de uma funcionalidade que visualize a rota percorrida e avalie sua eficiência através de algum tipo de algoritmo ou "avaliação" de rota, agregando valor à logística de transporte.

Todos os usuários que responderam ao formulário opinaram que como membro atual da equipe Caronaê, ou como ex-membro se colocando no lugar de um membro atual, utilizariam o painel e acham que seria valioso para entender o impacto do aplicativo e guiar as decisões da equipe. Por fim, foram abordados alguns assuntos nos comentários finais, como:

- **Aprofundamento da Análise Comportamental (Engajamento):** foi sugerido investigar o alto engajamento (maior porcentagem de ativos) dos usuários de Doutorado. É necessário realizar análises de perfil mais detalhadas para validar se este é um "fenômeno" comportamental ou uma coincidência nos dados.
- **Definição de Benchmarking para Ocupação Média:** foi recomendado adicionar um valor de referência para a ocupação média, como a média de ocupação de carros no Rio de Janeiro ou no Brasil.
- **Melhoria da Nomenclatura e Terminologia:** foi proposto mudar o título da aba "Lugares" por um termo mais técnico e representativo, como "Origem/Destino" ou "Zonas e Centros".
- **Interpretação e Conclusões Analíticas:** foi apontada a falta de uma seção ou aba final que apresente as principais conclusões extraídas dos dados.
- **Análise da Discrepância de Macaé:** um usuário comentou sobre ter sentido uma aparente discrepância nos dados de caronas válidas de/para Macaé, porém essa análise veio de uma interpretação errônea dos dados.

- **Transparência Financeira:** sugestão de incluir uma aba que aborde a transparência financeira do projeto, detalhando despesas, recursos e reforçando que nenhuma taxa foi cobrada dos usuários.

Além disso, alguns usuários parabenizaram a iniciativa e elogiaram o nome do projeto (Analisaê). Um respondente pontuou que “entender como que os dados do Caronaê influenciam a vida acadêmica é super importante para ajudar a universidade e a própria equipe do projeto a trabalhar para melhorar cada vez mais o dia a dia dos filhos da nossa querida Minerva”, reforçando a ideia por trás do trabalho.

6 CONCLUSÃO

Este trabalho teve como objetivo principal aprimorar a capacidade analítica do Projeto Caronaê, um sistema universitário de caronas compartilhadas, através da concepção e implementação de um novo ambiente de inteligência de negócio. Partindo da análise das limitações de uma abordagem inicial que utilizava diretamente o banco de dados transacional, a nova abordagem focou na aplicação de princípios de modelagem dimensional para a construção de um data warehouse dedicado, complementado por um processo robusto de ETL desenvolvido em Python e um painel com visualizações interativas no Tableau para execução de operações OLAP.

A transição metodológica do banco transacional para o dimensional não representou apenas uma escolha técnica, mas uma resposta estratégica às deficiências intrínsecas da utilização do ambiente transacional para fins de inteligência de negócio. A complexidade de um projeto como o Caronaê, com seus dados históricos e a necessidade de informações aprofundadas, exige uma infraestrutura de dados especializada para garantir a sustentabilidade e a confiabilidade das análises.

A implementação do novo ambiente analítico possibilitou análises mais eficientes e intuitivas dos dados de uso do aplicativo Caronaê, além de trazer mais robustez, integridade e qualidade para os dados com potencial analítico, que agora têm um ambiente próprio, separado da operação OLTP. A qualidade e relevância dos dados foi reforçada pela implementação de novas métricas, como a de usuários impactados.

O novo ambiente analítico potencializa o papel estratégico do Caronaê como parte do ecossistema maior de transporte de/para e dentro dos campi da UFRJ, pensado por projetos como o Plano de Gestão de Logística Sustentável da Universidade (PLS-UFRJ). O ambiente fornece aos gestores de mobilidade acesso a dados valiosos sobre as caronas que ocorrem de/para os campi pelo aplicativo, e sobre o perfil dos que oferecem e pegam essas caronas. É possível, assim, contextualizar de forma mais clara a função e o impacto das caronas e do Caronaê no ecossistema supracitado.

Por mais que o trabalho não traga análises mais complexas que utilizem conceitos de ciência de dados como aprendizado de máquina, ele fornece uma base sólida para que futuros membros da equipe Caronaê possam executar tais análises de forma eficiente.

Apesar dos avanços significativos, o presente estudo possui algumas limitações, como a falta de um servidor dedicado para o data warehouse, visto que o mesmo por enquanto

existe apenas localmente no computador do autor do trabalho. Além disso, destaca-se a falta de dados como o de gênero dos usuários e o do motivo pelo qual uma carona não aconteceu, que poderiam servir para criar análises que suprissem demandas apresentadas pelos respondentes do formulário de avaliação qualitativa.

Para um futuro trabalho, seria interessante implementar estratégias de atualização das SCDs (*Slowly Changing Dimensions*) do data warehouse, descritas por Barbieri como "dimensões que mudam com menos frequência ou mais vagarosamente" (BARBIERI, 2011, p. 187), para que se mantenha o histórico de registros anteriores que foram atualizados e seja possível incluí-los nas análises.

Além disso, trazer uma avaliação quantitativa para averiguar a diferença entre a velocidade das consultas direto do banco transacional e as ligadas ao data warehouse, adicionaria uma camada de respaldo à nova solução aqui introduzida.

