**UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES**
**CAMPUS DE ERECHIM**
**DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO CURSO DE CIÊNCIA DA COMPUTAÇÃO**

Bernardo Cenci Barro

# GERENCIADOR DE TAREFAS INTELIGENTE

**ERECHIM - RS**
**2023**

Bernardo Cenci Barro

# GERENCIADOR DE TAREFAS INTELIGENTE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel, Departamento de Engenharias e Ciência da Computação da Universidade Regional Integrada do Alto Uruguai e das Missões Campus de Erechim.

Orientador: Prof. Dr. Guilherme Afonso Madalozzo

**ERECHIM - RS**

**2023**

## AGRADECIMENTOS

Minha profunda gratidão é estendida a todas as pessoas que tiveram um papel significativo na realização deste trabalho de conclusão de curso. A concretização deste projeto só foi possível graças ao suporte e incentivo de muitos indivíduos e grupos que foram essenciais em minha trajetória acadêmica.

Em primeiro lugar, desejo expressar minha sincera gratidão à minha família, particularmente aos meus pais e a minha irmã, que sempre me apoiaram, fornecendo suporte emocional indispensável para a minha formação superior. Minha namorada, mesmo estando fisicamente distante, sempre me ofereceu seu apoio incondicional, demonstrando seu afeto e confiança em meu êxito.

Agradeço também a todos os meus colegas de curso, com quem compartilhei os desafios e alegrias desta caminhada acadêmica. Suas contribuições, debates e amizade enriqueceram imensamente esta experiência.

É imprescindível mencionar o papel crucial do meu orientador, Professor Guilherme Afonso Madalozzo. Sua orientação, sabedoria e suporte foram fundamentais no desenvolvimento deste trabalho. Sua rigorosa supervisão e comprometimento com meu desenvolvimento acadêmico foram de valor incalculável.

Além disso, estendo meus agradecimentos a todos os professores que fizeram parte da minha jornada educacional. Cada um deles, com sua expertise e experiência, contribuiu para o meu crescimento intelectual e expandiu minha perspectiva de mundo.

Por último, sou grato a todos que, de alguma forma, influenciaram e apoiaram meu percurso acadêmico. Este trabalho de conclusão de curso é o resultado de todos esses esforços coletivos, e sou imensamente agradecido pela oportunidade de aprender e evoluir neste ambiente acadêmico.

Meu sincero obrigado a todos.

## RESUMO

O "Gerenciador de Tarefas Inteligente" é um aplicativo móvel projetado especificamente para melhorar a produtividade e eficiência no gerenciamento de tarefas pessoais e profissionais. Este aplicativo se destaca pela integração de tecnologias avançadas de inteligência artificial e reconhecimento de voz, proporcionando uma experiência de usuário única e intuitiva. Com uma assistente virtual incorporada, o aplicativo permite aos usuários gerenciar suas tarefas de maneira eficaz, oferecendo funcionalidades como criação, edição, visualização e exclusão de tarefas, além de um calendário integrado para o acompanhamento de prazos e eventos. O desenvolvimento do aplicativo envolveu um processo meticuloso de estudo, prototipagem, desenvolvimento e análise de resultados, com ênfase na aplicação de conceitos de assistentes virtuais e APIs de reconhecimento de voz. O objetivo principal é fornecer uma ferramenta prática e inovadora que contribua significativamente para o aumento da produtividade e eficiência dos usuários, tanto em suas vidas pessoais quanto profissionais.


**Palavras-chave:** Aplicativo móvel, Inteligência artificial, Gerenciamento de tarefas.

## ABSTRACT

The "Intelligent Task Manager" is a mobile application specifically designed to enhance productivity and efficiency in managing personal and professional tasks. This app stands out for its integration of advanced artificial intelligence and voice recognition technologies, providing a unique and intuitive user experience. With an incorporated virtual assistant, the application enables users to effectively manage their tasks, offering functionalities such as task creation, editing, viewing, and deletion, along with an integrated calendar for tracking deadlines and events. The development of the app involved a meticulous process of study, prototyping, development, and analysis of results, with an emphasis on the application of virtual assistant concepts and voice recognition APIs. The main goal is to provide a practical and innovative tool that significantly contributes to increasing the productivity and efficiency of users in both their personal and professional lives.


