UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL
DA LUSOFONIA AFRO-BRASILEIRA
INSTITUTO DE ENGENHARIAS E DESENVOLVIMENTO SUSTENTÁVEL
BACHARELADO EM ENGENHARIA DE ENERGIAS

FRANCISCO EDESON ALVES BIZERRIL

APLICATIVO INTERCAMPI: DESENVOLVIMENTO, OPERACIONALIZAÇÃO E RELEVÂNCIA PARA A COMUNIDADE ACADÊMICA

REDENÇÃO - CE
2024

FRANCISCO EDESON ALVES BIZERRIL

APLICATIVO INTERCAMPI: DESENVOLVIMENTO, OPERACIONALIZAÇÃO E RELEVÂNCIA PARA A COMUNIDADE ACADÊMICA

Trabalho de Conclusão de Curso da Graduação em Engenharia de Energias do Instituto de Engenharias e Desenvolvimento Sustentável da Universidade da Integração Internacional da Lusofonia Afro-Brasileira como requisito para a obtenção do título de Engenheiro de Energia.
Orientador: Prof. Dr. George Leite Mamede
Coorientador: Prof. Dr. Tales Paiva Nogueira

REDENÇÃO - CE
2024

Universidade da Integração Internacional da Lusofonia Afro-Brasileira
Sistema de Bibliotecas da UNILAB
Catalogação de Publicação na Fonte.

Bizerril, Francisco Edeson Alves.

B625a

Aplicativo intercampi: desenvolvimento, operacionalização e relevância para a comunidade acadêmica / Francisco Edeson Alves Bizerril. - Redenção, 2024.
63f: il.

Monografia - Curso de Engenharia De Energias, Instituto De Engenharias E Desenvolvimento Sustentável, Universidade da Integração Internacional da Lusofonia Afro-Brasileira, Redenção, 2024.

Orientador: Prof. Dr. George Leite Mamede.
Coorientador: Prof. Dr. Tales Paiva Nogueira.

1. Aplicativos - Aplicativo Intercampi. 2. Análise de sentimentos. 3. Transporte universitário. 4. Inteligência artificial. I. Título

CE/UF/BSCA CDD 001.642

FRANCISCO EDESON ALVES BIZERRIL

APLICATIVO INTERCAMPI: DESENVOLVIMENTO, OPERACIONALIZAÇÃO E RELEVÂNCIA PARA A COMUNIDADE ACADÊMICA

Monografia apresentada como requisito para a obtenção do título de Bacharel em Engenharia de Energias, na Universidade da Integração Internacional da Lusofonia Afro-brasileira, UNILAB - Campos dos Palmares

Aprovado em: 13 de novembro de 2024.

BANCA EXAMINADORA

Prof. Dr. George Leite Mamede (Orientador)
Universidade da Integração Internacional da Lusofonia Afro-brasileira - UNILAB

Prof. Dr. Tales Paiva Nogueira (Coorientador)
Universidade da Integração Internacional da Lusofonia Afro-brasileira - UNILAB

Prof. Dr. Vitor Alencar de Mesquita
Universidade da Integração Internacional da Lusofonia Afro-brasileira - UNILAB

Primeiramente, gostaria de agradecer a Deus. Agradeço também aos meus pais, Aurisandra Alves e Luciano de Melo, pela educação, pelos valores que me transmitiram e pelo amor incondicional que sempre me ofereceram. À minha querida esposa, Leidiane Marques, sou grato pelo amor, pela paciência e pelo apoio constante em todos os momentos. Por fim, dedico um agradecimento especial ao meu filho, William Bizerril, meu príncipe e a razão da minha vida.

AGRADECIMENTOS

Em primeiro lugar, gostaria de expressar minha profunda gratidão à minha esposa, Leidiane Marques, cuja presença e apoio foram fundamentais para que eu pudesse alcançar este importante marco em minha vida.

Agradeço também a Maristela Lima, que me proporcionou suporte financeiro durante os anos de faculdade, especialmente nos momentos em que mais precisei.

Meus sinceros agradecimentos ao CNPQ pelo financiamento que possibilitou anos de pesquisa e à UNILAB pelo auxílio e apoio.

Ao Prof. Dr George Leite Mamede, sou grato pelas oportunidades, valiosas orientações ao longo de muitos anos e pelo suporte na realização deste trabalho.

Por fim, um agradecimento especial a todos que contribuíram, de forma direta ou indireta, e que sempre acreditaram em mim.

"Confie no Senhor de todo o seu coração e não se apoie em seu próprio entendimento; reconheça-o em todos os seus caminhos, e ele endireitará suas veredas." Provérbios 3:5-6

Feito com LaTeX

RESUMO

O presente trabalho tem como objetivo investigar a eficácia do aplicativo Intercampi, desenvolvido para a Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB), visando facilitar o acesso aos horários e rotas dos ônibus universitários. A pesquisa utiliza uma abordagem qualitativa, combinando a análise de dados coletados através do uso do aplicativo e a aplicação de um script de inteligência artificial para realizar a análise de sentimentos dos feedbacks dos usuários. As discussões abordam a importância do aplicativo na melhoria da experiência acadêmica, destacando como ele elimina a necessidade de consultar tabelas impressas e permite acesso em tempo real às informações de transporte. Os resultados indicam que o uso do Intercampi não apenas otimiza o deslocamento dos estudantes, mas também contribui para uma gestão mais eficiente do sistema de transporte universitário, com um impacto positivo na rotina acadêmica. A análise de sentimentos revela percepções majoritariamente favoráveis entre os usuários, evidenciando a relevância do aplicativo na comunidade acadêmica da UNILAB. Este estudo conclui que a implementação de soluções tecnológicas no ambiente universitário é essencial para promover maior integração e conectividade entre os usuários.

Palavras-chave: aplicativo Intercampi; análise de sentimentos; transporte universitário; inteligência artificial.

ABSTRACT

This study aims to investigate the effectiveness of the Intercampi app, developed for the University for International Integration of the Afro-Brazilian Lusophony (UNILAB), to facilitate access to university bus schedules and routes. The research uses a qualitative approach, combining the analysis of data collected through the use of the app and the application of an artificial intelligence script to perform sentiment analysis of user feedback. The discussions address the importance of the app in improving the academic experience, highlighting how it eliminates the need to consult printed tables and allows real-time access to transportation information. The results indicate that the use of Intercampi not only optimizes student commute, but also contributes to a more efficient management of the university transportation system, with a positive impact on academic routine. Sentiment analysis reveals mostly favorable perceptions among users, evidencing the relevance of the app in the UNILAB academic community. This study concludes that the implementation of technological solutions in the university environment is essential to promote greater integration and connectivity among users.

Keywords: Intercampi app; university transportation; sentiment analysis; academic community; artificial intelligence

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AAB	*Android App Bundle*
API	*Application Programming Interface*
APK	*Android Package*
BERT	*Bidirectional Encoder Representations from Transformers*
CRUD	*Create, Read, Update* e *Delete*
FCM	*Firebase Cloud Messaging*
HFT	*Hugging Face Transformers*
IA	Inteligência Artificia
IBGE	Instituto Brasileiro de Geografi e Estatistica
IDE	*Integrated Development Environment*
IoT	Internet das Coisas
MVC	*Model View Controller*
PLN	Processamento de Linguagem Natural
REST	*Representational State Transfer*
TICs	Tecnologias da Informação e Comunicação
UI	*User Interface*
UNILAB	Universidade da Integração Internacional da Lusofonia Afro-Brasileira
UTC	Tempo Universal Coordenado

SUMÁRIO

1 INTRODUÇÃO

Em algumas Universidades onde não é viável concentrar todas as atividades em um único local, o deslocamento se torna um dos grandes desafios para os estudantes, que já enfrentam as dificuldades inerentes à graduação, como pesquisas, atividades de extensão, provas com conteúdo extensos e listas de exercícios. Esse processo de deslocamento impacta significativamente a vida acadêmica dos alunos, podendo causar atrasos em provas importantes e altos custos financeiros, como também comprometer a sua permanência no ensino superior (Liveira Neto; Cavicchioli, 2024).

Nesse contexto, os ônibus universitários desempenham um papel crucial para solucionar o problema do deslocamento, facilitando o acesso gratuito a locais e serviços, como bibliotecas, laboratórios e restaurante universitário. Além disso, essa opção de transporte proporciona aos estudantes uma significativa redução de custos, permitindo que redirecionem seus recursos para áreas mais prioritárias, como aluguel, aquisição de material didático e alimentação (Silva, L. T. d. A., 2022). E é importante ressaltar, que professores, gestores, administradores e técnicos em geral, também utilizam o transporte universitário, desfrutando dessa facilidade em seu cotidiano.

Entretanto, conforme apontado por Zengo (2023), a divisão de transportes de algumas Universidades, carece de um sistema de gestão integrado e não consegue fornecer o suporte tecnológico necessário à comunidade. Assim, com base nos conceitos de *smart campus*, é fundamental que diversas iniciativas sejam implementadas para desenvolver, no futuro, um sistema de transporte inteligente. Isso pode ser alcançado por meio da adoção de tecnologias avançadas, visando construir um ambiente acadêmico mais eficiente, seguro e integrado (Schöning, 2014).

Assim, em meados de 2017, foi desenvolvido um aplicativo chamado Intercampi para a UNILAB (Universidade da Integração Internacional da Lusofonia Afro-Brasileira), disponibilizado gratuitamente na plataforma *Android*. Este aplicativo, construído em *Java*, foi projetado para facilitar a consulta rápida dos horários de transporte universitário, permitindo acesso em tempo real aos horários e rotas dos ônibus com ou sem conexão à internet (Oliveira R., 2024).

Antes da criação do Intercampi, era preciso consultar tabelas impressas fixadas em locais estratégicos como corredores, biblioteca e restaurante universitário, ou acessar arquivos em PDF para obter informações sobre os horários dos ônibus. Essa abordagem não apenas poderia gerar confusão, mas também causar frustração e desperdício de tempo, uma vez que a consulta e a visualização dos dados em tabelas não eram facilitadas, obrigando o usuário a vasculhar diversas listas em busca dos horários corretos.

Dessa forma, a introdução deste aplicativo representa uma inovação significativa nesse contexto. Através do mesmo, estudantes e profissionais podem acessar informações atualizadas sobre horários e rotas de maneira rápida e prática, eliminando a necessidade

de buscar dados em formatos anteriores, que eram menos acessíveis. Essa ferramenta não apenas melhora a experiência do usuário, mas também potencializa a eficiência do sistema de transporte universitário, permitindo o monitoramento em tempo real dos horários de forma dinâmica e o recebimento de notificações sobre eventuais alterações, tornando o deslocamento mais previsível e menos estressante.

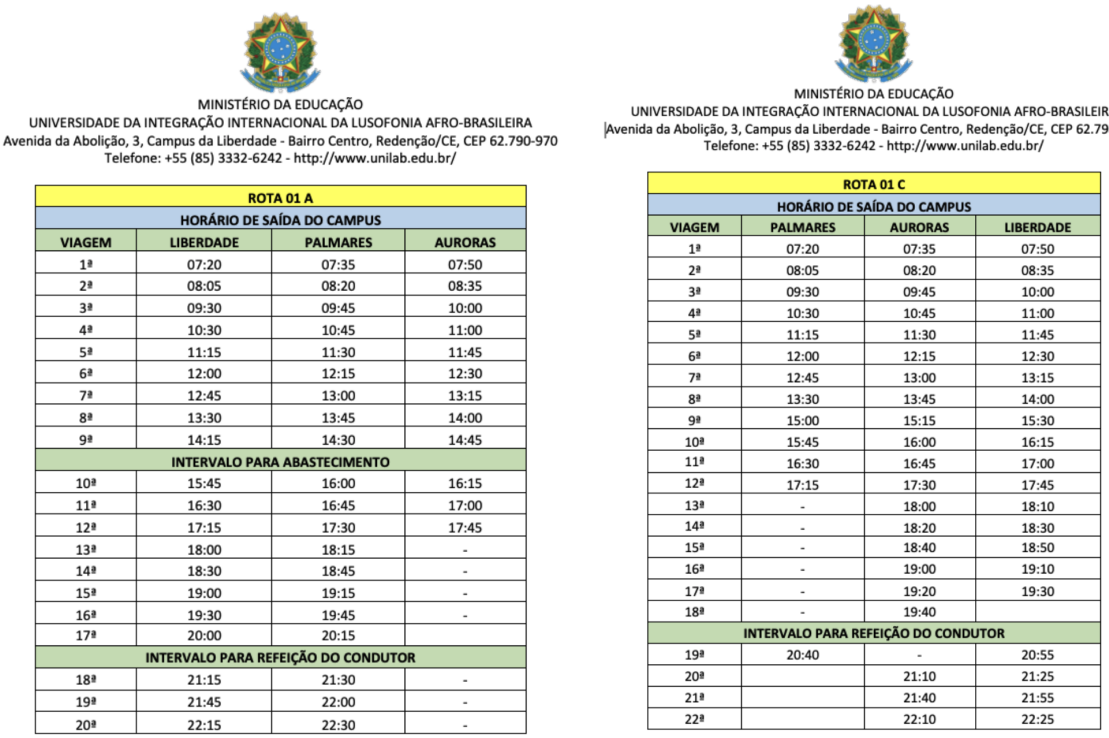
Outro fator importante, principalmente em um ambiente multicultural e diversificado como a UNILAB, é a análise de sentimentos aplicada às opiniões ou *feedbacks* fornecidos pelos usuários do aplicativo, o que se revela crucial para entender as percepções da comunidade acadêmica em relação ao aplicativo e sua integração com o sistema de transporte. Utilizando um algoritimo com inteligência artificial desenvolvido para este fim, essa análise permite uma avaliação imparcial das opiniões expressas, identificando rapidamente se os sentimentos são positivos, negativos ou neutros.

Por fim, investigar os impactos do aplicativo Intercampi na rotina acadêmica é essencial para compreender como essa tecnologia pode contribuir para uma experiência educacional mais fluida e integrada. Assim, este trabalho busca analisar essa questão em profundidade, destacando não apenas a construção e as funcionalidades do aplicativo, mas também sua relevância para toda a comunidade acadêmica da UNILAB. Também são apresentadas soluções para aprimorar o sistema o sistema de transporte universitário, beneficiando todos os usuários com maior integração e conectividade.

1.1 JUSTIFICATIVA

Antes da criação do aplicativo Intercampi, os estudantes precisavam fotografar diversas tabelas de horários e procurar entre as imagens para encontrar a informação desejada. Outra opção era baixar os arquivos PDF disponíveis no site da Universidade https://unilab.edu.br/onibus-intercampi, salvá-los e tentar localizá-los posteriormente no dispositivo em meio a outros arquivos. Mesmo assim, era necessário identificar a tabela correta e os horários específicos entre dezenas de itens, o que não era uma tarefa fácil no dia a dia. Devido a esses desafios, muitos preferiam não consultar as listas de horários e aguardar os ônibus às cegas.

Figura 1 – Exemplo de tabela de horários disponibilizadas pela universidade



MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL DA LUSOFONIA AFRO-BRASILEIRA
Avenida da Abolição, 3, Campus da Liberdade - Bairro Centro, Redenção/CE, CEP 62.790-970
Telefone: +55 (85) 3332-6242 - http://www.unilab.edu.br/

ROTA 01 A			
HORÁRIO DE SAÍDA DO CAMPUS			
VIAGEM	LIBERDADE	PALMARES	AURORAS
1ª	07:20	07:35	07:50
2ª	08:05	08:20	08:35
3ª	09:30	09:45	10:00
4ª	10:30	10:45	11:00
5ª	11:15	11:30	11:45
6ª	12:00	12:15	12:30
7ª	12:45	13:00	13:15
8ª	13:30	13:45	14:00
9ª	14:15	14:30	14:45
INTERVALO PARA ABASTECIMENTO			
10ª	15:45	16:00	16:15
11ª	16:30	16:45	17:00
12ª	17:15	17:30	17:45
13ª	18:00	18:15	-
14ª	18:30	18:45	-
15ª	19:00	19:15	-
16ª	19:30	19:45	-
17ª	20:00	20:15	
INTERVALO PARA REFEIÇÃO DO CONDUTOR			
18ª	21:15	21:30	-
19ª	21:45	22:00	-
20ª	22:15	22:30	-

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL DA LUSOFONIA AFRO-BRASILEIRA
Avenida da Abolição, 3, Campus da Liberdade - Bairro Centro, Redenção/CE, CEP 62.790-970
Telefone: +55 (85) 3332-6242 - http://www.unilab.edu.br/

ROTA 01 C			
HORÁRIO DE SAÍDA DO CAMPUS			
VIAGEM	PALMARES	AURORAS	LIBERDADE
1ª	07:20	07:35	07:50
2ª	08:05	08:20	08:35
3ª	09:30	09:45	10:00
4ª	10:30	10:45	11:00
5ª	11:15	11:30	11:45
6ª	12:00	12:15	12:30
7ª	12:45	13:00	13:15
8ª	13:30	13:45	14:00
9ª	15:00	15:15	15:30
10ª	15:45	16:00	16:15
11ª	16:30	16:45	17:00
12ª	17:15	17:30	17:45
13ª	-	18:00	18:10
14ª	-	18:20	18:30
15ª	-	18:40	18:50
16ª	-	19:00	19:10
17ª	-	19:20	19:30
18ª	-	19:40	
INTERVALO PARA REFEIÇÃO DO CONDUTOR			
19ª	20:40	-	20:55
20ª		21:10	21:25
21ª		21:40	21:55
22ª		22:10	22:25

Fonte: https://unilab.edu.br/onibus-intercampi/

Com o aplicativo Intercampi, os estudantes e profissionais da universidade têm em mãos acesso rápido e fácil a todos os horários disponíveis, sem necessidade de buscar ou interpretação de tabela de horários. Estando *online* (com conexão de *internet*), todos os dados serão atualizados remotamente e sem a necessidade de intervenção do usuário, e quando *offline* o aplicativo ainda estará disponível para uso normal.

O ícone do aplicativo na Figura 2 busca representar ou se aproximar da identidade visual da instituição. Todas as cores foram obtidas do logo da UNILAB e o fundo representa o mapa de localização do campus administrativo Liberdade. Assim, o aplicativo visa gerar uma conexão imediata com todos os estudantes e profissionais habituados com a universidade.

Figura 2 – Ícone do aplicativo Intercampi

Fonte: (Bizerril, F. E. A., 2024)

A primeira versão construída em *Java* utilizava o guia de estilo nativo do *Android* chamado de *Material Design* 1.0 (Google, 2024b). E por 6 anos, essa solução ajudou mais de 3 mil pessoas diariamente. Contudo, a sua manutenção era complexa e trabalhosa, em comparação a soluções mais atuais como o *Flutter*, pois para cada alteração e teste de execução era necessário compilar o aplicativo e fazer uma instalação nova. Com o *Java*, era possível compilar o aplicativo apenas para *Android*, e a codificação era mais verbosa e complexa. Assim, para agilizar o desenvolvimento, foi necessário reconstruí-lo do zero com tecnologia mais moderna e multiplataforma, como também utilizar o novo *Material Design* 3.0 (Google, 2024c). A tecnologia escolhida para essa tarefa foi o *Flutter* (Google, 2024a).