Por fim, é interessante tocar no ponto da possibilidade de integração dos dados do estudo sobre emissões conduzido em 2022 ao ambiente aqui proposto, atendendo aos pedidos de alguns dos respondentes do formulário de avaliação qualitativa. Estes dados trariam uma importante perspectiva ecológica que complementaria as análises de uso do aplicativo para trazer uma visão mais holística dos impactos do projeto não só na comunidade acadêmica da UFRJ, mas no meio ambiente e no debate sobre meios de transporte sustentáveis no ambiente urbano.

REFERÊNCIAS

BARBIERI, C. **BI2-Business Intelligence - Modelagem & Qualidade.** Rio de Janeiro: Elsevier, 2011. 416 p.

CODD, E. F. **A Relational Model of Data for Large Shared Data Banks**. Communications of the ACM, 1970.

CODD, E. F. **Further Normalization of the Data Base Relational Model**, 1971.

GOMES, N. **CARONAÊ: UNIFICANDO E AMPLIANDO AS CARONAS NA UFRJ.** Rio de Janeiro, 2019. Apresentação do Caronaê no Festival do Conhecimento UFRJ 2019, Universidade Federal do Rio de Janeiro. Disponível em https://drive.google.com/file/d/1NYfCuaVyB9ev5TB1ZMz-spGf5DcL731r/view?usp=sharing. Acesso em: 23 abr. 2025.

INMON, W. H. **Building the Data Warehouse.** 3 ed. Wiley, 2002.

KIMBALL, R.; ROSS, M. **The Data Warehouse Toolkit.** 2. ed. Wiley, 2002.

KIMBALL, R.; ROSS, M.; THORNTHWAITE, W.; MUNDY, J.; BECKER, B. **The Data Warehouse Lifecycle Toolkit.** 2. ed. Wiley, 1998.

MEYER, M. de F. **Caronaê – Uma alternativa para o Gerenciamento da Mobilidade.** Rio de Janeiro, 2019. Trabalho de Conclusão de Curso de Engenharia Ambiental da Escola Politécnica, Universidade Federal do Rio de Janeiro. Disponível em: https://www.ltc.coppe.ufrj.br/producao/trabalho/tcc/. Acesso em: 23 abr. 2025.

POWER, D. J. A Brief History of Decision Support Systems. **DSSResources.com**, 2007. Disponível em: https://dssresources.com/history/dsshistory.html. Acesso em: 06 abr. 2025

REDDY, G. S.; SRINIVASU, R.; RAO M. P. C.; RIKKULA, S. R. Data Warehousing, Data Mining, OLAP and OLTP Technologies are Essential Elements to Support Decision-Making

Process in Industries. **(IJCSE) International Journal on Computer Science and Engineering** Vol. 02, No. 09, 2010.

STEVENS, H. Hans Peter Luhn and the Birth of the Hashing Algorithm. **IEEE Spectrum**, 2018. Disponível em: https://spectrum.ieee.org/hans-peter-luhn-and-the-birth-of-the-hashing-algorithm. Acesso em: 26 abr. 2025.

APÊNDICE A – COMANDOS SQL

Comando de criação da tabela `Dim_Date`:

```
CREATE TABLE IF NOT EXISTS dim_date (
   date_sk INT NOT NULL,
   full_date DATE NOT NULL,
   day_of_week INT NOT NULL,
   day_name VARCHAR(20) NOT NULL,
   day_of_month INT NOT NULL,
   month INT NOT NULL,
   month_name VARCHAR(20) NOT NULL,
   semester INT NOT NULL,
   period VARCHAR(20) NOT NULL,
   year INT NOT NULL,

   PRIMARY KEY (date_sk)
);
```

Comando de inserção de dados na tabela `Dim_Date`:

```
INSERT INTO dim_date (
   date_sk, full_date, day_of_week, day_name, day_of_month,
   month, month_name, semester, period, year
) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
ON CONFLICT (date_sk) DO NOTHING;
```

Comando de criação da tabela `Dim_Hour`:

```
CREATE TABLE IF NOT EXISTS dim_hour (
   hour_sk INT NOT NULL,
   hour_of_day INT NOT NULL,
   minute_of_hour INT NOT NULL,
   time_of_day_bucket VARCHAR(50) NOT NULL,

   PRIMARY KEY (hour_sk)
```

```
);
```

Comando de inserção de dados na tabela `Dim_Hour`:

```
INSERT INTO dim_hour (
   hour_sk, hour_of_day, minute_of_hour, time_of_day_bucket
) VALUES (%s, %s, %s, %s)
ON CONFLICT (hour_sk) DO NOTHING;
```

Comando de criação da tabela `Dim_User`:

```
CREATE TABLE IF NOT EXISTS dim_user (
   user_sk SERIAL PRIMARY KEY,
   user_id INT UNIQUE NOT NULL, -- Chave de negócio original
   academic_affiliation VARCHAR(50),
   course VARCHAR(100),
   has_car BOOLEAN NOT NULL,
   car_model VARCHAR(100),
   is_banned BOOLEAN NOT NULL,
   institution_id INT NOT NULL,
   institution_name VARCHAR(255),
   created_at TIMESTAMP,
   updated_at TIMESTAMP,
   deleted_at TIMESTAMP
);
```

Comando de seleção de dados das tabelas `users` e `institutions` do banco transacional para a tabela `Dim_User`:

```
SELECT
   u.id AS user_id,
   u.profile AS academic_affiliation,
   u.course,
   u.car_owner AS has_car,
   u.car_model,
   u.banned AS is_banned,
```

```
    i.id AS institution_id,
    i.name AS institution_name,
    u.created_at,
    u.updated_at,
    u.deleted_at
FROM users u
LEFT JOIN institutions i ON u.institution_id = i.id
ORDER BY u.id;
```