**Keywords:** Mobile application, Artificial intelligence, Task management.

## LISTA DE FIGURAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AOT | *Ahead of Time* |
| API | *Application Programming Interface* |
| BaaS | *Backend As A Service* |
| FCM | *Firebase Cloud Messaging* |
| IA | *Inteligência Artificial* |
| JIT | *Just in Time* |
| JSON | *JavaScript Object Notation* |
| ML | *Machine Learning* |
| NDK | *Native Development Kit* |
| PC | *Personal Computer* |
| PLN | *Processamento de Linguagem Natural* |
| REST | *Representational State Transfer* |
| UI | *User Interface* |
| UID | *Unique Identifier* |
| UX | *User Experience* |
| XML | *Extensible Markup Language* |

**SUMÁRIO**

# 1 INTRODUÇÃO

Em um mundo onde a eficiência e a produtividade são cada vez mais valorizadas, tanto no âmbito profissional quanto pessoal, a capacidade de gerenciar eficazmente as tarefas do dia a dia torna-se um diferencial competitivo. O gerenciamento de tarefas é uma habilidade fundamental que permite às pessoas maximizarem seu tempo e recursos, potencializando seu desempenho (Vichare *et al*. 2022). No entanto, a complexidade das rotinas modernas frequentemente excede as capacidades tradicionais de organização, levando à necessidade de soluções inovadoras que possam auxiliar na gestão eficiente de atividades.

Diante desse cenário, a tecnologia, especialmente os aplicativos móveis, surge como um recurso valioso, transformando a maneira como gerenciamos tarefas e compromissos. A integração de Inteligência Artificial (IA) e reconhecimento de voz em aplicativos móveis tem demonstrado um potencial significativo para melhorar a eficiência e a produtividade na gestão de tarefas. Estudos como *"The Application of Artificial Intelligence Voice Recognition on Helping Elders Use Mobile Phones More Easily"* e "*Artificial Intelligence Can Improve Patient Management at the Time of a Pandemic: The Role of Voice Technology*" ilustram como essas tecnologias podem ser aplicadas para facilitar a interação do usuário com dispositivos móveis, oferecendo uma experiência mais intuitiva e personalizada (Yao, 2021; Jadczyk et al. 2020).

A Inteligência Artificial, o reconhecimento de voz e o Processamento de Linguagem Natural (PLN) são tecnologias que têm o potencial de aprimorar significativamente os aplicativos de gerenciamento de tarefas. A IA pode aprender com as preferências e comportamentos do usuário, oferecendo sugestões personalizadas e melhorando a experiência do usuário (Yao, 2021). O reconhecimento de voz, por sua vez, permite uma interação mais natural e mãos livres com o aplicativo, facilitando o uso em diferentes contextos, seja durante o deslocamento ou enquanto se realiza outra tarefa (Jadczyk *et al*. 2020).

A integração do PLN permite que os aplicativos compreendam e processem a linguagem humana com maior precisão, tornando possível a criação de notas e lembretes por meio de comandos de voz simples e intuitivos (Vichare *et al*. 2022). Esta capacidade de interpretar e agir sobre a linguagem natural é particularmente relevante em um cenário onde a rapidez e a facilidade de uso são cruciais para a adesão do

usuário e a eficácia do gerenciamento de tarefas.