Figura 3 – Layout antigo do aplicativo Intercampi de 2017

Fonte: (Bizerril, F. E. A., 2024)

Nesse contexto, além de reconstruir o aplicativo e renovar a sua interface, este trabalho irá apresentar a arquitetura e serviços utilizados, explicar como ele é operado remotamente, e fazer análise de sentimentos dos *feedbacks* dos usuários entre 2017 a 2024 para mensurar o impacto da solução na comunidade acadêmica.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Apresentar uma solução tecnológica via aplicativo multiplataforma para dispositivos móveis, com o registro de rotas e horários do sistema de transporte Intercampi, fornecido pela UNILAB com monitoramento continuado, e analisar seus impactos na comunidade acadêmica.

1.2.2 Objetivos Específicos

- Apresentar o aplicativo Intercampi e suas funcionalidades;
- Descrever a arquitetura e tecnologias empregadas na construção do aplicativo;
- Falar sobre a publicação e operacionalização do aplicativo;
- Analisar a relevância do aplicativo para a comunidade acadêmica da UNILAB com Inteligência Artificial.

2 REFERENCIAL TEÓRICO

2.1 SMART CAMPUS E SUAS TECNOLOGIAS

Os conceitos de *smart campus* e cidades inteligentes são interligados, ambos utilizando TICs (Tecnologias da Informação e Comunicação) para melhorar a qualidade de vida e a eficiência dos serviços. As cidades inteligentes implementam soluções como sistemas de transporte inteligentes, gestão de resíduos e plataformas digitais de participação cidadã, promovendo um ambiente urbano mais responsivo às necessidades da população (Silva Flor; Teixeira, 2018).

Segundo Schöning (2014), os *smart campi*, são ambientes que utilizam tecnologias avançadas para criar um espaço de aprendizado mais interativo e eficiente para a comunidade acadêmica. Isto visa incorporar soluções que facilitem a colaboração entre alunos, professores e administração, e contribuir para uma educação de qualidade e inovação. Para uma implementação com maior assertividade e adesão dessas tecnologias, deve-se considerar a diversidade dos usuários, garantindo que as soluções atendam às necessidades específicas da comunidade (Schöning, 2014).

A transformação dos *smart campi* é fortemente impulsionada pelo uso de aplicações que permitem a integração e a gestão eficiente dos dados coletados. Esses aplicativos não apenas facilitam a comunicação e a colaboração, mas também permitem uma melhor tomada de decisões, resultando em um ambiente acadêmico mais dinâmico e adaptável (Policastro, 2016). A literatura aponta que, embora existam muitas propostas teóricas para a implementação de *smart campus*, há uma necessidade urgente de estudos mais aprofundados e práticos que demonstrem sua eficácia (Policastro, 2016). De toda maneira, o desenvolvimento de aplicações e soluções tecnológicas pode resultar em grandes benefícios e para toda a comunidade.

Assim, encontrar soluções que atendam todos os requisitos necessários torna-se um grande desafio, pois depende da participação ativa dos “cidadãos dos campi“ e de adaptar a tecnologia para diversas situações específicas do dia a dia acadêmico. É necessário haver um equilíbrio entre as tecnologias implementadas e os benefícios percebidos por todos, de modo a garantir maior segurança e satisfação. Por isso é crucial atender a fundo as suas demandas e elevar a experiências dos usuários, em vez de se basear apenas em dados analíticos (Schöning, 2014).

2.2 APLICATIVOS MÓVEIS MULTIPLATAFORMA

O desenvolvimento de aplicativos móveis começou a ganhar destaque no final da década de 1990, com a introdução de dispositivos portáteis que permitiam a execução de funções simples. Esses primeiros aplicativos eram limitados em termos de funcionalidade e *design*, focando principalmente em atender a necessidades básicas, como agendas e cal-

culadoras. A verdadeira transformação ocorreu com o lançamento do *iPhone* em 2007 e do sistema *Android* em 2008, que revolucionaram o mercado ao permitir que desenvolvedores criassem aplicativos mais complexos e interativos, aproveitando as capacidades avançadas dos novos *smartphones* (Policastro, 2016). Além disso, a acessibilidade à internet móvel e o aumento da popularidade dos smartphones contribuíram para uma rápida expansão do ecossistema de aplicativos (Ricardo, 2016).

Com o surgimento das lojas virtuais, como a *App Store* da *Apple* e *Google Play* do *Google*, houve um crescimento exponencial no número de aplicativos disponíveis (Oliveria Junior 2013). Essas plataformas não só facilitaram o acesso dos usuários a uma variedade de aplicativos, mas também incentivaram empresas a criar soluções inovadoras em diferentes áreas, como saúde, educação e entretenimento. Essa democratização do acesso à tecnologia móvel possibilitou que pequenos desenvolvedores e *startups* entrassem no mercado, ampliando ainda mais a diversidade de aplicativos disponíveis (Borges, J. R. A.; Santos, 2019).

Atualmente, segundo o IBGE (Instituto Brasileiro de Geografia e Estatistica) em 2019, 98,6% dos domicílios brasileiros possuem algum tipo de aplicativo móvel e são uma parte essencial do cotidiano das pessoas, atuando nas mais diferentes áreas da vida, tais como saúde, educação, finanças, etc (Figueredo, 2017). A evolução contínua das tecnologias móveis promete ainda mais inovações no futuro, com tendências como inteligência artificial e IoT (Internet das Coisas) moldando o desenvolvimento de novos aplicativos. Assim, o histórico do desenvolvimento de aplicativos móveis é marcado por avanços tecnológicos significativos que continuam a transformar a forma como interagimos com o mundo ao nosso redor entre múltiplas plataformas.

O uso de aplicativos móveis tem aumentado significativamente, tornando-se uma prática comum entre os usuários que os empregam para acessar informações, se entreter e aproveitar as vantagens oferecidas por suas funcionalidades (Figueredo, 2017). Neste contexto, surge uma grande demanda por novas soluções diariamente, aumentando a urgência por desenvolvimento rápido e novas tecnologias. Para compreender melhor as possibilidades, é fundamental entender as diferenças entre o desenvolvimento nativo e a multiplataforma, suas vantagens e desvantagens, e seus diferentes objetivos.

A primeira abordagem, o desenvolvimento nativo, refere-se à criação de aplicativos para uma plataforma móvel específica utilizando linguagens de programação e ferramentas que são nativas daquela plataforma. Por exemplo, segundo Figueredo (2017), aplicati-vos para *iOS* são geralmente desenvolvidos em *Swift* ou *Objective-C*, enquanto aplicativos para *Android* são criados em *Java* ou *Kotlin*.

Essa abordagem tem como principais vantagens permitir aos desenvolvedores aproveitar ao máximo as funcionalidades e recursos exclusivo do sistema operacional e do dispositivo, como câmeras, GPS, notificações e padrões de interface nativa do sistema, resultando em um desempenho otimizado e uma experiência fluida e condizente com as

diretrizes da interface do sistema (Figueredo, 2017; Sanches, 2024; El-Kassas *et al.*, 2017). Contudo, aplicações nativas necessitam de muito mais tempo de constru-ção, em comparação a tecnologia híbrida. Estes problemas, aumentam o custo final de desenvolvimento, testes e manutenção do produto (Sanches, 2024; El-Kassas *et al.*, 2017).

> "O objetivo de qualquer empresa de desenvolvimento móvel é atingir o maior número possível de usuários, fornecendo o mesmo aplicativo para diferentes plataformas." (El-Kassas *et al.*, 2017)

Esta afirmação justifica muito bem e de modo geral o objetivo do desenvolvimento multiplataforma, que por sua vez é uma abordagem que permite criar aplicativos que podem ser compilados e executados em várias plataformas com um único código-fonte (El-Kassas *et al.*, 2017). Isso é alcançado através de *frameworks* que abstraem as diferenças entre os sistemas operacionais e criam API (*Application Programming Interface*) de acesso a recursos nativos do sistema, permitindo que os desenvolvedores escrevam o código uma vez e o utilizem em diferentes dispositivos (Padua Junior; Laranjo, 2020).

Vale destacar que a principal vantagem das tecnologias multiplataforma é a reutilização de um único código para o desenvolvimento de soluções que podem ser compiladas para múltiplos sistemas operacionais, tais como *Android*, *iOS, Windows, MacOS, Linux* etc; isso garante agilidade no desenvolvimento, baixo custo de construção, manutenção e teste, além de uma equidade visual em todas as plataformas (El-Kassas *et al.*, 2017).

Apesar dos diversos benefícios, o uso de *frameworks* apresenta uma série de problemas e desafios, especialmente aqueles que utilizam tecnologias *web* (*HTML, CSS, JavaScript* e similares) como parte da estrutura final das aplicações, por não conseguirem aproveitar toda a performance que o dispositivo oferece, além de que equidade visual também pode ser um problema para usuários que preferem a usabilidade distinta do seu sistema operacional (El-Kassas *et al.*, 2017).

2.3 *FLUTTER* EM PERSPECTIVA: VANTAGENS E DESVANTAGENS FRENTE A OUTRAS TECNOLOGIAS DE DESENVOLVIMENTO

O *Flutter* é um *framework* de desenvolvimento de aplicativos criado pela *Google*, lançado inicialmente em 2017. Desde sua introdução, o *Flutter* tem se destacado por permitir a criação de aplicações nativas para dispositivos móveis, *web* e *desktop* com uma única base de código. O principal diferencial do *Flutter* é seu uso da linguagem Dart, que possibilita uma programação reativa e orientada a objetos, facilitando o desenvolvimento de interfaces ricas e responsivas. O *Flutter* utiliza uma abordagem baseada em *widgets*, onde cada componente da interface do usuário é um *widget* que pode ser combinado e modificado para criar layouts complexos (Labornel; Oliveira, C. R., 2024).

O principal destaque do *Flutter* é seu desempenho excepcional devido ao seu motor gráfico, que compila o código diretamente para código nativo, permitindo animações suaves e rápidas. Além disso, o *Hot Reload* (recarga rápida) é uma funcionalidade que permite aos desenvolvedores ver as alterações em tempo real sem perder o estado atual da aplicação, aumentando a produtividade (Labornel; Oliveira, C. R., 2024). O *Flutter* também possui uma rica biblioteca de *widgets* personalizáveis e uma comunidade ativa que contribui com pacotes e *plugins*.

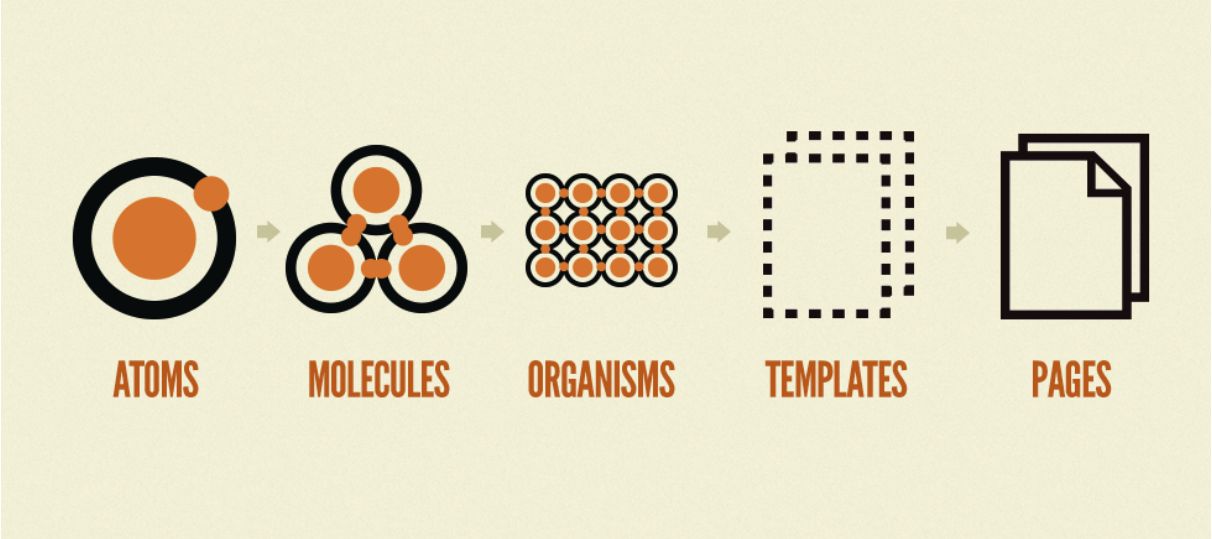
Uma das principais vantagens do *Flutter* é o fato de ser multiplataforma, por meio de seu uso, desenvolvedores podem criar aplicativos para *iOS, Android*, *Web* e *Desktop* com um único código-fonte. Desta forma, é comum a redução significativa do tempo e do custo de desenvolvimento, manutenção e teste, e por consequência atualizações mais ágeis (Padua Junior; Laranjo, 2020). Além disso, a performance é outra vantagem notável; a compilação para código nativo permite que os aplicativos desenvolvidos em *Flutter* tenham um desempenho semelhante aos aplicativos nativos (Flutter, 2024c).

Uma outra vantagem é a experiência do usuário, pois o *Flutter* oferece uma vasta gama de *widgets* personalizáveis que permitem criar interfaces atraentes e responsivas (Flutter, 2024e). Os desenvolvedores também podem facilmente implementar animações complexas que melhoram a experiência do usuário (Flutter, 2024b). A comunidade ativa do *Flutter* contribui constantemente com novos pacotes e *plugins*, em https://pub.dev, facilitando a integração com outras tecnologias e serviços (Flutter, 2024f; Ryan, 2020).

Apesar das suas vantagens, o *Flutter* apresenta algumas desvantagens. Uma delas é o tamanho do aplicativo; pois os que são criados com ele tendem a ser maiores em comparação com os aplicativos nativos. Isso se dá devido à inclusão do motor *Flutter* no pacote. Isso pode ser um fator limitante para usuários com dispositivos com espaço de armazenamento restrito. Outra desvantagem é a curva de aprendizado. Embora Dart, normalmente, seja uma linguagem fácil de aprender para desenvolvedores familiarizados com Java ou JavaScript (Hjort, 2019), aqueles mais habituados com linguagens como Swift ou Kotlin podem enfrentar desafios iniciais. Além disso, como o *F lutter* é relativamente novo em comparação com outras tecnologias como React Native ou Xamarin, pode haver menos recursos disponíveis em termos de documentação e suporte.

Vários casos de sucesso demonstram a eficácia do *F lutter* no desenvolvimento de aplicativos. O uso do *Flutter* permitiu ao *Google* manter uma única base de código enquanto atendia a milhões de usuários em todo o mundo. Um exemplo notável é o aplicativo *Google Ads* (aplicativo para gestão de anúncios do *Google*), que foi desenvolvido usando *Flutter* para oferecer uma experiência consistente em diferentes plataformas. Outro exemplo é o aplicativo Alibaba (vendas e comércio global), que utiliza o *Flutter* para desenvolver partes significativas de suas plataformas móveis. A Alibaba encontrou no *Flutter* uma solução eficaz para atender à demanda por aplicações rápidas e responsivas, destacando-se

Figura 4 – Composição do *Atomic Design* por átomos, moléculas, organismos, modelos e páginas.



Autor: (Frost, 2024)

no mercado competitivo (Uzun, 2023).

2.4 O PODER DO *ATOMIC DESIGN* NA ORGANIZAÇÃO DE PROJETO *FLUTTER*

O *Atomic Design* é uma metodologia de *design* que visa criar interfaces de forma sistemática e escalável. Desenvolvido por *Brad Frost*, esse conceito se baseia na ideia de que os componentes de uma interface podem ser divididos em partes menores, semelhantes a átomos, que se combinam para formar moléculas, organismos, templates e páginas. Essa abordagem permite que *designers* e desenvolvedores criem sistemas de *design* coesos e reutilizáveis, facilitando a manutenção e a consistência visual em projetos complexos (Frost, 2024).

Os princípios fundamentais do *Atomic Design* incluem a hierarquia dos componentes, onde cada nível representa uma complexidade crescente. Os átomos são os elementos mais básicos, como botões e ícones. As moléculas são combinações de átomos que formam unidades funcionais simples, como um campo de formulário com um rótulo. Os organismos são grupos de moléculas que criam seções distintas de uma interface, enquanto os *templates* organizam esses organismos em um *layout* específico. Por fim, as páginas são instâncias concretas dos templates que mostram o conteúdo real (Frost, 2024).

Uma das principais vantagens do *Atomic Design* é a sua capacidade de facilitar a manutenção de projetos. Como os componentes são construídos de forma modular e reutilizável, alterações em um único átomo ou molécula podem ser propagadas automaticamente para todas as instâncias onde esses componentes são utilizados. Isso reduz significativamente o tempo e o esforço necessários para implementar mudanças ou correções (Frost, 2024).

Esse conceito pode facilmente ser utilizado em aplicações construídas em qualquer tecnologia, como diz (Frost, 2024). No *Flutter*, em específico, o *Atomic Design* é particularmente eficaz devido à natureza modular do *framework*, pois este permite a criação de *widgets* reutilizáveis, que podem ser facilmente combinados para formar interfaces complexas. Ao adotar a abordagem do *Atomic Design*, os desenvolvedores podem estruturar seus *widgets* em diferentes níveis, começando pelos átomos (como botões e textos) até chegar a organismos mais complexos (como formulários ou listas) (Frost, 2024).

2.5 *FIREBASE*: VANTAGENS, DESVANTAGENS E INTEGRAÇÕES EFICIENTES

O *Firebase* é uma plataforma de desenvolvimento de aplicativos móveis e *web*, criada pelo *Google*, que oferece uma variedade de ferramentas e serviços para facilitar a criação, o gerenciamento e a escalabilidade de aplicativos com o auxílio de inteligência artificial. Ele fornece funcionalidades como autenticação, armazenamento em nuvem, banco de dados em tempo real, hospedagem e notificações *push*, permitindo que os desenvolvedores se concentrem na lógica do aplicativo e na experiência do usuário, sem a necessidade de se preocupar com a infraestrutura subjacente (Firebase, 2024a). O *Firebase* é especialmente popular entre desenvolvedores *Flutter* devido à sua integração fácil e ao suporte robusto para aplicações em tempo real (Tech, 2024; Gonçalves *et al.*, 2023).

A principal integração realizada com o *Firebase* neste trabalho foi com o *Firebase Firestore Database*, que é um banco de dados escalável e flexível que permite armazenar e sincronizar dados entre clientes em tempo real. Ele organiza os dados em documentos e coleções, facilitando a maioria das consultas e permitindo que os desenvolvedores criem aplicativos responsivos que atualizam automaticamente as informações na interface do usuário sem necessidade de recarregar a página (Araújo, 2021).