Comando de inserção de dados para a tabela `Dim_User`:

```
INSERT INTO dim_user (
    user_id, academic_affiliation, course,
    has_car, car_model,
    is_banned, institution_id, institution_name,
    created_at, updated_at, deleted_at
) VALUES (
    %(user_id)s, %(academic_affiliation)s, %(course)s,
    %(has_car)s, %(car_model)s,
    %(is_banned)s, %(institution_id)s, %(institution_name)s,
    %(created_at)s, %(updated_at)s, %(deleted_at)s
) ON CONFLICT (user_id) DO UPDATE SET
    academic_affiliation = EXCLUDED.academic_affiliation,
    course = EXCLUDED.course,
    has_car = EXCLUDED.has_car,
    car_model = EXCLUDED.car_model,
    is_banned = EXCLUDED.is_banned,
    institution_id = EXCLUDED.institution_id,
    institution_name = EXCLUDED.institution_name,
    created_at = EXCLUDED.created_at,
    updated_at = EXCLUDED.updated_at,
    deleted_at = EXCLUDED.deleted_at
```

Comando de criação da tabela `Dim_Place`:

```
CREATE TABLE IF NOT EXISTS dim_place (
    place_sk SERIAL PRIMARY KEY,
    place_name VARCHAR(255),
    place_type VARCHAR(50),

    hub_id INT UNIQUE,
    center VARCHAR(100),
    campus_id INT,
    campus_name VARCHAR(100),
    institution_id INT,
    institution_name VARCHAR(255),

    neighborhood_id INT UNIQUE,
    zone_id INT,
    zone_name VARCHAR(100)
);
```

Comando de seleção de dados das tabelas `hubs`, `campi` e `institutions` do banco transacional para a tabela `Dim_Place`:

```
SELECT
    h.id AS hub_id,
    h.name AS place_name,
    h.center,
    h.campus_id,
    c.name AS campus_name,
    c.institution_id,
    i.name AS institution_name
FROM hubs h
LEFT JOIN campi c ON h.campus_id = c.id
LEFT JOIN institutions i ON c.institution_id = i.id
ORDER BY h.id;
```

Comando de seleção de dados das tabelas `neighborhoods` e `zones` do banco transacional para a tabela `Dim_Place`:

```
SELECT
    n.id AS neighborhood_id,
    n.name AS place_name,
    n.zone_id,
    z.name AS zone_name
FROM neighborhoods n
LEFT JOIN zones z ON n.zone_id = z.id;
```

Comando de inserção de dados para a tabela `Dim_Place`:

```
INSERT INTO dim_place (
    place_name, place_type,
    hub_id, center, campus_id, campus_name, institution_id, institution_name,
    neighborhood_id, zone_id, zone_name
) VALUES (
    %(place_name)s, %(place_type)s,
    %(hub_id)s, %(center)s, %(campus_id)s, %(campus_name)s, %(institution_id)s,
    %(institution_name)s, %(neighborhood_id)s, %(zone_id)s, %(zone_name)s
)
ON CONFLICT (place_sk) DO UPDATE SET
    place_name = EXCLUDED.place_name,
    place_type = EXCLUDED.place_type,

    hub_id = EXCLUDED.hub_id,
    center = EXCLUDED.center,
    campus_id = EXCLUDED.campus_id,
    campus_name = EXCLUDED.campus_name,
    institution_id = EXCLUDED.institution_id,
    institution_name = EXCLUDED.institution_name,

    neighborhood_id = EXCLUDED.neighborhood_id,
    zone_id = EXCLUDED.zone_id,
```

```
    zone_name = EXCLUDED.zone_name
```

Comando de criação da tabela `Dim_Request_Status:`

```
CREATE TABLE IF NOT EXISTS dim_request_status (
    status_sk SERIAL PRIMARY KEY,
    status_name VARCHAR(50) UNIQUE NOT NULL,
    status_description VARCHAR(255)
);
```

Comando de seleção de dados da tabela `ride_user` do banco transacional para a tabela `Dim_Request_Status:`

```
SELECT DISTINCT status
FROM ride_user
WHERE status <> 'driver' AND status IS NOT NULL
ORDER BY status;
```

Comando de inserção de dados para a tabela `Dim_Request_Status:`

```
INSERT INTO dim_request_status (status_name, status_description)
VALUES (%s, %s)
ON CONFLICT (status_name) DO UPDATE SET
    status_description = EXCLUDED.status_description;
```

Comando de criação da tabela `Dim_Routine:`

```
CREATE TABLE IF NOT EXISTS dim_routine (
    routine_sk SERIAL PRIMARY KEY,
    routine_id INT UNIQUE,
    repeats_until TIMESTAMP,
    is_routine_monday BOOLEAN NOT NULL,
    is_routine_tuesday BOOLEAN NOT NULL,
    is_routine_wednesday BOOLEAN NOT NULL,
    is_routine_thursday BOOLEAN NOT NULL,
    is_routine_friday BOOLEAN NOT NULL,
    is_routine_saturday BOOLEAN NOT NULL,
```

```
    is_routine_sunday BOOLEAN NOT NULL,
    routine_days_description VARCHAR(255)
);
```