Neste cenário, o desenvolvimento de um aplicativo de gerenciamento de tarefas que incorpora IA, reconhecimento de voz e PLN representa uma evolução natural e necessária. O objetivo é criar uma ferramenta que não só ajude os usuários a organizarem suas tarefas de forma mais eficiente, mas que também aprenda e se adapte às suas necessidades específicas, melhorando continuamente a gestão do tempo e a produtividade (Vichare *et al*. 2022; Yao, 2021; Jadczyk *et al*. 2020).

Este trabalho é estruturado em seis capítulos, incluindo a Introdução. No segundo Capítulo, há uma discussão detalhada sobre as principais tecnologias usadas, estabelecendo um referencial teórico com os conceitos chave encontrados durante a revisão da literatura, que formam a base do estudo. O terceiro Capítulo descreve as ferramentas e métodos escolhidos que foram fundamentais para o desenvolvimento do sistema. No quarto Capítulo, o foco é como o aplicativo foi desenvolvido, incluindo uma descrição das ferramentas e métodos usados, bem como das interfaces do usuário. O quinto Capítulo olha para as melhorias e atualizações planejadas para o aplicativo. Por fim, o sexto Capítulo conclui o trabalho, trazendo as considerações finais do autor e um resumo das principais descobertas e conclusões do projeto.

# 2 REFERENCIAL TEÓRICO

Neste Capítulo, procede-se à construção de um referencial teórico robusto, que se configura como o alicerce essencial para o desenvolvimento do aplicativo em questão. Aborda-se, com a devida profundidade, o domínio dos aplicativos móveis, explorando especificamente a linguagem de programação Dart juntamente com o seu *framework* associado, Flutter. Além disso, discute-se a aplicação de inteligência artificial, com ênfase particular no reconhecimento de voz e no processamento de linguagem natural, delineando como estas tecnologias convergem para a funcionalidade e eficácia do aplicativo.

## 2.1 Desenvolvimento de Aplicativos Móveis

O desenvolvimento de aplicativos móveis é uma área de constante inovação e adaptação, impulsionada pela necessidade de atender às expectativas de usuários cada vez mais conectados e dependentes de soluções digitais. A integração de funcionalidades avançadas, como pagamentos móveis, geolocalização e integração com redes sociais, tornou-se padrão em muitos aplicativos, refletindo a evolução das demandas do mercado e das capacidades dos dispositivos móveis (Kumari *et al.* 2023).

A escolha da plataforma de desenvolvimento é uma decisão estratégica que pode determinar o alcance e a acessibilidade do aplicativo. O *Android*, com sua base de código aberto e ampla adoção em diversos dispositivos, oferece uma flexibilidade significativa para os desenvolvedores. Por outro lado, o *iOS*, com seu ecossistema mais restrito, proporciona uma experiência de usuário otimizada e um mercado potencialmente mais lucrativo para aplicativos pagos. A decisão entre essas plataformas ou a adoção de uma abordagem de desenvolvimento multiplataforma deve ser baseada em uma análise cuidadosa do público-alvo e dos objetivos de negócios do aplicativo (Tobing, 2023).

O desenvolvimento multiplataforma, facilitado por *framework*s como React Native e Flutter, permite que os desenvolvedores criem aplicativos que funcionam em múltiplos sistemas operacionais com uma única base de código. Isso pode reduzir significativamente o tempo e os custos de desenvolvimento, mas também exige uma

atenção especial à otimização de desempenho e à consistência da experiência do usuário em diferentes dispositivos. A escolha de uma ferramenta de desenvolvimento deve levar em consideração não apenas as necessidades atuais, mas também a facilidade de atualizações futuras e a manutenção a longo prazo (Castillo López *et al.* 2023).

Com a pandemia de *Covid-19*, a importância dos aplicativos móveis foi ainda mais enfatizada, tornando-se ferramentas vitais para a continuidade das atividades diárias em um cenário de distanciamento social. Isso impulsionou uma rápida transformação digital, exigindo dos desenvolvedores uma resposta ágil para criar aplicativos que atendessem às novas necessidades de comunicação, trabalho remoto, educação e saúde. A capacidade de desenvolver rapidamente aplicativos confiáveis e seguros tornou-se um diferencial competitivo importante no mercado atual (Novia; Matahari; Setiawan, 2023).