A segunda, foi com o *Firebase Cloud Messaging* é um serviço que permite enviar mensagens e notificações para dispositivos móveis ou web. Com o ele os desenvolvedores podem se comunicar com os usuários em tempo real, enviando alertas sobre atualizações ou eventos importantes, melhorando assim a experiência do usuário e aumentando o engajamento com o aplicativo (Firebase, 2024c).

Por fim, apesar das muitas vantagens e usos práticos, existem algumas desvantagens ao usar o *Firebase*. Uma delas é dependência da plataforma, já que os desenvolvedores ficam atrelados às limitações e políticas do *Google*. Além disso, custo pode ser um fator preocupante; embora o *Firebase* ofereça um plano gratuito, custos podem aumentar rapidamente conforme o uso se expande (Gonçalves *et al.*, 2023). Outra desvantagem é a complexidade em consultas no *Firestore*, principalmente com grande quantidade de dados; pois como ele é *NoSQL*, que se refere a bancos de dados que não utilizam a estrutura tabular tradicional e são projetados para escalabilidade e flexibilidade, algumas operações complexas podem ser mais difíceis de implementar e lentas, em comparação com bancos de dados relacional e tradicionais (SQL) (Gonçalves *et al.*, 2023; Araújo, 2021;

Firebase, 2024b).

2.6 ANÁLISE DE SENTIMENTOS COM INTELIGÊNCIA ARTIFICIAL - HUGGING FACE

Na área da computação, segundo Silva e Serrano (2023), a análise de sentimentos refere-se ao exame das opiniões, emoções, humor e sentimentos expressos por indivíduos em textos. Nesse contexto, é viável realizar uma análise de ânimo, que investiga se as emoções e/ou sentimentos contidos no texto são positivos, neutros ou negativos, utilizando PLN (Processamento de Linguagem Natural). Além disso, a análise de sentimentos inclui a detecção de emoções, que serve para classificar dados textuais de acordo com a emoção que o texto provoca.

Assim, a análise de sentimentos com IA (Inteligência Artificial) emergiu como uma ferramenta valiosa para a compreensão das emoções e opiniões expressas em textos digitais. Em um mundo onde as interações online proliferam, a capacidade de interpretar o que os usuários realmente pensam sobre produtos, serviços e marcas torna-se crucial para o sucesso dos negócios grandes e pequenos, principalmente no *marketing* e divulgação (Jesus; Ferreira, 2020; Dourado *et al.*, 2024). Este campo combina linguística e tecnologia, utilizando algoritmos avançados para transformar dados textuais em PLN em informações úteis que podem guiar decisões estratégicas e melhorar o entendimento e o relacionamento com o cliente.

A utilização da IA para analisar comentários dos usuários apresenta vantagens e desvantagens. Entre as vantagens, destaca-se a capacidade de processar grandes quantidades de dados rapidamente, permitindo uma análise em tempo real das opiniões dos consumidores (Silva, M. E. R. d.; Serrano, 2023). Além disso, a IA pode identificar padrões e tendências que poderiam passar despercebidos em análises tradicionais. Por outro lado, as desvantagens incluem a possibilidade de viés nos modelos, que podem levar a interpretações incorretas dos sentimentos expressos. Além disso, a análise automatizada pode falhar em captar nuances emocionais ou contextos culturais que um humano entenderia facilmente (Silva, M. E. R. d.; Serrano, 2023).

Uma excelente opção para esse tipo de análise é a *Hugging Face*, que é uma plataforma de IA de código aberto liberada sob a licença *Apache 2.0* e está disponível no GitHub em https://github.com/huggingface/transformers. Esta IA se destaca no desenvolvimento e compartilhamento de modelos de processamento de linguagem natural (Jain, 2024; Wolf *et al.*, 2020a). Com uma comunidade ativa e uma vasta biblioteca chamada *Transformers*, o *Hugging Face* oferece acesso a uma ampla gama de modelos pré-treinados que facilitam tarefas como tradução automática, geração de texto e análise de sentimentos. A plataforma também promove a colaboração entre pesquisadores e desenvolvedores, permitindo que inovações em IA sejam rapidamente compartilhadas e implementadas em diferentes aplicações (Jain, 2024; Wolf *et al.*, 2020a).

O modelo utilizado neste trabalho foi o *nlptown/bert-base-multilingual-uncased-sentiment*, que foi projetado para realizar análise de sentimentos em múltiplos idiomas no *Hugging Face*. Baseado na arquitetura BERT (*Bidirectional Encoder Representations from Transformers*), ele foi treinado para classificar sentimentos em cinco categorias, variando de muito negativo a muito positivo. Sua capacidade multilíngue permite que seja utilizado em diversos contextos linguísticos, tornando-o uma ferramenta valiosa para qualquer projeto ou negócio. Ao empregar esse modelo, as organizações podem obter *insights* sobre as percepções dos usuários em diferentes idiomas, melhorando assim sua estratégia de comunicação e engajamento com o cliente (Hugging Face, 2024).

3 MATERIAL E MÉTODOS

O aplicativo Intercampi foi desenvolvido utilizando a tecnologia *Dart* e *Flutter*, nas versões 3.0.5 e 3.10.5, respectivamente. Essa escolha se justifica por se tratar de uma tecnologia moderna, de alto desempenho e que permite a criação de aplicativos multiplataforma, ou seja, que podem ser executados em diversos sistemas operacionais, como *Android*, *iOS, Linux, macOS, Windows* e *Web*.

Uma das principais vantagens do uso de *Dart* e *Flutter* é a possibilidade de gerar um único código-fonte que pode ser compilado para diferentes plataformas. Isso significa que, com apenas uma base de código, é possível criar aplicativos que rodam em múltiplos sistemas operacionais, requerendo apenas pequenos ajustes específicos em casos particulares.

O desenvolvimento multiplataforma traz diversos benefícios. A eficiência é um deles, pois ao desenvolver uma única base de código, os esforços de manutenção e atualização são significativamente reduzidos. Outro benefício é a consistência, uma vez que a experiência do usuário pode ser mantida uniforme entre as diferentes plataformas ou customizadas, caso haja preferência. Além disso, a abordagem multiplataforma proporciona agilidade, permitindo que a equipe de desenvolvimento se concentre em entregar funcionalidades de forma mais rápida, sem precisar se preocupar com a adaptação para cada sistema operacional.

O código do aplicativo foi disponibilizado ao público na plataforma *GitHub* em https://github.com/EdesonABizerril/intercampi_unilab. Este projeto é classificado como de código aberto (*Open Source*), permitindo que qualquer interessado visualize, modifique e contribua para seu desenvolvimento.

3.1 CRIAÇÃO DE UM NOVO PROJETO

Durante a criação de um novo projeto com *Flutter*, é possível selecionar os sistemas operacionais que o aplicativo irá suportar. No caso do aplicativo Intercampi, foram habilitadas as plataformas *Android* e *iOS*. Existem diversas formas de criar um projeto e escolher quais plataformas serão utilizadas, e isso depende principalmente da ferramenta empregada para desenvolver o aplicativo.

Para criar um projeto *Flutter* que dê suporte às plataformas *Android* e *iOS*, o desenvolvedor deve executar o seguinte comando no terminal: "`flutter create -platforms android, ios nome_do_projeto`". Dessa forma, a base do aplicativo é criada e é possível abrir e acessar todos os arquivos do projeto através de um editor de código, como o *Visual Studio Code* ou o *Android Studio* (Flutter, 2024a).

No entanto, para que todas as dependências necessárias sejam instaladas, é preciso ter conexão com a internet para baixá-las dos repositórios oficiais e, em seguida, executar o comando "`flutter pub get`" no terminal. Esse comando irá baixar e instalar todas as

bibliotecas e pacotes necessários para o projeto (Flutter, 2024a).

3.2 EXECUTANDO O APLICATIVO

Diferentemente do *Android*, que pode ser executado em diversos sistemas operacionais como *Windows, MacOS e Linux*, os aplicativos para *iOS* só podem ser desenvolvidos e testados em computadores com *MacOS* da *Apple*. Portanto, para desenvolver e testar aplicativos para *iOS*, é necessário utilizar um computador *Mac* (IBM, 2024). E, independentemente da plataforma escolhida, *Android* ou *iOS*, é imprescindível utilizar ao menos um dispositivo móvel compatível para qualquer etapa do desenvolvimento, desde a criação até os testes finais.

Existem duas abordagens comuns para realizar essa tarefa: a primeira consiste em executar em um dispositivo físico, enquanto a segunda refere-se à execução em dispositivos virtuais. Os dispositivos físicos, nada mais são do que os celulares que todo mundo utiliza, de qualquer marca ou modelo, conectados ao computador via cabo USB. E os dispositivos virtuais, conhecidos popularmente como emuladores, podem ser utilizados (semelhante a dispositivos físicos) em computadores com sistemas operacionais compatíveis.

3.3 EXECUTANDO EM DISPOSITIVO REAIS

Para executar o aplicativo em um dispositivo *Android* real durante o desenvolvimento, é necessário habilitar o modo de depuração nas opções de desenvolvedor do dispositivo. Essas configurações estão presentes em todas as versões do sistema, mas são ocultas por padrão, pois apenas desenvolvedores interessados precisam alterá-las.

O procedimento exato para habilitar as opções de desenvolvedor pode variar de acordo com o modelo e versão do *Android*, portanto, é recomendado consultar a documentação específica do dispositivo. Em geral, é possível acessar essa opção nas configurações do sistema, no menu "Sobre o telefone" e tocando repetidamente no número do build até que as opções de desenvolvedor sejam desbloqueadas.

Assim, após conectar o dispositivo *Android* ao computador via USB e habilitar o modo de depuração (desenvolvimento), é possível selecionar o dispositivo no IDE (*Integrated Development Environment*) – Ambiente Integrado de Desenvolvimento - e executar o comando "`flutter run`" para iniciar a aplicação. Durante o desenvolvimento, o *Flutter* utiliza um modo de execução que prioriza a flexibilidade e a rapidez na visualização de alterações. Para executar com alta performance durante o desenvolvimento, a melhor alternativa é o modo profile, pois esta é uma configuração que permite analisar o desempenho de forma mais detalhada, equilibrando a capacidade de depuração com a eficiência de execução (Flutter, 2024d). Para executar o aplicativo em dispositivos virtuais, o procedimento é muito semelhante ao utilizado em dispositivos reais. A principal diferença é que, neste caso, tudo é executado no computador, sem a necessidade de utilizar um

dispositivo físico. Essa forma de utilização do aplicativo é bastante útil durante grande parte do desenvolvimento, pois aumenta a produtividade ao dispensar o uso de periféricos. No entanto, diversos procedimentos e testes só podem ser realizados em dispositivos reais.

É importante destacar que, no sistema operacional MacOS, é possível criar emuladores tanto para *Android* quanto para *iOS*. Já em outros sistemas operacionais, como *Windows e Linux*, não é possível criar emuladores para *iOS*. O procedimento para executar o aplicativo em emuladores é exatamente o mesmo utilizado para dispositivos reais, ou seja, basta selecionar o emulador desejado na IDE e executar o comando "`flutter run`".

3.4 SERVIDOR WEB PARA SINCRONIZAÇÃO ONLINE E OFFLINE

O *Firebase* foi escolhido como o *backend* ou servidor online para o aplicativo Intercampi devido a diversos fatores positivos dessa plataforma, tais como: serviços prontos para uso, facilidade de implementação, alta disponibilidade e segurança, além de uma excelente documentação e suporte oficial para aplicações *Flutter*. A maior parte dos serviços disponíveis são gratuitos, com limites altos de uso, sendo geralmente suficientes para aplicações de pequeno porte.

Portanto, o *Firebase* atende bem às necessidades do aplicativo Intercampi, fornecendo uma plataforma robusta, segura e de fácil integração, especialmente por se tratar de um projeto *Flutter*.

3.5 ANÁLISE DOS SENTIMENTOS DOS USUÁRIOS EM SEUS FEEDBACKS

Para uma analisar de forma detalhada os *feedbacks* e notas atribuídas ao aplicativo ao longo do tempo, foi necessário analisar alguns gráficos exportados do *Google Play* para um melhor entendimento e compreensão sobre a distribuição das notas e comentários. No total foram **301 avaliações** entre 2017 a 2024 com notas que variam de 1 a 5 (inclusive), e dentre elas **99 comentários**.

Para evitar preferências pessoais ou favoritismos durante a análise das opiniões, todos os dados foram exportados do *Google Play*, tratados e ajustados por funções do Excel (Anexo 1), e processados por uma IA disponibilizada pela plataforma Hugging Face Transformers (Anexo 2), utilizando seu modelo multilingue `nlptown/bert-base-multilingual-uncased-sentiment` (Hugging Face, 2024). Um exemplo dos resultados pós-processamento pela IA pode ser encontrado no Anexo 3.

4 RESULTADOS E DISCUSSÃO

4.1 APLICATIVO INTERCAMPI E SUAS FUNCIONALIDADES

O aplicativo Intercampi é uma solução importante para a mobilidade acadêmica, facilitando o acesso aos horários de ônibus que conectam os diferentes campi da UNILAB. Este aplicativo visa otimizar o deslocamento dos estudantes e colaboradores, oferecendo informações em tempo real sobre as rotas e horários de saída, promovendo uma experiência mais eficiente e integrada. Esta ferramenta busca, para além de contribuir para a redução do tempo de espera, também refletir o compromisso com a comunidade acadêmica para a modernização dos serviços oferecidos pela instituição.

4.1.1 Requisitos iniciais para o desenvolvimento

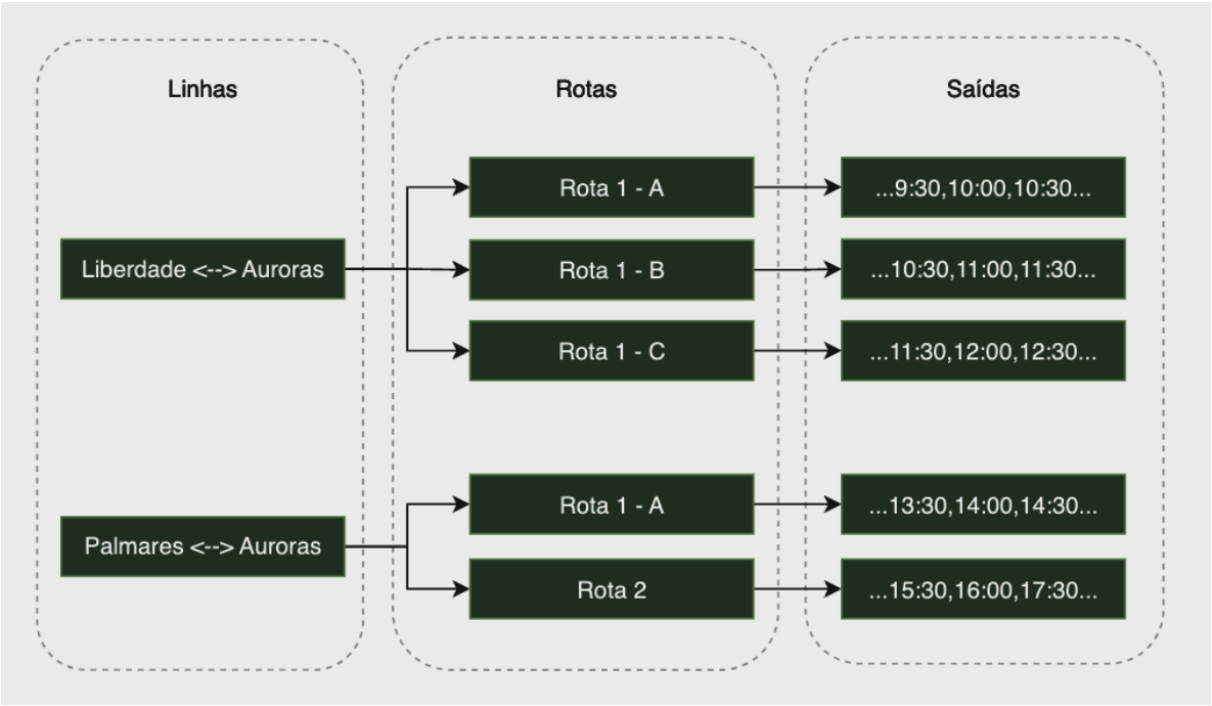
Assim, em 2017, para dar início à criação do aplicativo, foi necessário desenvolver uma solução que atendesse a todas as necessidades de acesso aos horários e ônibus, disponível em qualquer local e situação de uso. O objetivo era criar um aplicativo com sincronização *online* automática e acesso a todos os dados *offline*. Quando o aplicativo fosse inicializado *online*, todos os horários e rotas deveriam ser baixados ou atualizados automaticamente, sem a necessidade de intervenção do usuário. Quando utilizado *offline*, o aplicativo deveria ter todos os dados igualmente disponíveis, prontos para uso, pois nem sempre a conexão com a internet era uma realidade.

Para facilitar a compreensão e o desenvolvimento das funcionalidades, foram definidos alguns padrões de nomenclatura para todos os dados disponibilizados pela Universidade. As informações foram divididas em Linhas, Rotas e Saídas. Os percursos entre os campi da Universidade são chamados de Linha, por exemplo, o percursos entre os campi Liberdade e Aurora é uma Linha, e, da mesma maneira, Liberdade e Palmares é outra Linha. Todos os ônibus de uma mesma Linha podem ter várias Rotas e nomes distintos. Por último, as Saídas são os horários de saída de cada ônibus.

A Figura 5 mostra três áreas delimitadas por linhas tracejadas. À esquerda, há dois exemplos de Linhas - uma correspondendo ao trajeto entre o campus da Liberdade e o Auroras, e outra entre Palmares e Auroras. É importante notar que existem muitas variações entre os três campi, sendo esses apenas dois exemplos ilustrativos. No centro da imagem, estão representadas as rotas, que também podem variar bastante. Uma Linha pode ter apenas uma rota ou múltiplas rotas.

Por exemplo, a primeira Linha possui três rotas, enquanto a segunda possui apenas duas. O conteúdo à direita corresponde às listas de horários de cada saída de uma rota. Assim, em resumo, a imagem apresenta de forma esquemática a estrutura do sistema de transporte, com Linhas, rotas e horários, evidenciando a complexidade e variabilidade desse sistema nos diferentes campi.

Figura 5 – Exemplo de Linhas, rotas e saídas.



Fonte: (Bizerril, F. E. A., 2024)

4.1.2 Sincronização em tempo real

O aplicativo precisa realizar sincronizações automáticas e disponibilizar os dados mesmo na ausência de conexão com a internet. Portanto, para que o aplicativo possua sincronização em tempo real, é necessário criar um servidor *web* capaz de armazenar todos os dados e permanecer disponível 24 horas por dia, 7 dias por semana. Outro fator de extrema importância é o baixo custo de operação, visto que o aplicativo não possui investidores ou quaisquer meios de monetização. Essa condição configura um desafio, pois foi necessário encontrar um servidor de alta qualidade e preferencialmente gratuito. Após ampla pesquisa no mercado, a solução mais adequada, que resolve todos os problemas e ainda disponibiliza outros serviços úteis gratuitamente, foi o *Firebase*.