Comando de seleção de dados da tabela `rides` do banco transacional para a tabela `Dim_Routine`:

```
SELECT DISTINCT
    r.routine_id,
    r.repeats_until,
    r.week_days
FROM rides r
WHERE r.routine_id IS NOT NULL
    AND r.repeats_until IS NOT NULL
ORDER BY r.routine_id;
```

Comando de inserção de dados para a tabela `Dim_Routine`:

```
INSERT INTO dim_routine (
    routine_id, repeats_until,
    is_routine_monday, is_routine_tuesday, is_routine_wednesday,
    is_routine_thursday, is_routine_friday, is_routine_saturday, is_routine_sunday,
    routine_days_description
) VALUES (
    %(routine_id)s, %(repeats_until)s,
    %(is_routine_monday)s, %(is_routine_tuesday)s, %(is_routine_wednesday)s,
    %(is_routine_thursday)s, %(is_routine_friday)s, %(is_routine_saturday)s,
    %(is_routine_sunday)s, %(routine_days_description)s
) ON CONFLICT (routine_id) DO UPDATE SET
    repeats_until = EXCLUDED.repeats_until,
    is_routine_monday = EXCLUDED.is_routine_monday,
    is_routine_tuesday = EXCLUDED.is_routine_tuesday,
    is_routine_wednesday = EXCLUDED.is_routine_wednesday,
    is_routine_thursday = EXCLUDED.is_routine_thursday,
    is_routine_friday = EXCLUDED.is_routine_friday,
```

```
    is_routine_saturday = EXCLUDED.is_routine_saturday,
    is_routine_sunday = EXCLUDED.is_routine_sunday,
    routine_days_description = EXCLUDED.routine_days_description
```

Comando de criação da tabela `Dim_Ride_Flags`:

```
CREATE TABLE IF NOT EXISTS dim_ride_flags (
    ride_flags_sk SERIAL PRIMARY KEY,
    is_going_to_campus BOOLEAN NOT NULL,
    done BOOLEAN NOT NULL,
    deleted BOOLEAN NOT NULL,
    flags_description VARCHAR(255) UNIQUE -- Para facilitar a visualização e garantir unicidade da combinação textual
);
```

Comando de inserção de dados para a tabela `Dim_Ride_Flags`:

```
INSERT INTO dim_ride_flags (
    is_going_to_campus, done, deleted, flags_description
) VALUES (%s, %s, %s, %s);
```

Comando de criação da tabela `Fact_Ride`:

```
CREATE TABLE IF NOT EXISTS fact_ride (
    ride_sk SERIAL PRIMARY KEY, -- Chave primária para o fato
    ride_id INT UNIQUE NOT NULL, -- Dimensão degenerada

    -- SKs
    driver_user_sk INT NOT NULL,
    place_origin_sk INT NOT NULL,
    place_destination_sk INT NOT NULL,
    ride_flags_sk INT NOT NULL,
    routine_sk INT NOT NULL,

    -- SKs para o momento de CRIAÇÃO da carona
    creation_date_sk INT NOT NULL,
```

```
    creation_hour_sk INT NOT NULL,

    -- SKs para o momento de OCORRÊNCIA da carona
    occurrence_date_sk INT NOT NULL,
    occurrence_hour_sk INT NOT NULL,

    slots_count INT,
    messages_count INT DEFAULT 0,
    requests_count INT DEFAULT 0,
    accepted_requests_count INT DEFAULT 0,
    refused_requests_count INT DEFAULT 0,
    pending_requests_count INT DEFAULT 0,
    quit_requests_count INT DEFAULT 0,

    -- FKs
    FOREIGN KEY (driver_user_sk) REFERENCES dim_user(user_sk),
    FOREIGN KEY (place_origin_sk) REFERENCES dim_place(place_sk),
    FOREIGN KEY (place_destination_sk) REFERENCES dim_place(place_sk),
    FOREIGN KEY (ride_flags_sk) REFERENCES dim_ride_flags(ride_flags_sk),
    FOREIGN KEY (routine_sk) REFERENCES dim_routine(routine_sk),

    -- FKs para as dimensões de tempo
    FOREIGN KEY (creation_date_sk) REFERENCES dim_date(date_sk),
    FOREIGN KEY (creation_hour_sk) REFERENCES dim_hour(hour_sk),
    FOREIGN KEY (occurrence_date_sk) REFERENCES dim_date(date_sk),
    FOREIGN KEY (occurrence_hour_sk) REFERENCES dim_hour(hour_sk)
);
```

Comando de seleção de dados das tabelas `rides`, `ride_user` e `messages` do banco transacional para a tabela `Fact_Ride`:

```
SELECT
    r.id AS ride_id,
    r.routine_id,
```

```
    r.repeats_until,
    ru_driver.user_id AS driver_id,
    r.neighborhood AS neighborhood_name, -- Para pegarmos o sk
    r.hub AS hub_name, -- Para pegarmos o sk
    r.going AS is_going_to_campus,
    r.done,
    r.deleted_at,
    r.created_at, -- Chaves Temporais / Controle do ETL, marca d'água
    r.updated_at, -- Para controle do ETL, marca d'água
    r.date AS occurred_at, -- Chaves Temporais
    r.slots AS slots_count,
    msg.messages_count
FROM rides r
LEFT JOIN ride_user ru_driver ON r.id = ru_driver.ride_id AND ru_driver.status = 'driver'
LEFT JOIN (
    SELECT ride_id, COUNT(*) AS messages_count
    FROM messages
    GROUP BY ride_id
) AS msg ON r.id = msg.ride_id
WHERE (r.created_at >= '{last_etl_run_date}' OR r.updated_at >= '{last_etl_run_date}' OR
r.deleted_at >= '{last_etl_run_date}')
ORDER BY r.id;
```