## 2.2 Inteligência Artificial

A Inteligência Artificial, um campo pioneiro dentro da ciência da computação, se dedica à criação de sistemas e algoritmos que não apenas imitam, mas também potencializam a cognição humana. Esta disciplina abarca uma gama de técnicas sofisticadas, desde o raciocínio lógico até o aprendizado profundo, permitindo que as máquinas executem tarefas que requerem habilidades cognitivas complexas, como reconhecimento de padrões, tomada de decisões e processamento de linguagem natural. A IA tem se mostrado uma força motriz em inúmeras áreas, revolucionando campos tão diversos quanto a medicina, as finanças, o transporte, a educação e a indústria, e redefinindo nossa interação com a tecnologia em um ritmo sem precedentes (Carvalho Neto *et al.* 2023).

Desde sua concepção, a inteligência artificial foi idealizada para criar sistemas capazes de executar tarefas complexas com eficiência e autonomia, imitando a capacidade de adaptação e aprendizado que caracteriza a inteligência humana. Uma das subáreas mais dinâmicas da IA é o *Machine Learning* (ML), que se concentra em desenvolver algoritmos que capacitam as máquinas a aprender a partir de dados e aprimorar suas habilidades com o tempo. Esses algoritmos são a chave para a automação avançada, permitindo que as máquinas não apenas executem tarefas

rotineiras, mas também analisem grandes conjuntos de dados e resolvam problemas complexos. O impacto dessas tecnologias é vasto e profundo, oferecendo avanços significativos em diversos setores e melhorando a eficiência e a qualidade dos serviços e produtos oferecidos (Carvalho Neto *et al.* 2023).

### **2.2.1** Inteligência Artificial em Aplicativos Móveis e o Aprimoramento da Eficiência e Desempenho

A integração da inteligência artificial em aplicativos móveis é uma revolução que está redefinindo a interação entre usuários e dispositivos. A IA não é apenas uma ferramenta para automatizar tarefas, mas uma força transformadora que personaliza a experiência do usuário, tornando-a mais intuitiva e eficiente. Com o avanço do aprendizado de máquina e da capacidade de processamento de linguagem natural, os aplicativos móveis estão se tornando mais adaptativos e proativos. Eles aprendem e evoluem com o comportamento do usuário, antecipando necessidades e preferências, o que resulta em uma usabilidade aprimorada e, consequentemente, maior retenção de usuários em um mercado altamente competitivo (Kurzweil, 2005; Russell; Norvig, 2016).

A eficiência operacional dos aplicativos é significativamente aprimorada pela IA. Algoritmos inteligentes gerenciam os recursos do sistema de maneira otimizada, contribuindo para a redução do consumo de bateria e a melhoria do desempenho em uma ampla gama de dispositivos móveis. Esta gestão eficiente é crucial, considerando a diversidade de capacidades de *hardware* existentes no mercado. A IA permite que os aplicativos se ajustem e operem de forma eficaz, independentemente das limitações do dispositivo, garantindo uma experiência de usuário consistente e confiável (Haugeland, 1985).

A segurança é outra área em que a IA tem um impacto significativo nos aplicativos móveis. Técnicas avançadas de aprendizado de máquina são empregadas para monitorar e aprender padrões de uso, identificando comportamentos anormais que podem sinalizar tentativas de fraude ou invasão. Isso não só protege os usuários contra ameaças de segurança, mas também ajuda os desenvolvedores a entenderem melhor e responder a vulnerabilidades potenciais, mantendo a integridade e a confiança no aplicativo (Charniak; Mcdermott, 2018).