Se, em vez do *Firebase Firestore*, fosse utilizada outra API REST (*Representational State Transfer*) qualquer, seria necessário realizar manualmente todas as requisições de sincronização ou empregar uma solução baseada em *Websocket*, o que implicaria em maior complexidade. Essa abordagem apresentaria elevada dificuldade d e d esenvolvimento e custos significativos com o servidor de dados. Outro problema é a necessidade de criar manualmente uma estrutura de *cache* para armazenar os últimos dados recebidos. No entanto, com o *Cloud Firestore*, todo esse processo de implementação da *cache* e sincronização é realizado automaticamente.

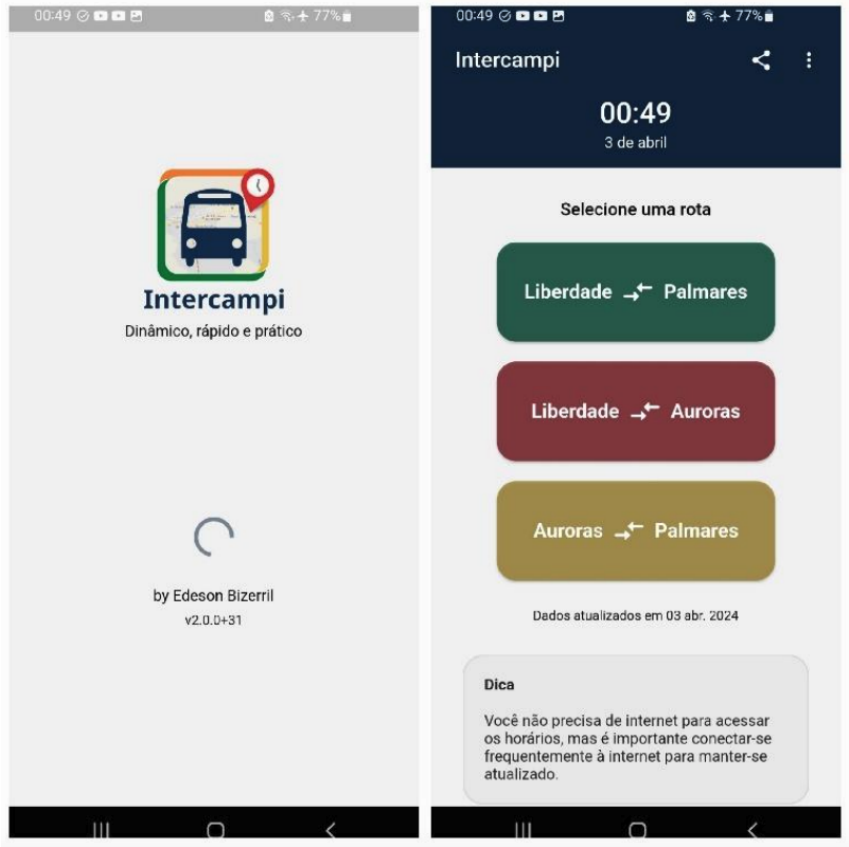
O processo de sincronização dos horários é iniciado após a instalação do aplicativo, em seu primeiro uso. Nesse momento, todos os dados são baixados e armazenados localmente, com o objetivo de serem acessados mais rapidamente e sem conexão com a internet

no segundo acesso em diante. O processo de sincronização dos dados é realizado sempre que novos horários e rotas são atualizadas, quando o usuário inicia o aplicativo, e não acontece em segundo plano. Todo esse processo de sincronização é realizado automaticamente com o uso do *Firebase Firestore* durante o carregamento inicial do aplicativo.

4.1.3 Regras de negócio

O aplicativo inicia em uma tela temporária, conhecida como *Splash Screen*, mostrado na Figura 6(a), cujo objetivo é auxiliar na busca por dados salvos em *cache*, inicializar serviços e exibir informações para o usuário. Em seguida, o usuário é direcionado para a tela inicial (*Home*), mostrada na Figura 6(b), que contém 3 botões para acesso às Linhas disponíveis (Liberdade – Palmares, Liberdade – Auroras e Auroras - Palmares), a data e a hora do dia para facilitar a consulta, a data da última atualização dos dados e uma mensagem sobre o funcionamento da sincronização dos dados.

Figura 6 – **a)** tela de carregamento. **b)** tela principal (home)

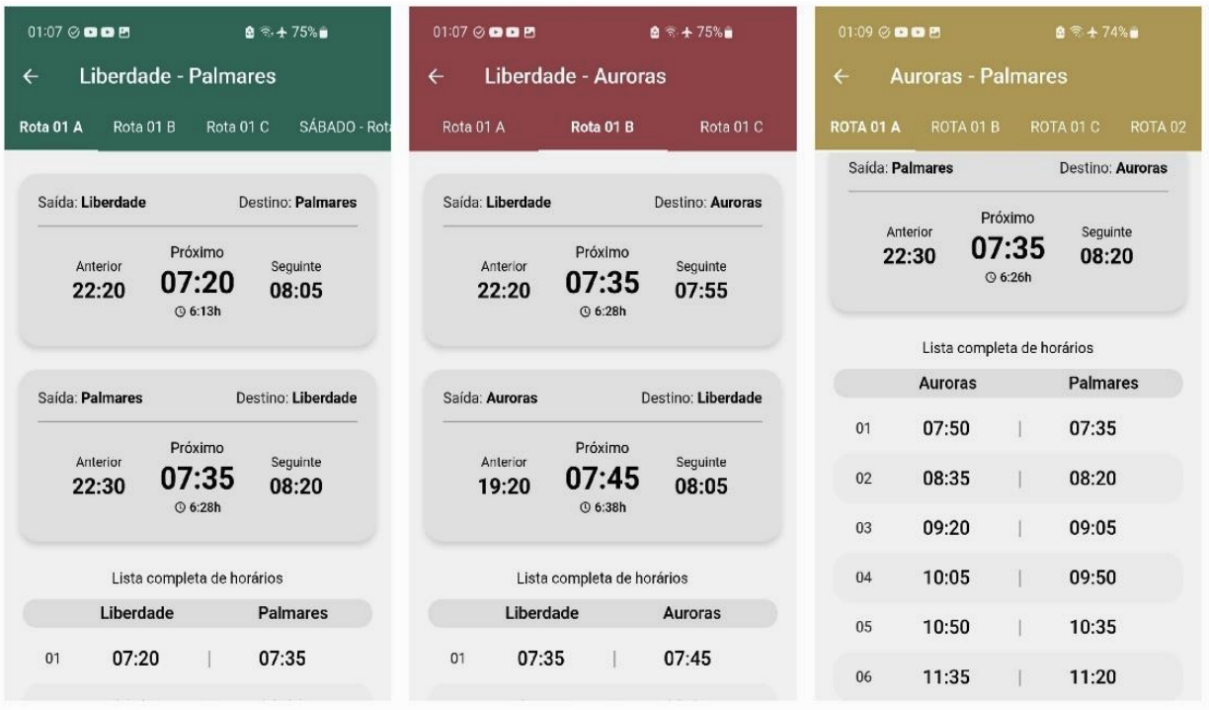


Fonte: (Bizerril, F. E. A., 2024)

Ao clicar em um dos botões de Linha, uma nova tela é aberta com diversas abas, onde cada uma representa uma rota. Se houver 3 rotas, serão exibidas 3 abas, com seus respectivos nomes e lista de horários (Figura 7). Cada rota inicia com a mesma cor da Linha correspondente, facilitando a leitura rápida, e cada uma possui três componentes: o painel de resumo dos horários de saída do ônibus atual saindo de um dos campi, o

painel de resumo dos horários de saída do ônibus atual saindo do outro campus, e a lista completa e enumerada de todos os horários, para consulta geral.

Figura 7 – Telas internas das três Linhas entre Auroras, Liberdade e Palmares



Fonte: (Bizerril, F. E. A., 2024)

Cada resumo de horários possui alguns dados importantes. Primeiro, na Figura 8 há a indicação da *Saída* e do *Destino* de cada Linha. Por exemplo, na imagem mais à esquerda da Figura 7, tem-se a *Saída* do campus da Liberdade em direção à chegada no Palmares. Logo abaixo, há três horários: *Anterior*, *Próximo* e *Seguinte*. O *Anterior* corresponde ao horário do ônibus que já passou, o último que saiu. O *Próximo* indica o horário em que o próximo ônibus virá. E o terceiro, o *Seguinte*, indica o horário que virá em seguida, logo após o *Próximo*. Por fim, há a indicação do tempo que ainda resta para que o próximo ônibus chegue, ou seja, o *Próximo*.

Para abordar essa questão, será utilizada uma representação matemática que permita identificar os horários anteriores, próximos e seguintes com base na hora atual do dispositivo do usuário. Essa abordagem se fundamenta na lógica de comparação entre a hora corrente e uma lista ordenada de horários.

Inicialmente, há definido uma lista de horários L, que pode ser expressa como $L = h_1, h_2, h_3, \ldots, h_n$, onde cada elemento h_i representa um horário específico. A hora corrente, obtida do dispositivo do usuário, é denotada por H_c. O objetivo é encontrar o menor índice i tal que o horário em $L[i]$ seja maior ou igual a H_c.
Esse índice pode ser determinado pela expressão:

$$i = argmin_j(L[j] \geq H_c)$$

Figura 8 – Representação da hora Anterior, Próximo, Seguinte e Tempo Restante para a próxima saída.

Fonte: (Bizerril, F. E. A., 2024)

Aqui, o termo *argmin* refere-se ao índice que minimiza a condição apresentada, ou seja, busca o menor índice j para o qual a condição é satisfeita. Essa operação é crucial para identificar rapidamente o próximo ônibus disponível. Assim, uma vez identificado o índice i, pode-se determinar os horários anterior (H_{ant}), próximo (H_{prx}) e seguinte (H_{seg}) da seguinte forma:

O horário anterior é dado por:

$$H_{ant} = L[i-1] \quad \text{se } i > 0$$

Caso contrário, se $i = 0$, consideramos o último horário da lista:

$$H_{ant} = L[n-1]$$

O horário Próximo H_{prox} é dado por:

$$H_{prox} = L[i]$$

O horário seguinte H_{seg} é definido como:

$$H_{seg} = L[i+1] \quad \text{se } i < n-1$$

Se $i = n-1$, então consideramos o primeiro horário da lista:

$$H_{seg} = L[0]$$

Além disso, para calcular o tempo restante até a chegada do próximo ônibus, utiliza-se a seguinte fórmula:

$$T_{rest} = H_{prx} - H_c$$

Essa abordagem matemática não apenas proporciona uma maneira clara e estruturada de selecionar horários de ônibus, mas também facilita a implementação em sistemas computacionais. A utilização da notação $argmin$ e a definição rigorosa dos índices garantem que as operações sejam executadas de forma eficiente e precisa. Assim, essa metodologia pode ser aplicada em sistemas de transporte público para melhorar a experiência dos usuários ao fornecer informações atualizadas sobre os horários dos ônibus, como foi o caso do aplicativo Intercampi.

4.2 ARQUITETURA E TECNOLOGIAS EMPREGADAS

Para a construção de um aplicativo, é essencial priorizar uma boa organização, qualidade de código, escalabilidade e facilidade na manutenção de futuras estruturas, não apenas pelo desenvolvedor inicial, mas também por outras pessoas. O uso de arquiteturas conhecidas e bem documentadas é fundamental e possui um grau de relevância muito alto. Tendo isso em consideração, foi utilizado um padrão arquitetural chamado Getx Pattern (Murakami, 2024a) que tem como base a mistura da arquitetura MVC (*Model View Controller*) e o paradigma reativo (Murakami, 2024b), o qual possui uma grande comunidade envolvida e foi empregado e testado em projetos pequenos, médios e grandes. Embora este projeto não seja de grande porte, mas sim bastante pequeno, é válido utilizar uma arquitetura poderosa, pois o aplicativo já estará preparado para crescer no futuro.

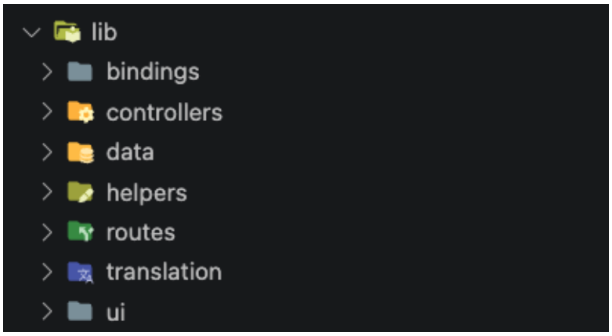
Todos os arquivos que compuseram o aplicativo ficaram por padrão dentro do diretório chamado *'lib'*, onde foi possível construir toda a arquitetura necessária para a uma solução adequada. Lembrando que o desenvolvedor não é obrigado a utilizá-lo, mas é um padrão estabelecido pela equipe de desenvolvimento do *Flutter*, como também aprovado por toda a comunidade de código aberto do mundo, por isso uma decisão lógica utilizar o diretório *'lib'* como o principal para armazenar todos os arquivos.

O gerenciador de estado utilizado foi o Getx (Borges, J., 2024a) e a arquitetura adotada nesse projeto foi a Getx Pattern (Murakami, 2024a), devido a sua grande popularidade e excelente documentação. Basicamente todos os arquivos foram organizados e padronizados em sete camadas:

- *Bindings:* nesta camada é possível fazer todas as configurações de injeção ou inversão de dependências;
- *Controller:* nela será codificada toda a parte lógica da aplicação e gerenciará o estado de todas as telas;
- *Data:* está é a camada em que todos os dados serão recebidos e manipulados, vindos de um servidor remoto ou *cache* (armazenamento local);
- *Helpers:* está camada contém diversas classes que ajudarão durante todo o processo de desenvolvimento, como máscaras, validadores e funções auxiliares;
- *Translation:* camada responsável por toda a parte de tradução e internacionalização do aplicativo;
- *UI: User Interface*, está camada é responsável pela interface com usuário.

Na camada *Bindings*, ficam todas as injeções de dependências. Nestes arquivos são configuradas a maior parte das instâncias de classes do projeto. Embora esse conceito não seja facilmente entendido por programadores iniciantes, é necessário para permitir que nenhuma classe possua implementação direta de qualquer outra dentro de si, mas sim recebendo-a via construtor, a fim de possibilitar a criação de testes unitários no futuro.

Figura 9 – Estrutura de pastas do projeto



Fonte: (Bizerril, F. E. A., 2024)

Um outro fato muito importante presente nos *Bindings* são as configurações de gerenciamento de dependências. Nesse caso, o pacote Getx faz todo o procedimento automaticamente, a fim de manter o aplicativo com o menor uso de memória possível. Basicamente, a maior parte das instâncias criadas na aplicação são inicializadas no *Lazy Mode* (modo preguiçoso), que permite que somente quando alguma rota do aplicativo necessitar de um recurso, recurso, apenas então esse elemento seja adicionado à memória. Uma vez que o usuário sai da tela ou da rota, o gerenciador de estado o remove, liberando, assim, memória *RAM*.

A camada de *Controller* pode ser dividida em duas partes, apesar de todo o código estar no mesmo arquivo. Teoricamente, temos a parte lógica e o gerenciamento de estado. Na parte lógica, foram adicionadas todas as regras de negócio, que correspondem ao fluxo principal e fluxos de exceção. O fluxo principal corresponde ao caminho mais óbvio que o usuário percorrerá no aplicativo, enquanto os fluxos de exceção são as condições que raramente ocorrem, como as situações de erro. Já o gerenciamento de estado corresponde à atualização das partes visuais do aplicativo. Ou seja, quando a informação for baixada ou alterada, as diversas partes da tela precisam ser atualizadas, e esse controle é chamado de gerenciamento de estado.

A camada de *Data* será responsável por receber, enviar e gerenciar todas as informações do aplicativo via *internet*, além de salvar e excluir dados da cache (armazenamento local). Nela teremos subcamadas chamadas de *repository* e *providers*. A partir do *repository*, o *controller* pode solicitar os dados que precisar, como por exemplo, os horários de uma rota específica, e o *repository* determinará se as informações virão de um servidor *online* ou local através dos *providers*. Já os *providers* terão a implementação específica da *API* ou da *cache* para buscar, salvar e gerenciar os dados. Em resumo, o *controller* solicita dados ao *repository*, e um ou mais providers retornam os dados solicitados para o *repository*, que por sua vez, manipula os dados e os retorna prontos ao *controller*.

A camada de *Helpers* é uma camada auxiliar bastante simples, na qual são adicionadas pastas contendo classes, funções e enumerações para auxiliar em diversas tarefas em todo o aplicativo. Por exemplo, quando é necessário converter datas e horários para um determinado formato, e utilizar esta conversão para exibição de informações em tela ou na manipulação de dados, pode-se criar um *helper* para fazer essas operações. Outro exemplo de uso dos *Helpers* é adicionar pré-configurações de diálogos e notificações diversas do aplicativo, facilitando assim a sua chamada em qualquer lugar.

A camada de *Translation* é responsável por armazenar e gerenciar todos os textos em diversas línguas. Essa camada não utiliza nenhum tipo de armazenamento local, mas sim é escrita por desenvolvedores em classes correspondentes a cada idioma. Cada texto apresentado no aplicativo pode ser traduzido para diversas línguas, seguindo as configurações do usuário ou do seu dispositivo. Por exemplo, se houver tradução em português e inglês, teremos duas classes responsáveis, uma para cada língua. Em ambas, haverá uma organização de textos do tipo chave-valor, como `{"route_name": "Nome da rota"}` em português, e em inglês `{"route_name": "Route Name"}`. Quando um texto precisar ser exibido ou manipulado, deverá ser utilizada uma *key*, por exemplo `"route_name".tr`, e o gerenciador de tradução (`.tr`) retornará o texto refere à `"route_name"` com base no idioma definido no dispositivo naquele momento.

A camada de *UI* corresponde à camada visual da aplicação. No *Flutter*, todos os componentes visuais são chamados de *widgets* - um botão, um texto, uma imagem, etc. Para compor e organizar esta camada, foi utilizado o conceito de *Atomic Design*. Esse conceito visa separar todos os componentes em subcamadas, seguindo os conceitos fundamentais da química, como átomos, moléculas, organismos, templates e páginas (Figura 4).

4.2.1 Servidor web - *Firebase*

O *Firebase* é um conjunto de serviços de computação em nuvem de *backend* e plataformas de desenvolvimento de aplicativos fornecidos pelo *Google*. Com o *Firebase*, foi possível criar um servidor *online* para qualquer aplicação em poucos minutos, permitindo o uso de funcionalidades em tempo real. Serviços como envio de notificações, hospedagem, acompanhamento de performance e diversas outras funcionalidades importantes para qualquer produto digital podem ser aplicados com muita facilidade. Além disso, para usos mais específicos, é possível hospedar bancos de dados não relacionais, serviços, autenticação e integração para uma variedade de tipos de aplicações, incluindo *Android*, *iOS, JavaScript, Node.js, Java, Unity, PHP, C++*, bem como o *Flutter* (Dhiman; Choudhary; Choudhary, 2023).