Comando de seleção de dados da tabela `ride_user` do banco transacional para a tabela `Fact_Ride`:

```
SELECT
    ride_id,
    status
FROM ride_user
WHERE (created_at >= '{last_etl_run_date}' OR updated_at >= '{last_etl_run_date}');
```

Comando de inserção de dados para a tabela `Fact_Ride`:

```
INSERT INTO fact_ride (
```

```
    ride_id,
    driver_user_sk, place_origin_sk, place_destination_sk, ride_flags_sk, routine_sk,
    creation_date_sk, creation_hour_sk, occurrence_date_sk, occurrence_hour_sk,
    slots_count, messages_count, requests_count, accepted_requests_count,
    refused_requests_count, pending_requests_count, quit_requests_count
) VALUES (
    %(ride_id)s, %(driver_user_sk)s, %(place_origin_sk)s, %(place_destination_sk)s,
    %(ride_flags_sk)s, %(routine_sk)s, %(creation_date_sk)s, %(creation_hour_sk)s,
    %(occurrence_date_sk)s, %(occurrence_hour_sk)s, %(slots_count)s, %(messages_count)s,
    %(requests_count)s, %(accepted_requests_count)s, %(refused_requests_count)s,
    %(pending_requests_count)s, %(quit_requests_count)s
) ON CONFLICT (ride_id) DO UPDATE SET
    driver_user_sk = EXCLUDED.driver_user_sk,
    place_origin_sk = EXCLUDED.place_origin_sk,
    place_destination_sk = EXCLUDED.place_destination_sk,
    ride_flags_sk = EXCLUDED.ride_flags_sk,
    routine_sk = EXCLUDED.routine_sk,

    creation_date_sk = EXCLUDED.creation_date_sk,
    creation_hour_sk = EXCLUDED.creation_hour_sk,
    occurrence_date_sk = EXCLUDED.occurrence_date_sk,
    occurrence_hour_sk = EXCLUDED.occurrence_hour_sk,

    slots_count = EXCLUDED.slots_count,
    messages_count = EXCLUDED.messages_count,
    requests_count = EXCLUDED.requests_count,
    accepted_requests_count = EXCLUDED.accepted_requests_count,
    refused_requests_count = EXCLUDED.refused_requests_count,
    pending_requests_count = EXCLUDED.pending_requests_count,
    quit_requests_count = EXCLUDED.quit_requests_count
```

Comando de criação da tabela `Fact_Ride_Request`:

```
CREATE TABLE IF NOT EXISTS fact_ride_request (
```

```
    request_pk SERIAL PRIMARY KEY, -- Chave primária para o fato
    request_id INT UNIQUE NOT NULL, -- Dimensão degenerada

    ride_sk INT NOT NULL, -- ID da carona a que se refere (FK para fact_ride.ride_sk)
    user_sk INT NOT NULL, -- Usuário que fez a interação (motorista ou caronista)
    status_sk INT NOT NULL, -- Status final da interação

    -- SKs para o momento de CRIAÇÃO da interação
    creation_date_sk INT NOT NULL,
    creation_hour_sk INT NOT NULL,

    -- SKs para o momento de ATUALIZAÇÃO da interação
    update_date_sk INT NOT NULL,
    update_hour_sk INT NOT NULL,

    -- FKs
    FOREIGN KEY (ride_sk) REFERENCES fact_ride(ride_sk),
    FOREIGN KEY (user_sk) REFERENCES dim_user(user_sk),
    FOREIGN KEY (status_sk) REFERENCES dim_request_status(status_sk),

    -- FKs para as dimensões de tempo
    FOREIGN KEY (creation_date_sk) REFERENCES dim_date(date_sk),
    FOREIGN KEY (creation_hour_sk) REFERENCES dim_hour(hour_sk),
    FOREIGN KEY (update_date_sk) REFERENCES dim_date(date_sk),
    FOREIGN KEY (update_hour_sk) REFERENCES dim_hour(hour_sk)
);
```

Comando de seleção de dados da tabela `ride_user` do banco transacional para a tabela `Fact_Ride_Request`:

```
SELECT
    id AS request_id,
    ride_id,
    user_id,
```

```
    created_at,
    updated_at,
    status
FROM ride_user
WHERE status <> 'driver' -- já tirando os pedidos de criação de carona
AND (created_at >= '{last_etl_run_date}' OR updated_at >= '{last_etl_run_date}')
ORDER BY id;
```

Comando de inserção de dados para a tabela `Fact_Ride_Request`:

```
INSERT INTO fact_ride_request (
    request_id, ride_sk, user_sk, status_sk,
    creation_date_sk, creation_hour_sk, update_date_sk, update_hour_sk
) VALUES (
    %(request_id)s, %(ride_sk)s, %(user_sk)s, %(status_sk)s,
    %(creation_date_sk)s, %(creation_hour_sk)s, %(update_date_sk)s, %(update_hour_sk)s
) ON CONFLICT (request_id) DO UPDATE SET
    ride_sk = EXCLUDED.ride_sk,
    user_sk = EXCLUDED.user_sk,
    status_sk = EXCLUDED.status_sk,

    creation_date_sk = EXCLUDED.creation_date_sk,
    creation_hour_sk = EXCLUDED.creation_hour_sk,

    update_date_sk = EXCLUDED.update_date_sk,
    update_hour_sk = EXCLUDED.update_hour_sk
```