Além disso, a IA está transformando o ciclo de vida do desenvolvimento de aplicativos móveis. Desde a concepção até a manutenção, insights fornecidos pela IA podem acelerar o desenvolvimento e melhorar a qualidade do produto. A IA pode ajudar a identificar tendências de uso e *feedback* dos usuários, permitindo que os desenvolvedores se concentrem em melhorias e recursos que realmente importam para o público-alvo. Isso resulta em um ciclo de desenvolvimento mais eficiente e em produtos que estão mais alinhados com as expectativas dos usuários.

A IA também está impulsionando a personalização em uma escala sem precedentes. Aplicativos móveis equipados com IA podem oferecer experiências altamente personalizadas, ajustando o conteúdo e as funcionalidades às preferências individuais do usuário. Isso é possível graças à capacidade da IA de analisar grandes volumes de dados e identificar padrões e preferências únicas. Essa personalização não apenas melhora a satisfação do usuário, mas também abre novas oportunidades para os desenvolvedores criarem serviços e produtos mais direcionados e eficazes (Russell; Norvig, 2016).

Por fim, a IA está facilitando a criação de interfaces de usuário mais naturais e interativas. Com o avanço das tecnologias de reconhecimento de voz e visão computacional, os aplicativos móveis estão se tornando mais acessíveis e convenientes. Os usuários podem interagir com seus aplicativos de maneira mais natural, usando a voz ou gestos, o que pode aumentar a acessibilidade para usuários com diferentes capacidades e preferências. Isso não apenas melhora a experiência do usuário, mas também amplia o alcance dos aplicativos móveis para um público mais amplo (Poole; Mackworth; Goebel, 1998).

## 2.3 Reconhecimento de Voz

O reconhecimento de voz é uma área fascinante da computação que se entrelaça com diversas disciplinas, como linguística, ciência da computação e engenharia elétrica. A história do reconhecimento de voz remonta aos anos 1950, com os primeiros sistemas capazes de reconhecer dígitos falados por um único usuário (Davis; Biddulph; Balashek, 1952). Desde então, a tecnologia evoluiu de simples dispositivos de reconhecimento de padrões para sistemas complexos que utilizam inteligência artificial e aprendizado de máquina para entender a fala humana com

precisão cada vez maior.

Nos anos 1970 e 1980, o reconhecimento de voz começou a ser explorado para aplicações práticas, mas foi somente com o advento da internet e o aumento da capacidade de processamento dos computadores que a tecnologia começou a se popularizar. O reconhecimento de voz tornou-se mais robusto e adaptável, capaz de reconhecer diferentes sotaques e variações na fala (Rabiner; Juang, 1993).

Com a chegada dos *smartphones*, o reconhecimento de voz encontrou um novo e vasto campo de aplicação. Aplicativos móveis começaram a incorporar essa tecnologia para oferecer uma interação mais natural e mãos livres com o usuário. O gerenciamento de tarefas, em particular, é uma área que se beneficia enormemente do reconhecimento de voz, permitindo que os usuários adicionem, modifiquem e consultem suas tarefas de maneira rápida e intuitiva.

A eficiência do reconhecimento de voz em aplicativos de gerenciamento de tarefas é evidenciada pela sua capacidade de processar comandos de voz em tempo real, mesmo em ambientes ruidosos ou em situações em que o usuário não pode interagir fisicamente com o dispositivo. Isso é possível graças a algoritmos avançados de processamento de sinal e redes neurais profundas, que aprendem a partir de grandes conjuntos de dados de fala para melhorar a precisão do reconhecimento (Hinton *et al*. 2012).

Além disso, a integração do reconhecimento de voz com tecnologias de assistentes virtuais, como *Siri*, *Google Assistant* e *Alexa*, ampliou ainda mais as possibilidades de interação com aplicativos de gerenciamento de tarefas. Os usuários podem agora, por exemplo, pedir ao seu assistente virtual para agendar uma reunião ou lembrá-los de uma tarefa pendente, tudo isso usando comandos de voz naturais (Amershi *et al*. 2019).