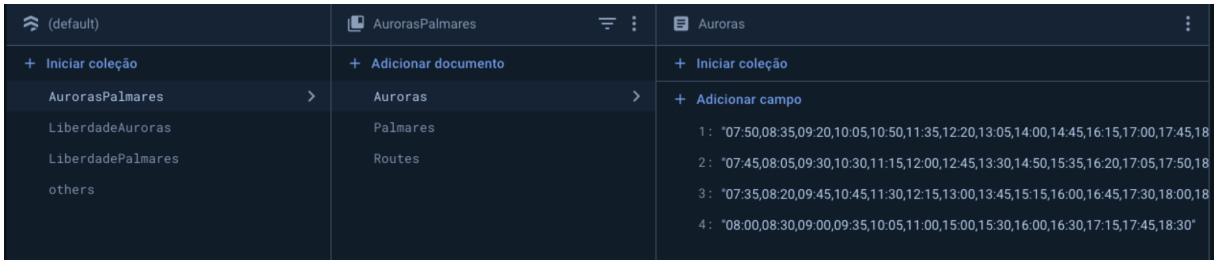
Porém, dentre todas as funcionalidades oferecidas pelo *Firebase*, o aplicativo Intercampi utilizou apenas os serviços de *Cloud Firestore* e *CloudMessaging*. O *Cloud Firestore* foi usado para realizar operações CRUD (*Create, Read, Update* e *Delete*) de dados, ou seja, criar, ler, atualizar e remover informações do servidor *online* quando necessário. Já

o *CloudMessaging* é responsável pelo serviço de *push*, que envia notificações aos usuários mesmo quando o aplicativo estiver fechado.

4.2.2 *Cloud Firestore*

Com o *Cloud Firestore*, foi possível criar um banco de dados não relacional utilizando informações do tipo chave-valor. Nele, adicionamos informações sobre Linhas, que representam o trajeto entre dois campus, como por exemplo, saindo do campus "A" para o campus "B". Além disso, incluímos informações sobre rotas, que representam os ônibus disponíveis para cada *Linha*, por exemplo, `"Rota 1 - A", "Rota 1 - B", "Rota 2 - A"` e etc. Por último, as listas de horários para cada rota seguem o formato de 24 horas e cada saída deve ser separada por vírgula: "`[...09:20,10:05,10:50,11:35...]`"

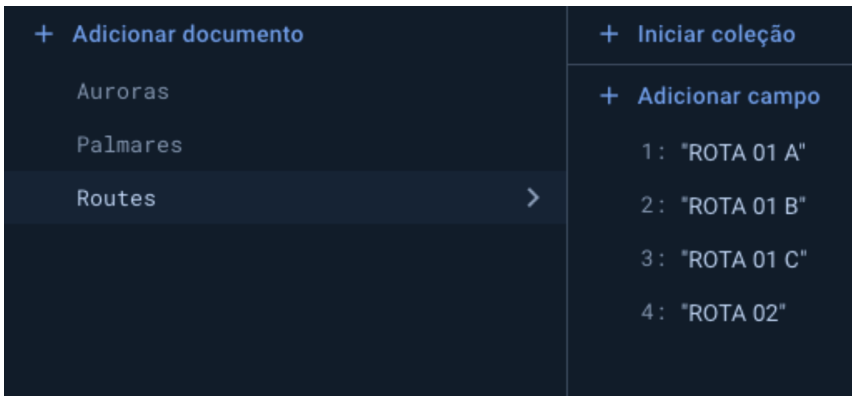
Figura 10 – *Firestore*: coleções, documentos e listas de dados



Fonte: (Bizerril, F. E. A., 2024)

A Figura 10 é dividida em três colunas: a primeira coluna exibe as Linhas existentes atualmente (coleções), a coluna central mostra a saída de cada *campus* (documentos), e a coluna da direita apresenta quatro rotas e seus respectivos horários. Pode-se observar que na primeira coluna há o item "*others*", que foi criado para conter alguns pequenos detalhes não muito importantes presentes no aplicativo, como dicas e afins. Já na coluna central, há o item "*Routes*", que é usado para armazenar os nomes das rotas, como por exemplo "ROTA 01 A".

Figura 11 – Nomes das rotas no *Firestore*



Fonte: (Bizerril, F. E. A., 2024)

Para situações em que não há conexão com a *internet* (*offline*), todos os horários e rotas devem estar disponíveis para o usuário neste modo sem necessidade de intervenção. Ou seja, após a primeira sincronização *online*, todos os dados ficam salvos localmente para serem consultados quando não houver conexão. Além disso, sempre que as informações forem atualizadas em futuras sincronizações, os novos dados devem substituir os antigos e uma notificação *push* deve ser enviada aos usuários.

Para o recebimento dos dados do *Cloud Firestore*, foi utilizado um padrão bastante conhecido chamado de *Model*. Através do seu uso, pode-se adicionar e manipular neste todas as informações e particularidades dos dados recebidos do servidor *web* (*Cloud Firestore*) e criar variáveis mais fáceis de utilizar durante o desenvolvimento. Por exemplo, todos os horários de saída e de chegada de uma rota foram ajustados em um *model*, e foram enviados para diversas camadas mais internas da aplicação. Dados como, por exemplo, data e hora no formato UTC (Tempo Universal Coordenado) `"2024-02-10 02:11:31.947Z"` são convertidos facilmente para o formato `DateTime` no Dart, facilitando muito a manipulação de informações, e o mesmo vale para qualquer outro tipo de dado, como `string, int, double, booleano` e etc.

Para salvamento dos dados locais (*cache*), foi utilizado um pacote externo chamado `get_storage` na versão 2.1.1. Esse pacote está disponível no repositório oficial de pacotes do *Flutter* (Borges, J., 2024b). Para diminuir o acoplamento de dependências externas, foi criado um *adapter*, que é um padrão comum de desenvolvimento, para isolar todo o uso desse pacote. Com a utilização do *adapter*, a aplicação não possui acesso direto ao `get_storage`, o que facilitará a troca por outra dependência no futuro, caso seja necessário.

Como mostra o padrão *adapter*, ele pode ser utilizado para encaixar bibliotecas ou pacotes externos à aplicação (cliente) sem deixá-las acopladas, ou seja, as regras de negócio e as demais camadas do cliente não estão ligadas diretamente com essa biblioteca ou pacote. A vantagem dessa abordagem se dá quando precisamos, durante as manutenções futuras, trocar essas dependências por qualquer outra. Portanto, o adaptador é um intermediador que recebe solicitações do cliente e converte os dados da dependência para um formato já esperado. Se houver mudança da dependência, altera-se apenas a maneira como os dados são manipulados dentro do adaptador, para que a resposta seja a mesma (Dong; Zhao; Peng, 2007).

Observa-se na Figura 12 que há um *adapter* chamado `BoxStorageAdapter`. Este recebe em seu construtor uma instância de um pacote externo chamado *GetStorage*. Pode-se observar também que há uma interface chamada `LocalStorage`. Com isso, toda a aplicação utilizará a interface como meio de acesso ao *adapter* e consequentemente ao pacote, e assim somente o *adapter* terá contato direto a dependências externas. Dessa forma, em caso de necessidade de alteração do *GetStorage* no futuro, será necessário alterar somente o adaptador, sem impactar o restante da aplicação.

Figura 12 – Padrão Adapter para o Box Storage, como exemplo

```dart
class BoxStorageAdapter implements LocalStorage {
  BoxStorageAdapter({required this.box});

  final GetStorage box;

  @override
  Future<void> saveData<T>({required String key, required T value}) async {
    await box.write(key, value);
  }

  @override
  T? fetchData<T>({required String key}) {
    return box.read<T?>(key);
  }
```

Fonte: (Bizerril, F. E. A., 2024)

Figura 13 – Simulação de notificação do *Cloud Messaging*

Fonte: *Firebase Cloud Messaging*

4.2.3 *CloudMessaging*

O FCM (*Firebase Cloud Messaging*) é um serviço em nuvem gratuito, focado na criação, gestão e envio de notificações remotas (*push notification*) para os dispositivos móveis. Através do mesmo é possível enviar informações como título, mensagem e imagem para um grupo ou usuários específicos, como explica a Figura 13. Um outro ponto de destaque é a possibilidade de segmentar o envio de mensagens por público-alvo, palavras-chave e por sistema operacional. O serviço consiste no envio de mensagens rápidas e leves que aparecem na barra de notificação dos dispositivos e possuem *design* customizável conforme a plataforma que esteja recebendo.

Há diversas possibilidades de uso desse tipo de tecnologia, pois esta resolve diversos

problemas dos aplicativos. Por exemplo, em aplicativos de vendas, é possível notificar sempre que uma venda *online* tenha sido realizada ou cancelada, demonstrar problemas ocorridos ou alertar sobre metas a serem atingidas. Plataformas de *streaming* utilizam bastante esse recurso para anunciar novos títulos, promoções e novidades de seus serviços. No entanto, se a utilização for muito intensa e os usuários não entenderem a sua real necessidade, esse recurso pode causar o bloqueio de todas as notificações ou até mesmo desinstalação do aplicativo por parte do usuário. É muito comum encontrar usuários que nunca permitem que aplicativos notifiquem, independentemente da necessidade ou importância que isso tenha para a aplicação.

Entendendo todos os benefícios e cuidados com a utilização das *push notifications*, o aplicativo Intercampi se propõe a utilizá-las apenas para anunciar que novos horários e rotas foram atualizados e possibilitar que os usuários cliquem e sejam direcionados automaticamente para o aplicativo, permitindo a sincronização dos novos dados. Outros casos de uso no aplicativo, mas bastante raros, servem para orientação quanto ao uso do aplicativo em geral ou para informar sobre quaisquer impossibilidades e erros ocorridos.

4.3 PUBLICAÇÃO E OPERACIONALIZAÇÃO DO APLICATIVO

Diferentemente de diversos outros dispositivos e sistemas operacionais, os celulares modernos e controlados por toque nasceram com lojas para publicação, gestão, comércio e uso de aplicativos para os mais diversos fins, como também para os mais diversos públicos e nacionalidades. Além disso, o monitoramento detalhado de *downloads*, notas de *feedback* e *log* de erros ocorridos torna a publicação de aplicativos nestas lojas um requisito de extrema importância para adesão e retenção de novos usuários. Devido a essas questões, o aplicativo Intercampi também foi publicado via loja de aplicativos, a fim de alcançar um público mais amplo e garantir sua adoção e uso contínuo pelos estudantes.

4.3.1 A loja de apps para publicação

O funcionamento básico da loja do *Android*, chamada de *Google Play*, e do iOS, chamada de *App Store*, é praticamente o mesmo. Usuários podem encontrar todos os apps disponíveis que são compatíveis com o seu dispositivo, enquanto os incompatíveis não são vistos. É possível, não só baixar o aplicativo pela primeira vez, como também atualizar para versões mais modernas e realizar compras dentro dos apps. Além disso, outros pontos muito importantes são a possibilidade de os usuários darem *feedbacks* sobre seu uso e experiências, e de escolherem entre diversas opções para realizar uma mesma atividade.

O aplicativo Intercampi foi inicialmente disponibilizado apenas na loja de aplicativos do *Android*, a *Google Play*. Isso se deve ao fato de que os custos para manter aplicativos na Apple Store são muito elevados e periódicos, girando em torno de $100 (cem dólares) por ano, enquanto no *Google Play,* o pagamento único de $25 (vinte e cinco dólares). Como

este projeto não visa lucro e não possui nenhum meio de monetização, utilizar serviços de baixo custo é fundamental. Outro fator de grande importância, é a quantidade muito expressiva de usuários que utilizam *Android* em relação a iOS. Apesar do aplicativo ser multiplataforma e estar pronto para ser publicado na plataforma da *Apple*, isso não será possível devido a esses problemas.

Portanto, neste capítulo serão fornecidos detalhes específicos apenas para a publicação no *Google Play*, embora sejam abordados apenas dados comuns, que também são requisitos na *Apple*. O aplicativo Intercampi pode ser acessado no *Google Play* para *Android* em: https://play.google.com/store/apps/details?id=com.edesonabizerril.newintercampi.

4.3.2 Requisitos para publicação na loja

Todos os aplicativos publicados em qualquer loja de aplicativos passam por rigorosos procedimentos para identificação de código malicioso, uso adequado de anúncios, uso correto de permissões, definição dos objetivos para qual aplicativo foi gerado, dados estatísticos, como também para a reconhecimento de erros e etc. Tudo isso é feito para entregar aplicativos com a melhor qualidade possível para todos os usuários. Desta forma, é muito comum que sejam realizadas diversas alterações no projeto para que este se adeque a todos os requisitos e normas legais das plataformas.

Para que o processo de submissão do aplicativo ocorra, é necessário que ele seja compilado em formato de código binário do tipo APK (*Android Package*) ou AAB (*Android App Bundle*). Esses formatos, principalmente o AAB, garantem alta performance para execução nos mais diversos dispositivos compatíveis. A seleção de como e quando o aplicativo é disponibilizado para os usuários pode ser configurada com muita precisão nas lojas de aplicativos. Dados como quantidade de usuários, desinstalações, locais de uso, retenção e diversas outras informações podem ser analisados, o que fornece uma ampla gama de informações para a tomada de decisões.

Nas lojas de aplicativos, é possível publicar de diversas maneiras. A primeira delas é chamada de "Teste Interno", que serve para disponibilizar o aplicativo apenas para o time de desenvolvimento, a fim de realizar testes. Quando o objetivo é disponibilizar para um grupo maior, pode-se utilizar o "Teste Fechado", mas somente pessoas cadastradas podem ter acesso. Porém, quando o projeto está mais estabilizado, o "Teste Aberto" pode ser utilizado por até 1 mil pessoas com dispositivos compatíveis. Contudo, quando o projeto está estável, a versão de "Produção" é a mais correta e não possui limites de usuários ou restrições por padrão. Em resumo, os dois primeiros modos são para uso restrito de desenvolvimento e não passam por nenhuma revisão da loja, enquanto os demais são para o público em geral e sofrem rigorosas revisões (Google, 2024d).

Com o objetivo de identificar o aplicativo para o público, alguns dados básicos são requeridos no *Google Play*, na sessão de "Recursos de Página de Detalhes" (Figura 14), dados como nome do aplicativo, descrição curta e completa são fundamentais, pois

através destes o aplicativo pode ser encontrado de forma orgânica na loja, ou seja, um usuário pode pesquisar por algum termo e encontrar o aplicativo nos primeiros resultados. A importância de cada parâmetro para a busca é dada na mesma ordem de aparição, sendo o nome do aplicativo o mais importante, seguido pela breve descrição e, por último, a descrição completa. A descrição é um fator muito relevante, pois define o contexto do aplicativo para a loja, permitindo que o usuário entenda o que o aplicativo faz, além de permitir uma melhor indexação e otimização das buscas.

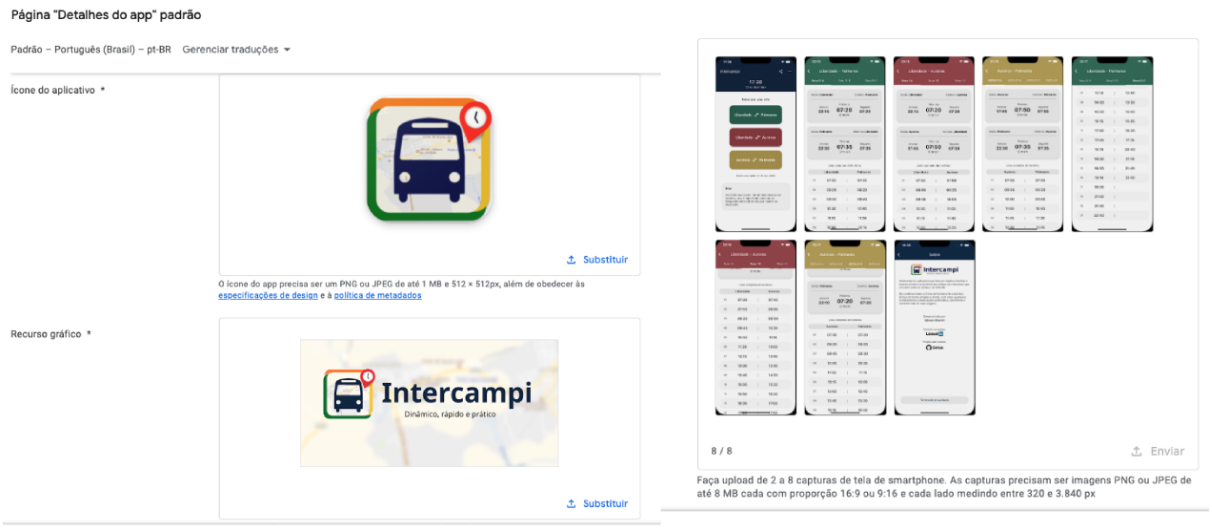
Figura 14 – Recursos da página de detalhes do Play Console

Fonte: *Google Play* Console

Seguindo os requisitos de importância, também são requeridos alguns recursos gráficos na sessão "Elementos Gráficos" do *Google Play* (Figura 15). O ícone é o mais importante de todos, pois através dele, a grande maioria dos usuários se interessam pelo aplicativo. Quanto mais atraente e representativo do que o público-alvo quer ver, melhor o ícone. Por isso, foi criado um ícone por meio do qual se busca representar bem o aplicativo e a Universidade. O *banner*, que é a imagem horizontal, aparece em situações mais específicas e não é muito comum de ver em aplicativos, geralmente sendo mais utilizado em sites. Para o aplicativo, foi criado um *banner* com a fonte característica e um fundo da localização da Unilab no *Google Maps*. Já as capturas de tela são extremamente úteis para fazer as pessoas realmente baixarem o aplicativo, pois por meio destas, é possível ter uma prévia do que é oferecido. As capturas de tela do aplicativo demonstram todas as telas e suas variações.