Comandos de truncamento das tabelas:

```
TRUNCATE TABLE fact_ride_request RESTART IDENTITY CASCADE;
TRUNCATE TABLE fact_ride RESTART IDENTITY CASCADE;
TRUNCATE TABLE dim_ride_flags RESTART IDENTITY CASCADE;
TRUNCATE TABLE dim_routine RESTART IDENTITY CASCADE;
TRUNCATE TABLE dim_request_status RESTART IDENTITY CASCADE;
```

```
TRUNCATE TABLE dim_place RESTART IDENTITY CASCADE;
TRUNCATE TABLE dim_user RESTART IDENTITY CASCADE;
TRUNCATE TABLE dim_hour RESTART IDENTITY CASCADE;
TRUNCATE TABLE dim_date RESTART IDENTITY CASCADE;
```

APÊNDICE B – FORMULÁRIO DE AVALIAÇÃO QUALITATIVA

Pesquisa Qualitativa do Painel Gerencial Analisaê - TCC Daniel Levacov

Este formulário tem como objetivo angariar as opiniões, sugestões e experiências de membros atuais e antigos da equipe Caronaê sobre o **Painel Gerencial Analisaê**: um painel gerencial criado por mim (Daniel Levacov) para meu Trabalho de Conclusão de Curso de Ciência da Computação na UFRJ.

Este painel faz parte do **Ambiente Analítico Analisaê**, criado por mim em meu TCC, que também conta com um Data Warehouse (banco de dados otimizado para análises) e um processo de ETL (extração, transformação e carga de dados). O objetivo do painel é entregar análises que sirvam como um insumo para a equipe Caronaê, a fim de informar suas decisões para o futuro do projeto.

É importante ressaltar que todas as informações fornecidas são anônimas e serão utilizadas exclusivamente para o TCC e para fins de aprimoramento do painel. Garantimos a total confidencialidade dos dados.

O preenchimento deste formulário levará cerca de **5 minutos.**

Sua contribuição é fundamental para o sucesso deste trabalho. Agradeço desde já pela colaboração!

* Indica uma pergunta obrigatória

Você autoriza suas respostas a serem coletadas, armazenadas e analisadas? *

- ◯ Sim
- ◯ Não

Como você avaliaria sua experiência geral com o Painel Gerencial Analisaê? *

- ◯ Ótima
- ◯ Boa
- ◯ Regular
- ◯ Ruim
- ◯ Péssima

Você achou o painel intuitivo e fácil de usar? *

- ◯ Sim, foi fácil usar
- ◯ Sim, porém tive certas dificuldades
- ◯ Não, foi difícil usar

Achou o painel visualmente agradável? *

- ◯ Sim
- ◯ Não

Achou as informações mostradas no painel úteis e/ou interessantes? *

- ◯ Sim
- ◯ Não

O que mais chamou sua atenção positivamente no painel? *

Sua resposta

Houve algo que chamou sua atenção negativamente no painel? *

Sua resposta

Você se deparou com algum erro ou problema técnico durante sua experiência usando o painel? *

Sua resposta

Há alguma funcionalidade que atualmente não está presente no painel que você gostaria de ver?

Sua resposta

Como membro atual da equipe Caronaê, ou como ex-membro, se colocando no lugar de um membro atual, você utilizaria este painel? *

- ◯ Não utilizaria
- ◯ Sim, apenas de curiosidade
- ◯ Sim, acho que seria valioso para entender o impacto do aplicativo e guiar as decisões da equipe

Há algo mais que gostaria de compartilhar sobre o Painel Gerencial Analisaê? Deixe qualquer comentário ou sugestão aqui.

Sua resposta

Enviar

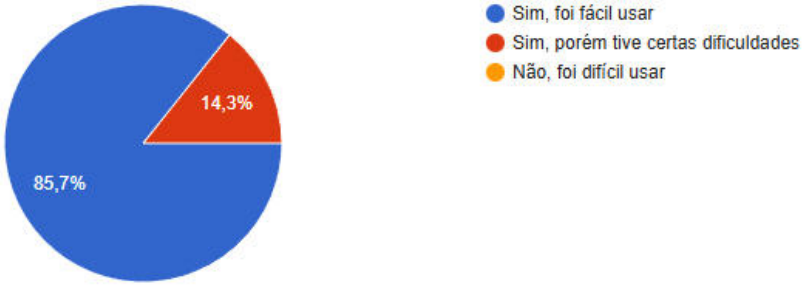
Limpar formulário

APÊNDICE C – RESULTADOS DO FORMULÁRIO DE AVALIAÇÃO QUALITATIVA

Você achou o painel intuitivo e fácil de usar?

Copiar gráfico

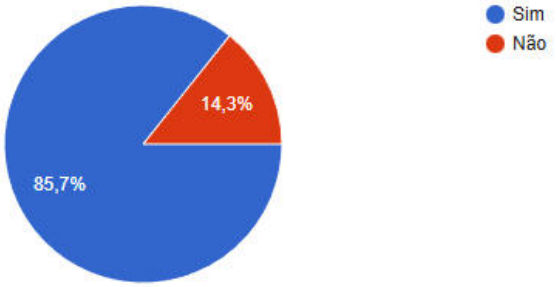
7 respostas



Achou o painel visualmente agradável?