No entanto, apesar dos avanços, ainda existem desafios a serem superados. A precisão do reconhecimento de voz pode ser afetada por diversos fatores, como ruído de fundo, sotaques e dialetos, e a maneira como as pessoas naturalmente pausam e enfatizam palavras durante a fala. Pesquisadores continuam trabalhando para melhorar os modelos de linguagem e as técnicas de aprendizado de máquina para tornar o reconhecimento de voz ainda mais confiável (Chan *et al*. 2016).

O futuro do reconhecimento de voz em aplicativos móveis parece promissor, com pesquisas apontando para sistemas cada vez mais integrados e personalizados. A tendência é que a tecnologia se torne onipresente, com aplicativos capazes de

entender não apenas o que é dito, mas também o contexto e a intenção por trás das palavras, proporcionando uma experiência de usuário ainda mais rica e eficiente (Saon *et al*. 2017).

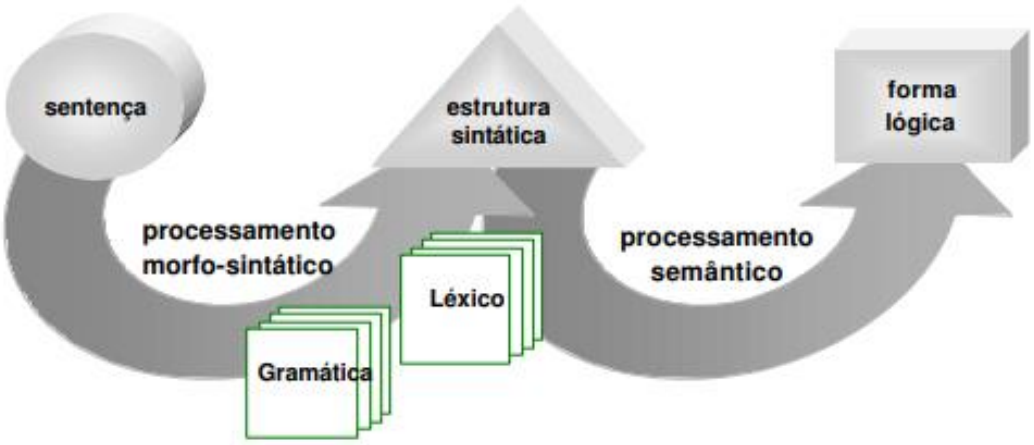
## 2.4 Processamento de Linguagem Natural

O processamento da linguagem natural aborda computacionalmente os diversos aspectos da comunicação humana, incluindo sons, palavras, sentenças e discursos, considerando formatos e referências, estruturas e significados, contextos e usos. Em um sentido amplo, o objetivo do PLN é permitir que o computador se comunique em linguagem humana, abrangendo vários níveis de entendimento e/ou geração de sons, palavras, sentenças e discursos. Estes níveis incluem o fonético e fonológico, relacionado ao som das palavras, o morfológico, que trata da construção das palavras a partir de unidades de significado primitivas, o sintático, que aborda o relacionamento das palavras nas frases, o semântico, focado no significado das palavras e sentenças, e o pragmático, que considera o uso de frases e sentenças em diferentes contextos.

A representação do significado de uma sentença, independente de contexto, é obtida através de sua forma lógica, que codifica os possíveis sentidos de cada palavra e identifica os relacionamentos semânticos entre palavras e frases. A estrutura sintática de uma sentença é obtida pelo processamento morfossintático, regida por leis gramaticais definidas em uma gramática, com informações adicionais fornecidas por um léxico. O mapeamento da estrutura sintática para a forma lógica é realizado pelo processamento semântico, onde o léxico também desempenha um papel fundamental (Gonzalez, 2003). A Figura 1 ilustra o processo de transformação de uma sentença em sua forma lógica.