A grande maioria dos aplicativos possui um fluxo de publicação e manutenção minimamente bem definido, pelo próprio uso comum dos aplicativos. Geralmente, os aplicativos são publicados inicialmente com recursos simples, mas que definem bem o que o aplicativo pretende entregar. Com o passar do tempo, os usuários iniciam um processo de avaliação nas lojas, o que pode gerar alguns relatórios de erros. Por meio desses dados,

Figura 15 – Elementos gráficos requeridos pelo *Google Play*



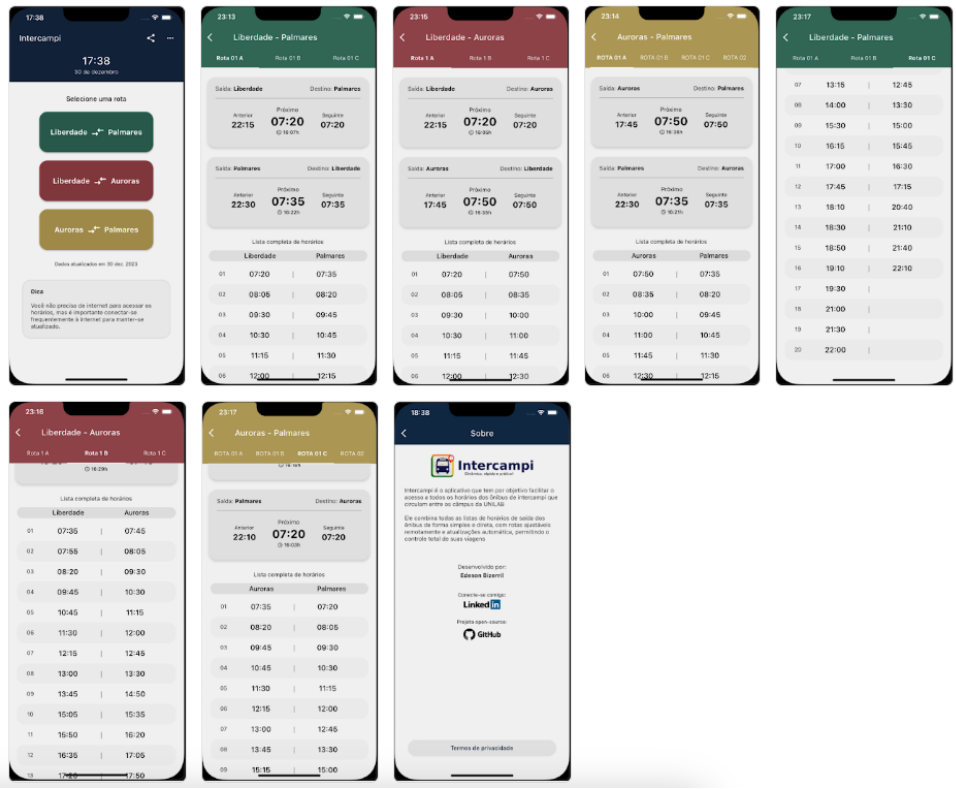
Fonte: Google Play Console

é possível identificar os pontos de correções e melhorias e subir novas versões sempre que necessário. Pequenos incrementos são realizados a cada atualização, e isto, com o tempo, molda uma aplicação robusta e mais adequada para as necessidades de todos os usuários.

Os aplicativos de qualquer loja são publicados utilizando o versionamento semântico, que é um padrão numérico composto por três números separados por pontos, seguindo a estrutura *'major.minor.patch'*. O `Major` é o primeiro número que, define a versão principal do aplicativo. Geralmente, uma mudança na versão principal indica grandes atualizações ou mudanças significativas, como a adição de novos recursos, alterações na estrutura fundamental do aplicativo ou incompatibilidade com versões anteriores. O `Minor`, segundo número, representa a versão secundária do aplicativo. Uma mudança neste número indica atualizações menores, como a inclusão de novos recursos ou melhorias sem que haja grandes alterações na funcionalidade principal. O `Patch`, o terceiro número, refere-se a pequenas correções de erros ou atualizações de segurança que não adicionam novos recursos ou funcionalidades significativas ao aplicativo.

Todas as vezes que uma nova atualização é publicada no aplicativo Intercampi, um novo número de versão é atribuído e acrescido em no mínimo `+1`, sempre um acréscimo em relação à versão anterior. Ou seja, no primeiro momento pode ser publicado uma versão com qualquer valor, por exemplo, a versão `1.0.0`. E com o passar do tempo, novas versões são lançadas e adicionadas com valores maiores, como `1.0.1`. Este acréscimo permite à loja entender que uma nova versão está disponível para o aplicativo, bem como permite ao dispositivo atualizar automaticamente se as configurações de atualização automática estiverem habilitadas. Caso contrário, os usuários podem ser notificados sobre a disponibilidade de uma atualização e têm a opção de baixá-la manualmente.

Figura 16 – Todas as novas tela de aplicativo Intercampi



Fonte: (Bizerril, F. E. A., 2024)

4.4 RELEVÂNCIA DO APLICATIVO PARA A COMUNIDADE ACADÊMICA DA UNILAB

4.4.1 O uso do aplicativo no dia a dia acadêmico

A UNILAB possui quatro campi universitários, sendo um localizado no estado da Bahia (campus dos Malês) e três no estado do Ceará (campi da Liberdade, das Auroras e dos Palmares). Os campi cearenses possuem em 2024, 10 rotas que circulam entre todos os campus a cada 20 minutos. No entanto, a quantidade de rotas muda constantemente, o que torna o gerenciamento e a manutenção das atualizações uma tarefa trabalhosa. O aplicativo Intercampi facilita o acesso a essas informações, permitindo que os estudantes consultem os horários e rotas de forma rápida e prática no seu dia a dia acadêmico.

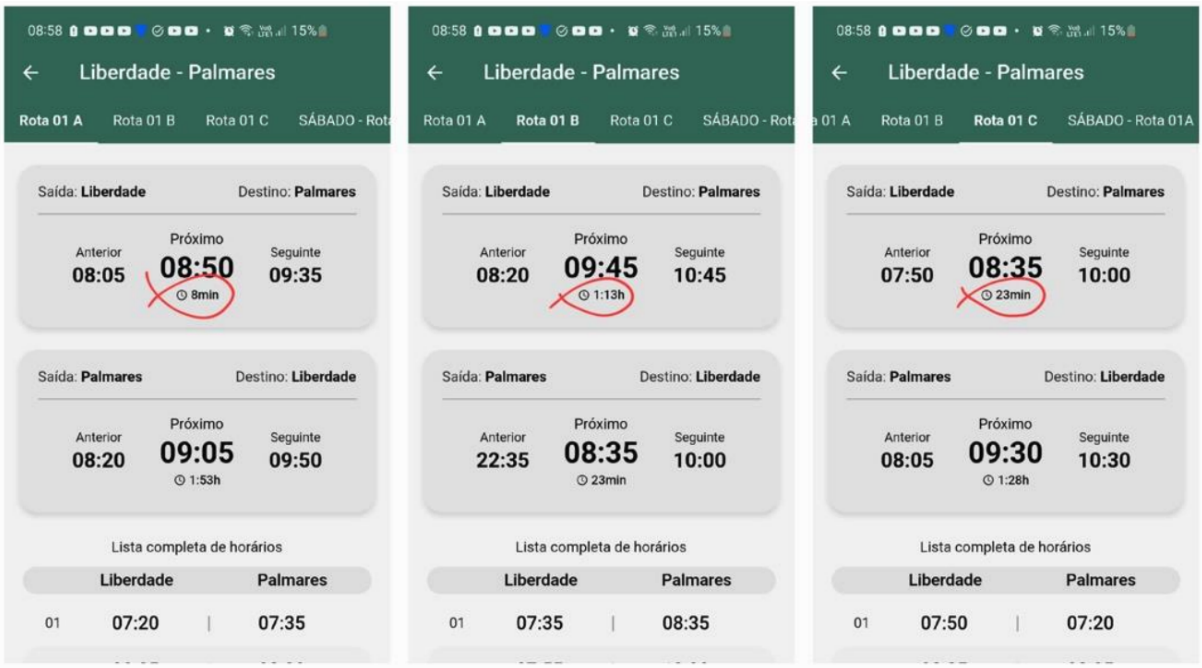
O departamento de transporte universitário disponibiliza diversas listas de horários dos ônibus para uso durante os períodos letivos e não letivos. Eventualmente, ocorrem mudanças e o departamento de comunicação envia novas listas com os horários atualizados por e-mail e no site oficial da universidade. Professores, alunos, técnicos, servidores e demais pessoas podem ter acesso às listas de horários através desses dois meios, bem como em papéis impressos disponíveis em diversos locais. Apesar dessa divulgação, o acesso às informações não é prático, e a maioria das pessoas permanece desatualizada ou sem nenhuma informação sobre os horários.

Sem acesso às listas de horários, diversos alunos e profissionais da universidade gastam muito tempo esperando em pontos de ônibus e paradas, pois não sabem exatamente os horários dos transportes. Essa falta de informação pode levar a atrasos, perda de compromissos e, em alguns casos, até mesmo a assaltos enquanto aguardam o transporte. Logo, ter acesso aos horários e rotas via um aplicativo móvel permitiria um acesso rápido e atualizado a todas as informações com sincronização automática, já que a maioria das pessoas mantém seus telefones sempre à mão. Isso possibilitaria um melhor planejamento das viagens e evitaria transtornos desnecessários.

Agora, com o uso do aplicativo Intercampi, ficou muito fácil manter-se organizado e atualizado com as mais recentes listas de horários, pois é possível acessar qualquer Linha e escolher a rota que sairá o mais próximo do horário desejado. Assim, é simples organizar a saída com antecipação, sem a necessidade de passar muito tempo esperando o ônibus em locais públicos. Essas informações, apesar de simples, são poderosas, pois permitem um melhor controle do tempo e tranquilidade para resolver problemas no dia a dia acadêmico.

Por exemplo, supondo que um estudante precisa sair de casa, pegar um ônibus no campus da Liberdade e chegar às 10h da manhã para a realização de uma prova no campus Palmares. Com o aplicativo em mãos, seria possível acessar o primeiro botão do aplicativo, de cor verde, onde está escrito “Liberdade – Palmares”, e ter acesso a todas as informações da Linha. Uma vez nesta tela, é possível ver todas as rotas disponíveis, na parte superior do aplicativo, rolar para a direita, e saber qual o ônibus mais próximo de sair dentre todas elas. Um recurso muito útil é observar o indicador de tempo restante em minutos para a chegada do próximo ônibus a sair (Figura 17).

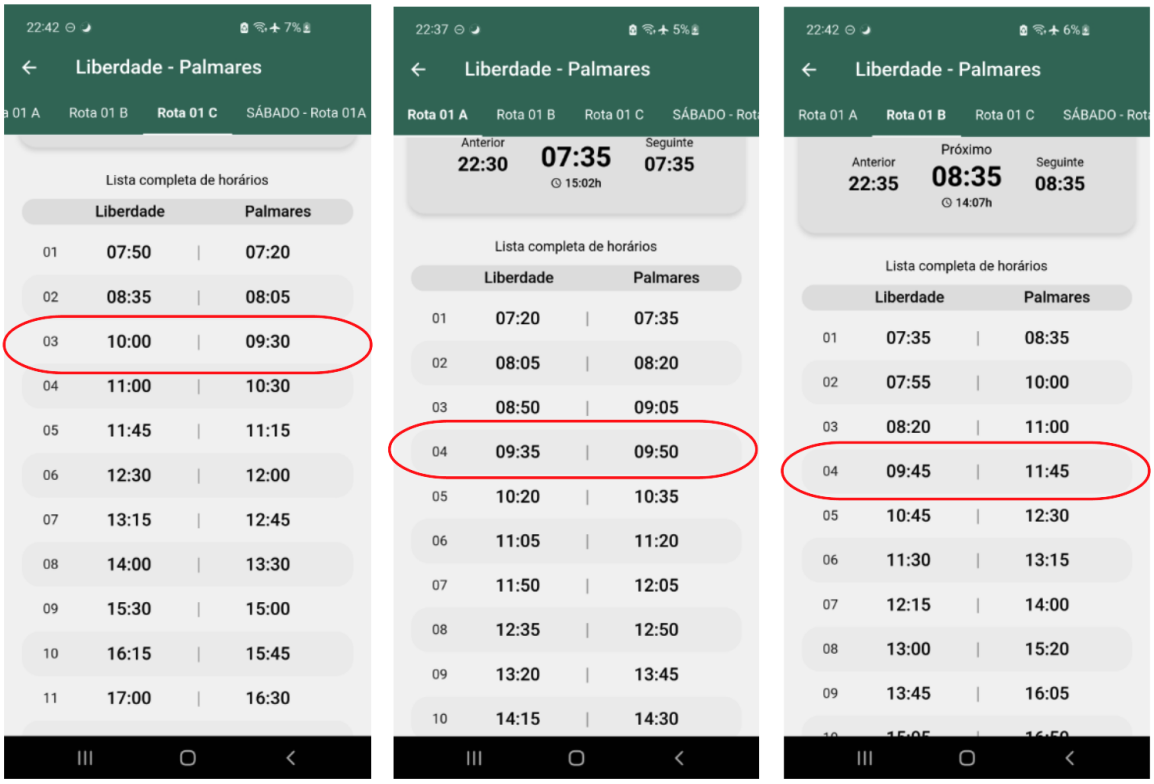
Figura 17 – Demonstração do tempo restante para a próxima saída das três rotas



Fonte: (Bizerril, F. E. A., 2024)

A seleção do melhor horário para a saída neste exemplo (Figura 18), pode ser encontrada por meio da comparação entre todos os horários restantes para cada saída. Como dito anteriormente, ao rolar para a direita, é possível visualizar essas informações rapidamente. Contudo, se o horário requerido não é o próximo ônibus, ao rolar tela de qualquer rota para baixo, é possível visualizar todos os horários e identificar qual será o melhor para sair e chegar a tempo. Similarmente, fazer o planejamento para retornar para casa segue o mesmo princípio básico de análise dos horários, contudo, desta vez, analisando a saída do Palmares.

Figura 18 – Seleção do mais próximo horário de saída às 10h da manhã nas três rotas



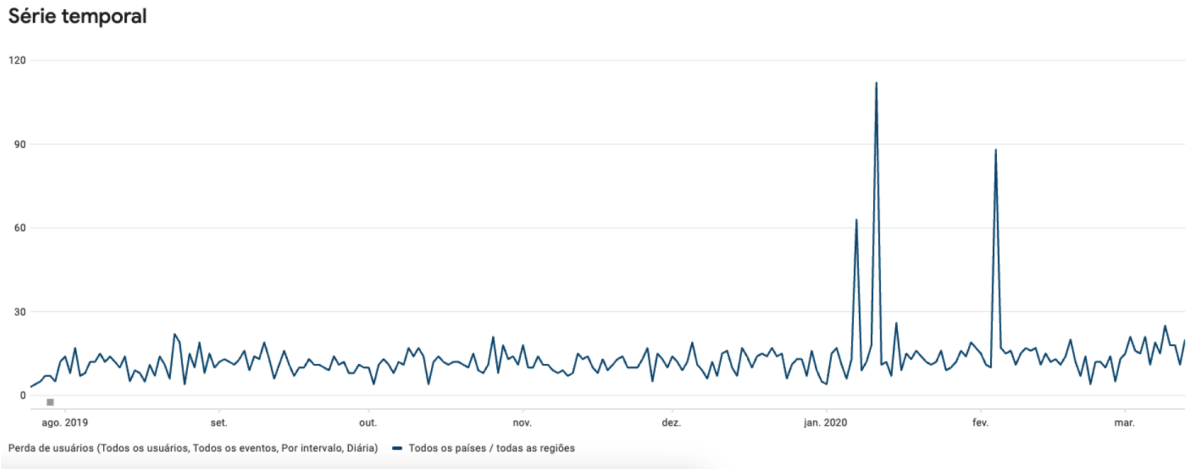
Fonte: (Bizerril, F. E. A., 2024)

4.4.2 Estatísticas de uso e comportamento dos usuários

Para compreender melhor o impacto do aplicativo no meio acadêmico, é essencial analisar as estatísticas de uso e o comportamento dos usuários desde seu lançamento, em 15 de setembro de 2018, até 2019. As estatísticas de uso fornecidas pelo *Google Play Console* disponíveis em seu painel de *analytics*, oferecem *insights* valiosos sobre a adoção e o crescimento do aplicativo. Com essas métricas é possível analisar o número de *downloads*, a retenção de usuários, as avaliações e comentários, bem como a distribuição demográfica dos usuários. Além disso, essas estatísticas permitem identificar padrões de uso, como os períodos de maior atividade no aplicativo e o engajamento.

É claro que a quantidade de *downloads* de um aplicativo não significa a quantidade de usuários ativos. Mesmo que um aplicativo tenha uma grande quantidade de *downloads*, pode haver uma baixa retenção de usuários. Contudo, por conta do aplicativo Intercampi ser um aplicativo pequeno e com um público-alvo bastante segmentado, é fácil analisar de perto o comportamento dos usuários e relacionar o número de *downloads* com o comportamento deles. Observando a série temporal na Figura 19, é possível notar a perda de usuários ao longo de todo o período de vida do aplicativo, o que pode indicar que muitos usuários instalam, usam o aplicativo, e desinstalam em seguida, havendo uma alta frequência média de desinstalação.

Figura 19 – Série temporal da perda de usuários no período total

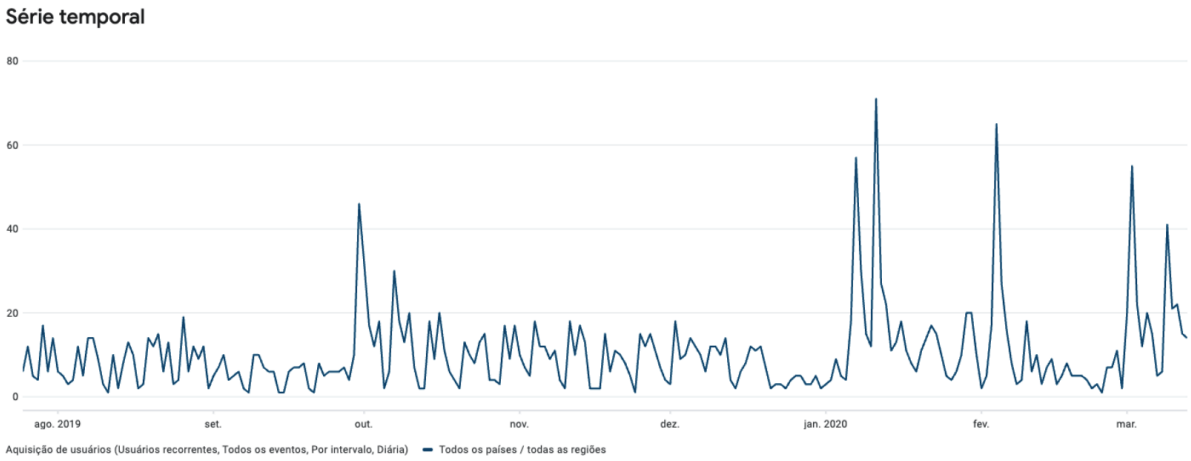


Fonte: Google Play Console

A hipótese anterior torna-se bastante plausível quando comparada à série anterior de desinstalação com a quantidade de downloads de usuários recorrentes Figura 20, ou seja, usuários que voltaram a instalar o aplicativo após ao menos uma desinstalação. Observa-se que o comportamento de reinstalação se assemelha bastante ao de desinstalação. A média é muito próxima e até mesmo os pontos de picos são semelhantes. Desconsiderando os grandes picos de desinstalação de 2020, os números de reinstalação são melhores. Isso indica que grande parte dos usuários de fato reinstalam o aplicativo quando precisam e o removem em seguida. Com essa base, as análises posteriores utilizarão o número de downloads como parâmetro para mostrar o engajamento dos usuários ao longo do tempo.