Copiar gráfico

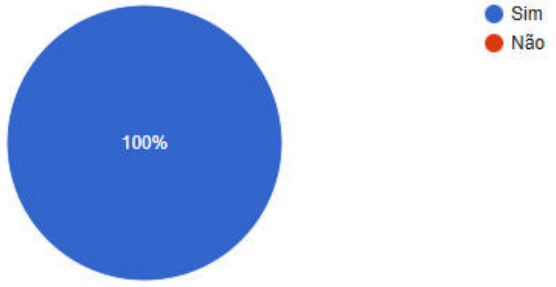
7 respostas



Achou as informações mostradas no painel úteis e/ou interessantes?

Copiar gráfico

7 respostas



Quadro 11: Respostas sobre pontos positivos do painel Analisaê - parte da avaliação qualitativa

O que mais chamou sua atenção positivamente no painel?
“A quantidade de dados e a relação entre eles amei saber quantos usuários ativos que tinhamos por nivel acadêmico e quantas caronas ja foram realizadas por lugar na ufrj e por zona, da pra entender bem mais o publico que usou o caronae, sabe quantos usuarios tem carro cadastrado”
“facilidade de manipular e clareza das informações”
“Os agrupamentos ficaram muito legais (abas)! A escolha de apresentar as cores em gradiente ficou bem legal também!”
“a interatividade com as analises propostas, podendo selecionar dados específicos dentro do que é disponibilizado no painel.”
“Toda a facilidade para filtrar por informações na aba de "Números gerais".”
“Achei ótimo poder selecionar certas informações para poder visualizar e esconder contextos e situações específicas para entender melhor os dados de caronas (por exemplo: ver caronas de/para minha zona ou centro de origem ou destino) – achei incrível essa possibilidade e nível de detalhamento!”
“O número de análises e sua distribuição por categorias.”

Quadro 12: Respostas sobre pontos negativos do painel Analisaê - parte da avaliação qualitativa

Houve algo que chamou sua atenção negativamente no painel?
“O filtro de data nao é pratico de usar porque aparecem muitos anos e fica dificil ser precioso em relação ao periodo em que aconteceu Eu nao sei se tem como sabermos o motivo do porque as caronas não foram válidas, se tem algum percentual que é por motivo tecnico do aplicativo etc. Nao sei se esses usuarios com carro cadastrado qnts deles fizeram corridas, entao faltou mais dados sobre os motoristas até para fazer comparativo entre quanto de procura temos dos usuários e quanto que ofertamos em caronas”
“o tempo de loading ao selecionar uma guia ou uma opção”
“Pelo laptop, tela 15", o tableau ficou recortado. A fonte usada poderia ser menor. As zonas haviam sido atribuidas cores específicas que poderiam estar retratadas no painel. Não há nenhum lastro da identidade visual elaborada, ficando um pouco frio e impessoal demais - faltou um 'tapa de designer'”
“A princípio, não. Interagindo com os dados no painel, na aba "lugares", cliquei em uma

Houve algo que chamou sua atenção negativamente no painel?
das zonas e ela passou a aparecer sozinha no gráfico, e não consegui retornar ao original. (não sei se isso é uma questão ou eu que não soube usar direito).”
“Não houve / Não percebi.”
“Não.”
“Não”

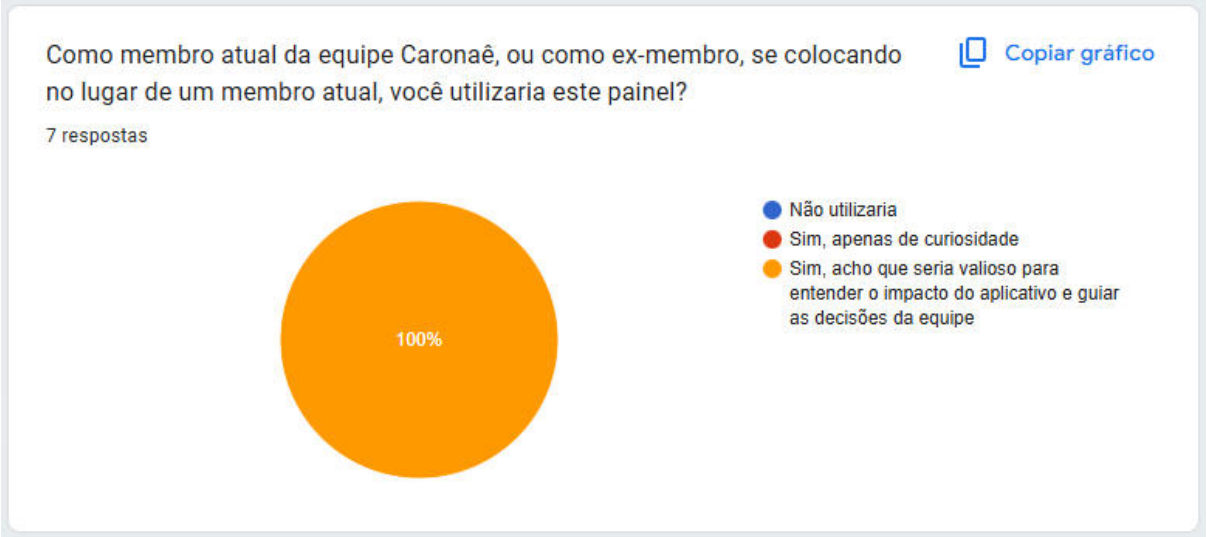
Quadro 13: Respostas sobre erros e problemas técnicos do painel Analisaê - parte da avaliação qualitativa

Você se deparou com algum erro ou problema técnico durante sua experiência usando o painel?
“Não”
“Apenas lentidão”
“Dificuldade em manusear dados em telas menores - pouca responsividade do painel (pode ser que seja um problema do Tableau mesmo e não do painel estruturado, não sei)”
“Aba Lugares - No gráfico % de Caronas Válidas de/para cada Zona, aparece destacada a Zona Sul e não o gráfico com todas as barras das demais zonas”
“Essa que relatei acima.”
“Não.”
“Nenhum! Funcionou muito bem, mesmo no Safari (que costuma ser mais chatinho com alguns sistemas e sites).”