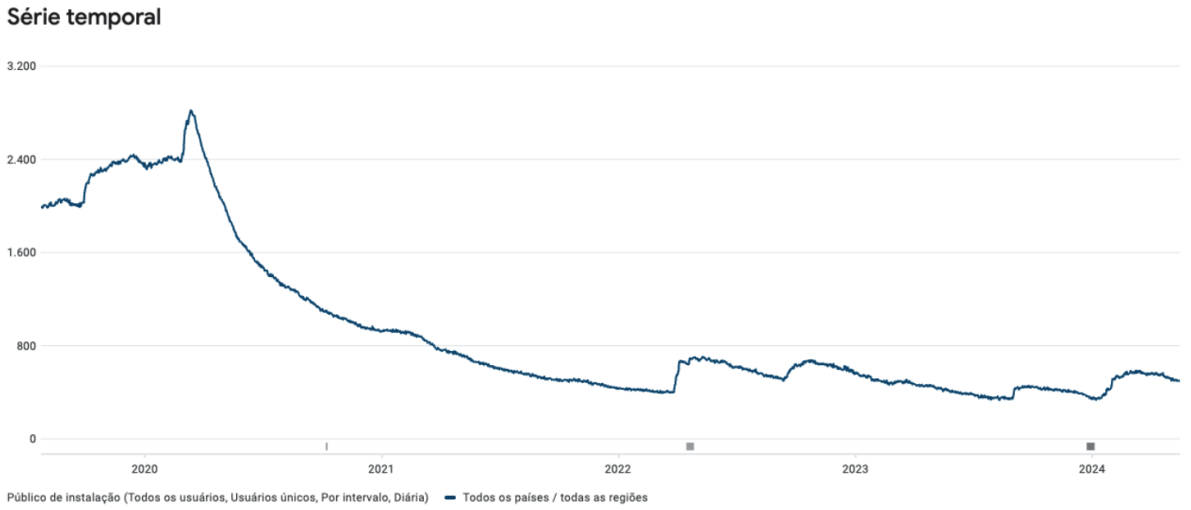
A Figura 21 apresenta a quantidade de downloads durante todo o período de vida do aplicativo, entre 2019 até o momento em 2024. A curva presente revela tendências significativas que refletem o engajamento e a retenção dos usuários. Entre o início da série e meados de fevereiro de 2020, houve um aumento gradual e orgânico no número de downloads, devido à divulgação interna do aplicativo na Universidade, alcançando um pico de aproximadamente 3500 ativos. Pode-se observar também que, logo após esse período, houve uma queda contínua e gradual bastante significativa, devido ao encerramento das atividades acadêmicas durante a pandemia de COVID-19 (OPAS/OMS, 2024). Contudo,

Figura 20 – Série temporal da aquisição de usuários recorrentes durante todo o período



Fonte: Google Play Console

Figura 21 – Quantidade downloads desde o lançamento



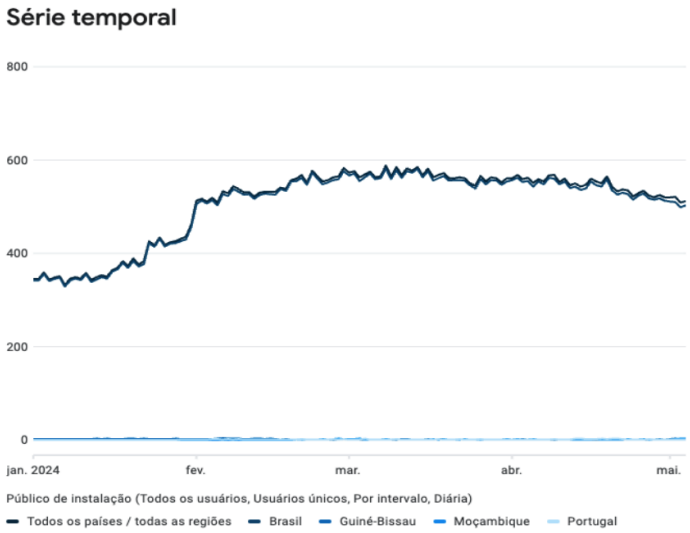
Fonte: Google Play Console

com a volta às aulas presenciais, espera-se que esses números se recuperem organicamente, com o tempo e de forma gradual, a medida que os usuários o compartilham entre si e em grupos de alunos.

Um outro problema, que pode ser visto na Figura 22, e que atrapalhou o crescimento do aplicativo foi a dificuldade de manutenção utilizando a tecnologia por meio da qual este foi construído inicialmente, com Java no *Android* nativo. Com o passar dos anos, o aplicativo ficou defasado e a *Google Play* exigiu algumas alterações e atualizações. Contudo, devido à dificuldade no desenvolvimento e manutenção, novas submissões não foram realizadas e o aplicativo foi banido da loja. Este problema também explica a significativa redução no número de downloads observada nas séries temporais anteriores.

Para melhor entender o comportamento comum dos usuários ao longo do tempo, deve-se analisar algumas partes desta linha temporal da Figura 22. Como um exemplo específico, no início da série, apresenta-se a quantidade de usuários e um crescimento

Figura 22 – Série temporal de instalações entre janeiro e maio de 2024



Fonte: Google Play Console

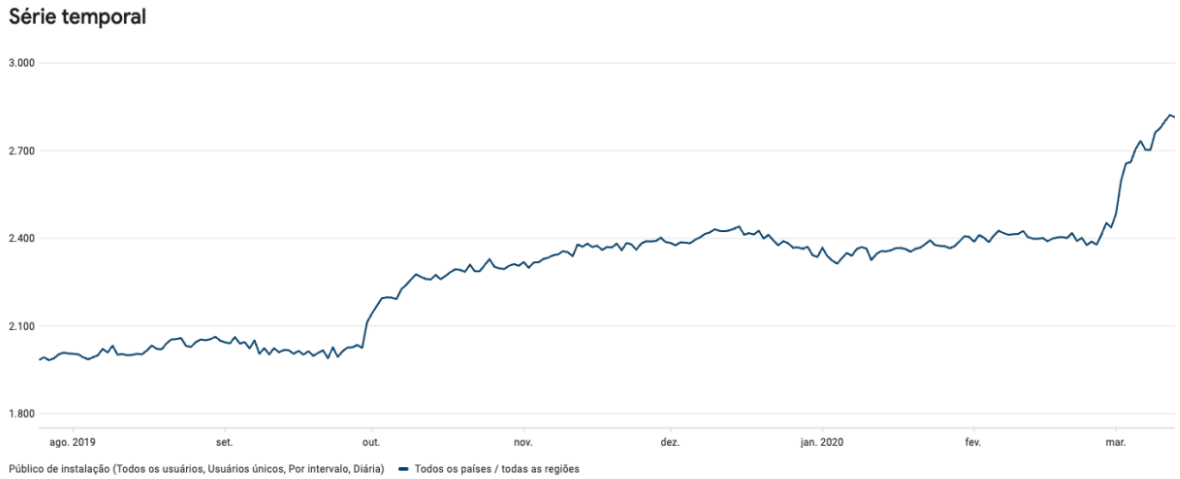
bastante elevado entre fevereiro e março. Logo após, estabiliza-se até meados de abril e cai próximo a maio. Isso é um exemplo claro de comportamento comum e esperado dos usuários no aplicativo, pois representa o início, meio e fim do período letivo. No início, a busca pela ferramenta aumenta consideravelmente devido ao desconhecimento das novas listas de horários. A partir do segundo mês, se estabiliza, pois algumas pessoas aprendem o momento de saída dos principais ônibus que necessitam no dia a dia e acabam desinstalando o aplicativo. Além disso, a chegada do período de férias influencia ainda mais na decisão de desinstalar.

Para fazer uma estimativa sobre o comportamento comum dos usuários ao longo do tempo, deve-se analisar algumas partes desta linha temporal da Figura 22. Por exemplo, no início da série, observa-se uma quantidade significativa de usuários e um crescimento acentuado entre fevereiro e março. Após esse período, há uma estabilização até meados de abril, seguida por uma queda próxima a maio. Embora não seja possível definir precisamente um padrão comum de comportamento dos usuários sem investigações diretas com os usuários, acredita-se que os dados demostram o interesse pelo aplicativo durante o início, meio e fim do período letivo. O aumento inicial na utilização do aplicativo pode ser atribuído ao desconhecimento das novas listas de horários, enquanto a estabilização e a subsequente desinstalação podem refletir a adaptação dos usuários ao serviço e a influência do período de férias.

Da mesma maneira, pode-se analisar e perceber nos demais meses correspondentes aos períodos letivos, o mesmo comportamento de subida do número de usuários, uma estabilização temporária e uma queda ao final. Na Figura 23, é possível ver três períodos: o primeiro entre agosto e outubro de 2019, o segundo entre outubro de 2019 a janeiro de 2020 e o último de outubro a início de abril de 2020. Nota-se que nos três é possível ver

o mesmo padrão de crescimento orgânico, desconsiderando que no último há um grande pico devido à Pandemia (OPAS/OMS, 2024).

Figura 23 – Série temporal de instalações de todo o período de vida do aplicativo



Fonte: Google Play Console

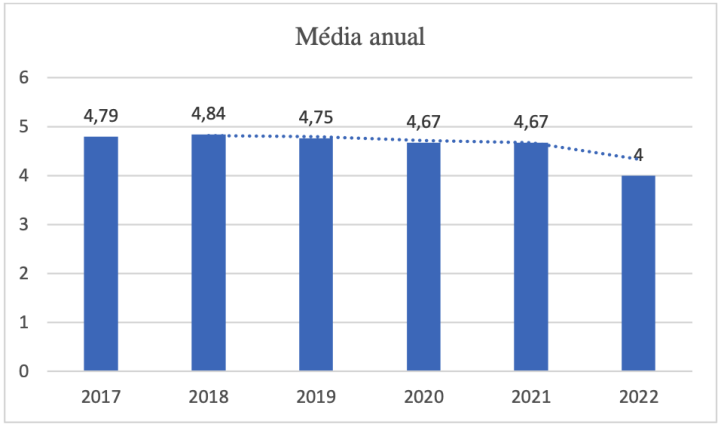
4.4.3 Feedbacks e notas: distribuição e análise de sentimentos com inteligência artificial

As lojas de aplicativos, tais como *Google Play* e *Apple Store*, permitem aos usuários expressar suas opiniões sobre qualquer aplicativo por meio de *feedbacks* na página do aplicativo, tornando evidentes as qualidades e defeitos do produto. Para isso, os usuários podem atribuir notas de 1 a 5 (inclusive) e fornecer um comentário detalhando suas opiniões. Embora a qualificação por nota seja obrigatória, o texto escrito não o é, fazendo com que grande parte dos *feedbacks* contenha apenas notas. Portanto, analisar todos esses dados é um caminho lógico e muito importante para avaliar a satisfação e o impacto causado pelo aplicativo em toda a comunidade acadêmica.

O período de todos os dados inicia-se desde a primeira avaliação em dezembro de 2017 até a última registrada em março de 2022. As notas atribuídas possuem uma **média geral de 4,81**, mediana 5,0 e moda 5,0. Estes dados quando observados e comparados a média anual entre todos os anos, observa-se valores bem próximos, como pode ser visto na Figura 24. Nota-se que a menor nota média foi de 4,67 de 5,0 até 2021, com o valor mais baixo em 2022 igual a 4,0. Nota-se na Figura 24, que, em geral, há uma constância em notas altas, acima de 4,0.

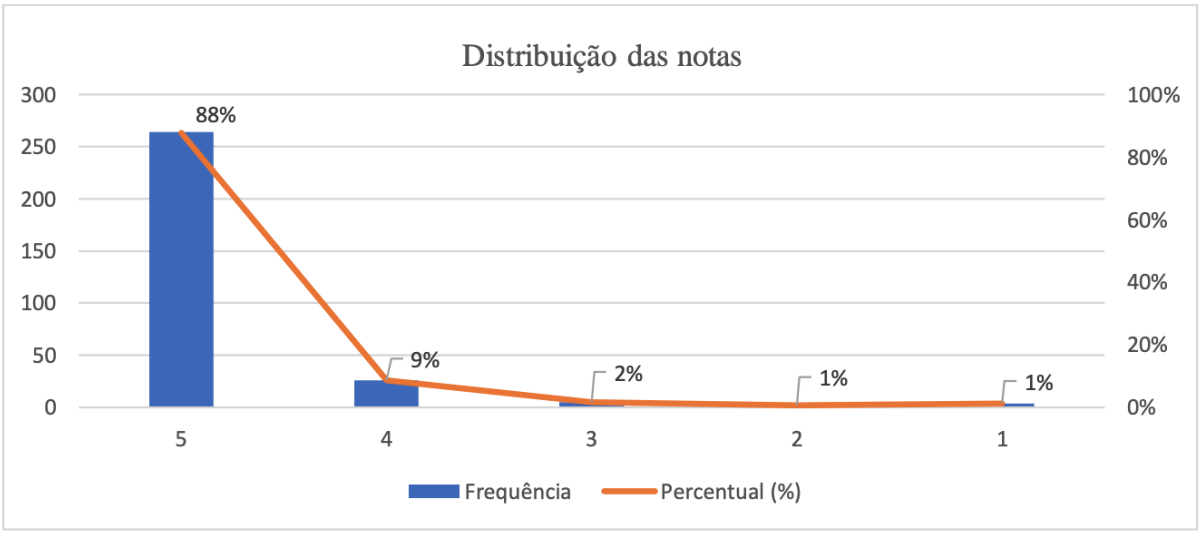
Na Figura 25 também é possível observar a satisfação dos usuários através da distribuição das notas, levando em consideração a porcentagem de cada nota acumulada. Dentre todas as 301 avaliações, aproximadamente 265 atribuíram nota 5 ao aplicativo, correspondendo a 88% dos usuários, enquanto apenas 9% deram nota 4, e todas as demais

Figura 24 – Distribuição das médias anuais



Fonte: (Bizerril, F. E. A., 2024)

Figura 25 – Distribuição das notas por frequência e percentual



Fonte: (Bizerril, F. E. A., 2024)

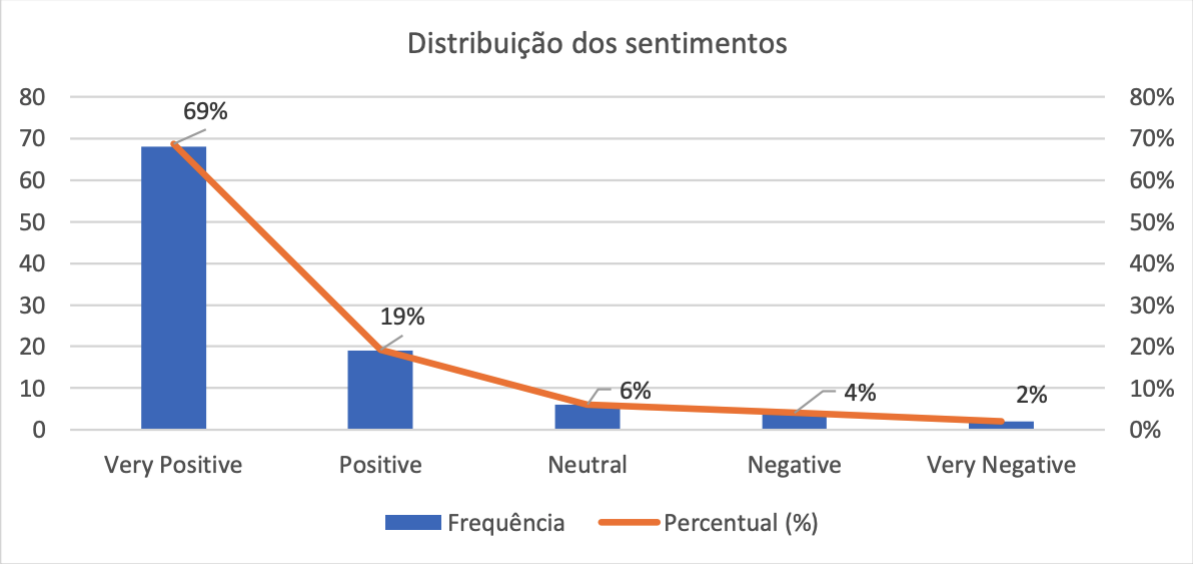
notas inferiores não passaram de 2%. Esses resultados demonstram claramente um alto grau de satisfação por parte dos usuários com as soluções entregues pelo aplicativo.

Todos os comentários fornecidos pelos usuários em seus *feedbacks* demonstram suas opiniões acerca de sua experiência e do valor percebido no aplicativo. Assim, uma análise textual pode revelar os sentimentos da comunidade ao utilizar a aplicação. Para isso, a HFT (*Hugging Face Transformers*) foi utilizada para analisar automaticamente todos os textos e definir os sentimentos demonstrados pelos usuários. Há diversos modelos disponíveis nessa biblioteca, porém, para analisar textos em português, foi necessário um modelo multilíngue.

A HFT é uma biblioteca de inteligência artificial de código aberto para PLN (Processamento de Linguagem Natural) que permite a compreensão e a geração de texto de maneira sofisticada. Essa biblioteca disponibiliza uma ampla gama de modelos pré-treinados para tarefas de PLN, como tradução, resumo, resposta a perguntas e, com

particular aplicação neste trabalho, análise de sentimentos. Esses modelos são baseados em arquiteturas avançadas, como *BERT, GPT, RoBERTa*, entre outras, que têm se mostrado extremamente eficazes em capturar nuances linguísticas e contextuais em textos (Wolf *et al.*, 2020b).

Figura 26 – Distribuição dos sentimentos



Fonte: (Bizerril, F. E. A., 2024)

Um dos modelos específicos disponibilizados pela Hugging Face e que foi utilizado nesse trabalho é o `nlptown/bert-base-multilingual-uncased-sentiment`. Este modelo é especialmente projetado para análise de sentimentos, sendo treinado em um grande conjunto de dados multilíngue e capaz de operar em diversos idiomas. O Hugging Face é uma variante do BERT, uma arquitetura que revolucionou o campo do PLN com sua capacidade de processar o contexto bidirecionalmente, ou seja, considerar tanto as palavras anteriores quanto as posteriores para interpretar o significado de um termo em uma frase (Wolf *et al.*, 2020b).

Após o tratamento dos dados coletados do *Google Play*, correspondente a todo o período de vida do aplicativo (Anexo A), foi aplicado o modelo Hugging Face com uso do Python (Anexo B). Assim, todas as opiniões foram classificadas automaticamente em **5 maneiras**: `muito negativas`, `negativas`, `neutras`, `positivas` e `muito positivas`. Desta forma, foram processados 99 textos de opiniões dos usuários, onde 69% classificaram o aplicativo como `Very Positive`, ou seja, muito positivo, e 19% como positivo (Anexo C). Em outras palavras, 88% das pessoas que se dispuseram a dar *feedback* acerca do aplicativo aprovaram a solução utilizada.

5 CONSIDERAÇÕES FINAIS

Por fim, este trabalho apresentou uma descrição detalhada do desenvolvimento de uma nova solução em *Flutter* para o aplicativo Intercampi. O objetivo dessa nova solução foi continuar garantindo acesso aos horários e rotas de ônibus em qualquer local e situação de uso, com sincronização *online* automática e acesso a todos os dados *offline*. Foi utilizada a arquitetura do aplicativo baseada no padrão Getx Pattern, por ser bem documentado e utilizado em projetos de diferentes tamanhos, preparando o aplicativo para crescer no futuro. Utilizou-se o *Firebase* como servidor remoto para dados e envio de notificações *push*. Além disso, demonstrou-se na prática a utilização dos conceitos de *Atomic Design*, que divide os componentes em subcamadas, facilitando a manutenção e a expansão do aplicativo.

Com base nos dados da *Google Play*, onde foi publicado, e na análise por Inteligência Artificial dos *feedbacks*, pode-se concluir que o aplicativo Intercampi apresenta uma excelente avaliação e satisfação por parte dos usuários. As notas atribuídas demonstram uma constância em valores altos, acima de 4, com a grande maioria classificando o aplicativo como muito positivo. Essa percepção positiva é reforçada pela análise textual dos comentários, que revelam sentimentos extremamente favoráveis em relação à experiência e ao valor proporcionado pela aplicação. Portanto, os resultados obtidos através da avaliação das notas e da análise de sentimentos dos *feedbacks* comprovam que o aplicativo é uma solução de grande qualidade e impacto positivo para a comunidade acadêmica, atendendo de maneira excelente às necessidades e expectativas dos usuários.

Os próximos passos deste trabalho incluirão a adição de diversas funcionalidades para aprimorar ainda mais a aplicação. Entre elas, estão previstos a implementação de testes unitários, uma interface de gerenciamento de horários, leitura automática de PDF, alarmes, avisos de perigo em locais públicos, e monitoramento dos ônibus por GPS com acompanhamento no aplicativo. Essas melhorias e visam ampliar os recursos disponíveis aos usuários, tornando a aplicação ainda mais completa e atendendo a um espectro ainda mais amplo de necessidades da comunidade acadêmica. Com a incorporação dessas novas funcionalidades em pesquisas posteriores, espera-se que a satisfação e o impacto positivo do aplicativo junto aos usuários sejam ainda mais evidentes em futuras avaliações.

REFERÊNCIAS

ARAÚJO, Matheus Felipe Gonçalves Oliveira de. **Solução de software para sistema de doações**. UFU, 2021.

BORGES, Jonata. **Get - Flutter package**. Disponível em: <https://pub.dev/packages/get>. Acesso em: 18 out. 2024.

BORGES, Jonata. **get_storage - Flutter package**. 2024. Disponível em: <https://pub.dev/packages/get_storage>. Acesso em: 18 out. 2024.

BORGES, Jonathan R. A.; Santos, Paulo C. **Frameworks para desenvolvimento de aplicativos móveis:** mapeamento, análise e comparação de funcionalidades. IFSULDEMINAS, 2019.

DHIMAN, Nidhi; CHOUDHARY, Ankush; CHAUDHARY, Satish. **Review on Comparative Study of Flutter App and Android App.** International Journal for Research in Applied Science and Engineering Technology, 2023.

DONG, Jing; ZHAO, Yajing; PENG, Tu. **Architecture and Design Pattern Discovery Techniques - A Review**. *In*: Software Engineering Research and Practice. [*S.l.*: *s.n.*], 2007.

DOURADO, Luís Antônio da Silva; GUARIENTI, Gracyeli Santos Souza; FERNANDES, Azenaide; OLIVEIRA, Frederico Santos de; SIQUEIRA, Vitor de Morais. **Análise de sentimentos de tweets em português-brasileiro utilizando inteligência artificial.** Editora Científica, 2024.

FIGUEREDO, Matheus de Oliveira Marques. **Desenvolvimento de aplicações móveis: avaliação do esforço necessário para as plataformas nativa e multiplataforma.** IFSULDEMINAS, 2017.

FIREBASE. **FIREBASE - Google's Mobile and Web App Development Platform**. 2024. Disponível em: <https://firebase.google.com/?hl=pt-br>. Acesso em: 23 out. 2024.

FIREBASE. **FIREBASE - Prós e contras explicados**. 2024. Disponível em: <https://blog.back4app.com/pt/firebase-pros-e-contras/>. Acesso em: 21 out. 2024.

FIREBASE. **Firebase Cloud Messaging.** 2024. Disponível em: <https://firebase.google.com/docs/cloud-messaging/?hl=pt-br>. Acesso em: 21 out. 2024.

FLUTTER. **Android - Flutter.** 2024. Disponível em: <https://docs.flutter.dev/deployment/android>. Acesso em: 18 out. 2024.

FLUTTER. **Animations - Flutter.** 2024. Disponível em: <https://docs.flutter.dev/ui/animations>. Acesso em: 04 out. 2024.

FLUTTER. **Flutter architectural overview - Flutter.** 2024. Disponível em: <https://docs.flutter.dev/resources/architectural-overview#flutters-rendering-model>. Acesso em: 02 out. 2024.

FLUTTER. **Flutter performance profiling -** Flutter. Disponível em: <https://docs.flutter.dev/perf/ui-performance>. Acesso em: 12 out. 2024.

FLUTTER. **Layout - Flutter**. Disponível em: <https://docs.flutter.dev/ui/layout>. Acesso em: 02 out. 2024.

FLUTTER. **Using packages | Flutter**. Disponível em: <https://docs.flutter.dev/packages-and-plugins/using-packages>. Acesso em: 04 out. 2024.

FROST, Brad. **Atomic Design Methodology: Atomic Design by Brad Frost**. Disponível em: <https://atomicdesign.bradfrost.com/chapter-2/>. Acesso em: 13 out. 2024.

GONÇALVES, André; LEMOS, Gonçalo; GONÇALVES, Tiago; BELO, Ricardo. **Sistema de Autenticação (FaceShield)**. UAL, 2023.

GOOGLE. **Flutter - Build apps for any screen**. 2024. Disponível em: <https://flutter.dev/>. Acesso em: 10 out. 2024.

GOOGLE. **Introduction - Material Design**. 2024. Disponível em: <https://m1.material.io/material-design/introduction.html>. Acesso em: 16 out. 2024.

GOOGLE. **Material Design**. 2024. Disponível em: <https://m3.material.io/>. Acesso em: 18 out. 2024.

GOOGLE. **Preparar e lançar uma versão - Ajuda do Play Console.** Disponível em: <https://support.google.com/googleplay/android-developer/answer/9859348?hl=pt-BR&ref_topic=7072031&sjid=3179829019654622692-SA>. Acesso em: 19 out. 2024.

HJORT, Elin. **Evaluation of React Native and Flutter for cross-platform mobile application development.** URN, 2019.

HUGGING FACE. **nlptown/bert-base-multilingual-uncased-sentiment · Hugging Face.** 2024. Disponível em: <https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment>. Acesso em: 12 out. 2024

IBM. **O que é desenvolvimento de aplicativos IOS? - IBM**. 2024. Disponível em: <https://www.ibm.com/br-pt/topics/ios-app-development>. Acesso em: 12 out. 2024.

JAIN, Shashank Mohan. **Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems - SpringerLink.** 2024. Disponível em: <https://link.springer.com/book/10.1007/978-1-4842-8844-3>. Acesso em: 13 out. 2024.

JESUS, Elias Lemes de; Ferreira, Daniela Carvalho Monteiro. APLICANDO PLN PARA ANÁLISE DE SENTIMENTOS DO TWITTER. **PUC MINAS,** 2020.

EL-KASSAS, Wafaa S.; ABDULLAH, Bassem A.; YOUSEF, Ahmed H.;WAHBA, Ayman M. Taxonomy of Cross-Platform Mobile Applications Development Approaches. **Ain Shams Engineering Journal,** 2017. ISSN 2090-4479.

LABORNEL, Lucas B.; OLIVEIRA, Caique R. Avaliacão de desempenho de aplicativos nas tecnologias Android, Flutter e React Native, com o uso de metricas diretas e da analise de sentimentos. **PUC MINAS**, 2024.

MURAKAMI, Kauê. **getx_pattern - uma proposta pra você que utiliza o GetX.** 2024. Disponível em: <https://kauemurakami.github.io/getx_pattern/>. Acesso em: 18 out. 2024.

MURAKAMI, Kauê. Um pouco sobre o GetX Pattern. Disponível em: <https://kauemurakami.medium.com/um-pouco-sobre-o-getx-pattern-efb191187d7>. Acesso em: 23 out. 2024. 2024.

OLIVEIRA, Rafael. **Aplicativo facilita o acesso aos horários de ônibus que circulam entre os campi da Unilab no Ceará - Seção de Portais e Aplicações WEB (SPA/DSI/DTI/Unilab)**. 2024. Disponível em: <https://unilab.edu.br/2017/10/06/aplicativo-facilita-o-acesso-aos-horarios-de-onibus-que-circulam-entre-os-campi-da-unilab-no-ceara/>. Acesso em: 16 out. 2024.

OLIVEIRA JUNIOR, Marcos Antonio de. Desenvolvimento de um aplicativo para a plataforma móvel android para m-commerce de alimentos. **Repositório digital da UFSM**, 2013.

OLIVEIRA NETO, Thiago; CAVICCHIOLI, Maria. Mudanças de percursos e propostas: estudo de caso do transporte coletivo do campus da UFAM em Manaus com o uso do SIG. **Estrabão**, 2024.

OPAS/OMS. **Histórico da pandemia de COVID-19 - OPAS/OMS - Organização Pan-Americana da Saúde**. 2024. Disponível em: <https://www.paho.org/pt/covid19/historico-da-pandemia-covid-19>. Acesso em: 19 out. 2024.

PADUA JUNIOR, Henry Borges; LARANJO, Rafael Bisinotto. Os benefícios do desenvolvimento multiplataformas. **UNIUBE**, 2020. Trabalho de Conclusão de Curso (TCC).

POLICASTRO, Rafael Gurgel. MartList: Aplicação Móvel para Criação de Listas de Compras Digitais. **CEFET-RJ**, 2016.

RICARDO, Rodrigues. Novas Tecnologias da Informação e da Comunicação. **Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco - Campus Recife e a Universidade Federal de Santa Maria para a Rede e-Tec Brasil**, 2016.

RYAN, Mark. Implementing a Design System for Mobile Cross Platform Development. **Theseus**, 2020.

SANCHES, Leonardo. **Aplicativo nativo ou híbrido: qual escolher? - Leanwork Blog**. 2024. Disponível em: <https://blog.leanwork.com.br/aplicativo-nativo-ou-hibrido-qual-escolher/>. Acesso em: 15 out. 2024.

SCHÖNING, Johannes. Does a “Smart Campus” Create “Smart People”? From Smart Cities to Smart Campuses — Supporting the “Campus Citizens”. **Specialist Meeting—Advancing the Spatially Enabled Smart Campus**, 2014.

SILVA, Laira Thamys de Araujo. Transporte universitário como política de acesso e permanência no Ensino Superior: Análise situacional sobre o programa Passe Livre Universitário no município de Itaocara – RJ (2013-2020). **Galoá Proceedings**, 2022.

SILVA, Maria Eduarda Ramos da; SERRANO, Paulo Henrique Souto Maior. Análise de sentimentos em textos de redes sociais: uma comparação entre o ChatGPT e métodos tradicionais. **Cadernos de Comunicação**, 2023.

SILVA FLOR, Clarissa da; TEIXEIRA, Clarissa Stefani. CIDADES INTELIGENTES E EMPREENDEDORAS: UM ESTUDO COMPARATIVO ENTRE RANKINGS. *In*: ANAIS do Congresso Internacional de Conhecimento e Inovação–ciki. [*S.l.*: *s.n.*], 2018.

TECH. **How to Build Real-time Apps with Flutter and Firebase : The Ultimate Guide - CodeEpsilon**. 2024. Disponível em: <https://www.codeepsilon.com/how-to-build-real-time-apps-with-flutter-and-firebase-the-ultimate-guide/>. Acesso em: 27 out. 2024.

UNILAB, Seção de Portais e Aplicações WEB (SPA/DSI/DTI/Unilab). **Ônibus Intercampi**. 2024. Disponível em: <https://unilab.edu.br/onibus-intercampi/>. Acesso em: 16 out. 2024.

UZUN, Yesim. Integration of Flutter Framework in Real-Life Applications: Technical and Development Practices. **UNIPD**, 2023.

WOLF, Thomas; DEBUT, Lysandre; SANH, Victor; CHAUMOND, Julien; DELANGUE, Clement *et al.* Transformers: State-of-the-Art Natural Language Processing. *In*: PROCEEDINGS of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. [*S.l.*]: Association for Computational Linguistics, 2020.

WOLF, Thomas *et al.* Transformers: State-of-the-art natural language processing. *In*: PROCEEDINGS of the 2020 conference on empirical methods in natural language processing: system demonstrations. [*S.l.*: *s.n.*], 2020.

ZENGO, Manuel Lucala. Desenvolvimento de um sistema tecnológico integrado para o gerenciamento e monitoramento dos veículos da Unilab. **UNILAB**, 2023.

Anexos

Anexo A – Exemplo de dados coletados do Google Play

Exemplo dos dados coletados do Google play após o primeiro tratamento no Excel.

data;nota;texto da review

28/03/2022;5;👏👏👏👏

04/04/2022;1;Ameeeeiii super recomendo Seria bom colocar uma rota pra Barreira tmb...

07/05/2022;5;Amei o app 😍😍😍!!! A cada dia mais fascinada pela praticidade da Unilab.

05/08/2022;5;Boa!

18/01/2021;5;É uma ótima forma de ficar bem informado dos horários do Intercamp.

25/10/2021;4;Genial

26/11/2021;5;Gostei muito do aplicativo! Sugiro que coloque uma notificação pra lembrar o usuário sobre o horário do próximo ônibus, seria uma boa.

06/01/2020;4;Incrível!!!! 👏👏👏👏👏👏👏

31/01/2020;5;Muito bom o aplicativo Parabéns ao desenvolver. Realmente estávamos precisando de algo assim. Recomendo

06/02/2020;5;show

25/02/2020;5;Show demais

02/03/2020;5;

07/05/2019;3;

07/05/2019;5;5 estrelas pela ideia maravilhosa.

07/05/2019;4;Adorei muito bom o aplicativo..

07/05/2019;5;Amei😂😭

08/05/2019;5;Excelente iniciativa

09/05/2019;1;Muito bom

09/05/2019;5;Muito bom parabéns pelo o aplicativo.👏👏

09/05/2019;5;Parabéns pela iniciativa.

10/05/2019;5;Parabéns, show

10/05/2019;5;

12/05/2019;5;

13/05/2019;5;

14/05/2019;5;Bom trabalho!

15/05/2019;5;Muito util...poderia adicionar notificações e em alguns casos a pessoa poderia dizer se quer receber notificação na hora ou 10 min antes para poder se programar

22/05/2019;5;

Anexo B – Código completo da Inteligência Artificial

Hugging Face Transformers com o modelo *nlptown/bert-base-multilingual-uncased-sentiment* escrito em Python.

```
import pandas as pd
from transformers import pipeline

# Carregar os dados do arquivo CSV
df = pd.read_csv('feedbacks_intercampi.csv', sep=';')

# Inicializar o pipeline de análise de sentimento com um modelo multilinguagem
sentiment_analysis = pipeline("sentiment-analysis", model="nlptown/bert-base-multilingual-uncased-sentiment")

# Função para analisar o sentimento de um texto
def analyze_sentiment(text):
    if isinstance(text, str):
        result = sentiment_analysis(text)
        label = result[0]['label']
        score = result[0]['score']
        # Converter rótulos para um formato mais intuitivo
        if label == '1 star':
            label = 'Very Negative'
        elif label == '2 stars':
            label = 'Negative'
        elif label == '3 stars':
            label = 'Neutral'
        elif label == '4 stars':
            label = 'Positive'
        elif label == '5 stars':
            label = 'Very Positive'
        return label, score
    else:
        return None, None

# Aplicar a análise de sentimento às reviews
df['sentiment_label'], df['sentiment_score'] = zip(*df['texto da review'].apply(analyze_sentiment))

# Ver os resultados
print(df[['data', 'nota', 'texto da review', 'sentiment_label', 'sentiment_score']])

# Salvar o DataFrame com as novas colunas em um novo arquivo CSV
df.to_csv('review_data_with_sentiment2.csv', index=False)
```

Anexo C – Dados resultantes pós-processamento com IA

Parte dos dados após o processamento pela Inteligência Artificial. Resultado final.

data,nota,texto da review,sentiment_label,sentiment_score
28/03/2022,5,👏👏👏👏,Very Positive,0.3085779845714569
04/04/2022,1,Ameeeeiii super recomendo Seria bom colocar uma rota pra Barreira tmb...,Very Positive,0.6196050643920898
07/05/2022,5,Amei o app 😍😍😍!!! A cada dia mais fascinada pela praticidade da Unilab.,Very Positive,0.773659884929657
05/08/2022,5,Boa!,Positive,0.4047538936138153
18/01/2021,5,É uma ótima forma de ficar bem informado dos horários do Intercamp.,Positive,0.47060903906822205
25/10/2021,4,Genial,Very Positive,0.8408788442611694
26/11/2021,5,"Gostei muito do aplicativo! Sugiro que coloque uma notificação pra lembrar o usuário sobre o horário do próximo ônibus, seria uma boa.",Positive,0.3924850523471832
06/01/2020,4,Incrível!!!! 👏👏👏👏👏👏👏,Very Negative,0.4353735148906708
31/01/2020,5,Muito bom o aplicativo Parabéns ao desenvolver. Realmente estávamos precisando de algo assim. Recomendo,Very Positive,0.610787034034729
06/02/2020,5,show,Very Positive,0.3741426169872284
25/02/2020,5,Show demais,Very Negative,0.29349467158317566
02/03/2020,5,,,
02/03/2020,1,,,
02/03/2020,5,,,
05/03/2020,5,,,
12/03/2020,5,,,
30/01/2019,5,,,
27/02/2019,4,,,
06/03/2019,5,,,
07/03/2019,5,,,
11/03/2019,5,,,
29/04/2019,5,,,
02/05/2019,5,,,
07/05/2019,3,,,
07/05/2019,5,5 estrelas pela ideia maravilhosa.,Very Positive,0.9904047250747681
07/05/2019,4,Adorei muito bom o aplicativo..,Very Positive,0.6184130907058716
07/05/2019,5,Amei😂🤗,Very Positive,0.3085779845714569
08/05/2019,5,Excelente iniciativa,Very Positive,0.7531045079231262
09/05/2019,1,Muito bom,Very Positive,0.6459388732910156
09/05/2019,5,Muito bom parabéns pelo o aplicativo.👏👏,Very Positive,0.5423296689987183
09/05/2019,5,Parabéns pela iniciativa.,Very Positive,0.3112245202064514
10/05/2019,5,"Parabéns, show",Very Positive,0.2840613126754761
10/05/2019,5,,,
12/05/2019,5,,,
12/05/2019,5,,,
...