Quadro 14: Respostas sobre funcionalidades que os usuários gostariam de ver no painel Analisaê - parte da avaliação qualitativa

Há alguma funcionalidade que atualmente não está presente no painel que você gostaria de ver?
“Filtro de data como um select (selecionar mês, ano, periodo) ou campo de texto para inserir o periodo que quero visualizar o dado”
“Cálculo estimado de redução de emissões GEE/CO2 por ocasião das caronas”
“Talvez algo em relação a algum problema relatado na carona (lembro que havia no app um espaço para isso)? tipo uma aba em que houvesse esses dados, para que pudessem ser acessados facilmente.”
“Poderia ter algo que visualizasse a rota que foi feita do passageiro e checar com algum

Há alguma funcionalidade que atualmente não está presente no painel que você gostaria de ver?
tipo de algoritmo se aquela é uma rota ótima, ou se poderia dar uma "avaliação" relativa a essa rota que foi feita.”
“Acho que seria interessante ter informações de sexo das pessoas para entender hábitos (como por exemplo: quais informações conseguiríamos tirar das caronas que mulheres oferecem e/ou pegam? normalmente, mulheres pegam caronas para mulheres? é mais provável de uma mulher oferecendo carona aceitar uma mulher ou um homem? caso as métricas sejam como esperamos que sejam, como podemos tornar a experiência de todos mais confortável e segura para que essas métricas melhorem?), além disso seria interessante ver métricas associadas à emissão de CO2 de acordo com o modelo do veículo e todas as informações que vem a partir disso: apoiando até mesmo a criação de relatórios semestrais de como o Caronaê objetiva não só facilitar a vida das pessoas ao ir e voltar da UFRJ, mas também como pode melhorar a questão ambiental que vivemos hoje em dia. São iniciativas como essa que influenciam positivamente toda a luta que estamos vendo na COP30, por exemplo.”



Quadro 15: Comentários ou sugestões finais sobre o painel Analisaê - parte da avaliação qualitativa

Há algo mais que gostaria de compartilhar sobre o Painel Gerencial Analisaê? Deixe qualquer comentário ou sugestão aqui.
“Aparentemente os usuários de Doutorado foram os que tiveram maior engajamento (maior% de ativos). Seria interessante entender este "fenômeno", validar se foi apenas uma coincidência ou se tem algo mais específico/comportamental/perfil que explique Na aba de Ocupação média, seria legal ter a referência da média Brasil e mundo (o que tiver de dado disponvel) para balizar um pouco se 2,54 - 2,57 - 2,56 são valores bons ou ruins. Se não me falha a memória, no município do Rio a média era inferior a 2 quando

Há algo mais que gostaria de compartilhar sobre o Painel Gerencial Analisaê? Deixe qualquer comentário ou sugestão aqui.
escrevemos a justificativa teórica para o concurso Soluções Sustentáveis do Fundo Verde A aba Lugares poderia ter seu título substituído por Origem/Destino, zonas e bairros ou algo um pouco mais técnico. Lugares é muito vago e pouco representativo. Chama atenção a % de caronas válidas de/para Macaé (19,2%), ainda que seja a menor % de caronas por centro (1,38%). Ficou um pouco confuso na realidade como isso ocorre. Senti falta de uma aba final com análises em si, e não apenas a apresentação dos gráficos/dados, sabe? O que se tira disso tudo? Acho que pra fins de transparência também seria legal uma aba com as despesas do projeto (quanto de recurso ja entrou, pra onde ja foi, etc) e esclarecimento de que nenhuma taxa ou % foi cobrada de qualquer usuário para uso do sistema e oferta/usufruto de caronas”
“Parabéns pelo trabalho, Daniel!”
“Gostaria de parabenizar pelo ótimo trabalho e iniciativa de trazer o projeto para mais perto das pessoas não só de dentro do Caronaê mas também possivelmente de fora – entender como que os dados do Caronaê influenciam a vida acadêmica é super importante para ajudar a universidade e a própria equipe do projeto a trabalhar para melhorar cada vez mais o dia a dia dos filhos da nossa querida Minerva. Fiquei muito feliz de ver o Caronaê seguindo com esse projeto de Painel com o incrível nome Analisaê (muito criativo e que pegada gostosa de combinar com o famoso naming "Caronaê"). Parabéns novamente, Daniel! Sugestão: sempre que faço um formulário ou questionário web, costumo dar a opção aos respondentes de enviar uma cópia de suas respostas para o email deles, acho uma ótima prática e acredito ser uma ótima sugestão de melhoria desse formulário. Assim as pessoas ficam mais seguras de que as respostas que escreveu foram recebidas exatamente da forma com que escreveram criando essa espécie de documentação e rastro digital das respostas para o lado respondente também e até mesmo é uma forma de possibilitar os respondentes revisitarem as suas respostas, caso queiram (eu faço isso o tempo todo!). Fora isso, grande abraço!”