Historicamente, o PLN começou com sistemas baseados em regras rígidas e evoluiu para técnicas mais sofisticadas que utilizam aprendizado de máquina e inteligência artificial. Esta evolução permitiu uma compreensão mais profunda e interações mais naturais com a linguagem humana. A importância da extração de conhecimento no contexto do PLN, onde a capacidade de extrair informações significativas de grandes conjuntos de dados textuais se tornou um aspecto crucial, é ressaltada por Sara Catarina Silva Pinto.

Figura 1 – Transformações da sentença na estrutura sintática e na forma lógica




Fonte: Gonzalez (2003).

As aplicações práticas do PLN são diversas e impactantes. Em sistemas de resposta automática e análise de sentimentos em redes sociais, o PLN permite uma análise rápida e eficiente de grandes volumes de dados textuais. Renata Vieira explora como o PLN é aplicado na análise de linguagens especializadas e corpora, demonstrando sua versatilidade em diferentes contextos linguísticos. Além disso, uma das aplicações mais desafiadoras e úteis do PLN é a tradução automática. Esta aplicação não apenas facilita a comunicação entre falantes de diferentes idiomas, mas também abre portas para uma compreensão mais profunda de textos e discursos multilíngues. A tradução automática, embora tenha avançado significativamente, ainda enfrenta desafios, especialmente em relação à captura de nuances culturais e contextuais.

O PLN também levanta questões éticas relacionadas à privacidade e ao viés algorítmico. Estas questões são cruciais para o desenvolvimento responsável de tecnologias baseadas em PLN, pois têm o potencial de influenciar percepções e decisões em diversos níveis. A ética no PLN não se limita apenas à privacidade dos dados, mas também abrange a forma como os algoritmos são treinados e os dados são utilizados, garantindo que não perpetuem preconceitos ou vieses existentes. Além disso, a transparência nos algoritmos de PLN é fundamental para garantir a confiança do usuário e a responsabilidade dos desenvolvedores.

# 3 MATERIAIS E MÉTODOS

Neste Capítulo, detalharei as ferramentas de desenvolvimento selecionadas para este projeto, enfatizando suas funcionalidades e a razão pela qual foram escolhidas. Além disso, discutirei as abordagens adotadas no desenvolvimento, abrangendo desde o *design* das telas até a elaboração dos fluxos do projeto. Esta Seção visa proporcionar uma compreensão abrangente das metodologias e tecnologias empregadas, bem como a lógica por trás das decisões de *design* e desenvolvimento que moldaram o curso deste trabalho.

## 3.1 Ferramentas de Desenvolvimento

Nesta Seção, abordaremos as ferramentas selecionadas para o desenvolvimento do aplicativo, incluindo a linguagem de programação e o *framework* adotados, bem como serviços externos como *Firebase* e *Google Calendar API*, detalhando suas funções e o papel que desempenham no projeto.

### 3.1.1 Flutter

Flutter, um *framework* desenvolvido pela Google, foi primeiramente anunciado em 2015 durante uma apresentação de Eric Seidel. Em sua fase experimental, era conhecido como *Sky*, sendo posteriormente rebatizado para Flutter (Bueno, 2021). Conforme descrito na documentação oficial da ferramenta, Flutter é definido como o "*kit* de ferramentas de UI portátil do Google para criar aplicativos belos e compilados de forma nativa para dispositivos móveis, *web* e *desktop* a partir de uma única base de código. O Flutter é utilizado por desenvolvedores e organizações ao redor do mundo, sendo gratuito e de código aberto" (Flutter, 2023).

O projeto Flutter teve início como um experimento conduzido por desenvolvedores do Google, que buscavam otimizar o navegador *Chrome* removendo camadas de suporte de compatibilidade, visando uma execução mais rápida. Surpreendentemente, os resultados dos testes indicaram que o Flutter processava até vinte vezes mais rápido que o *Chrome* (Mainkar; Giordano, 2019).

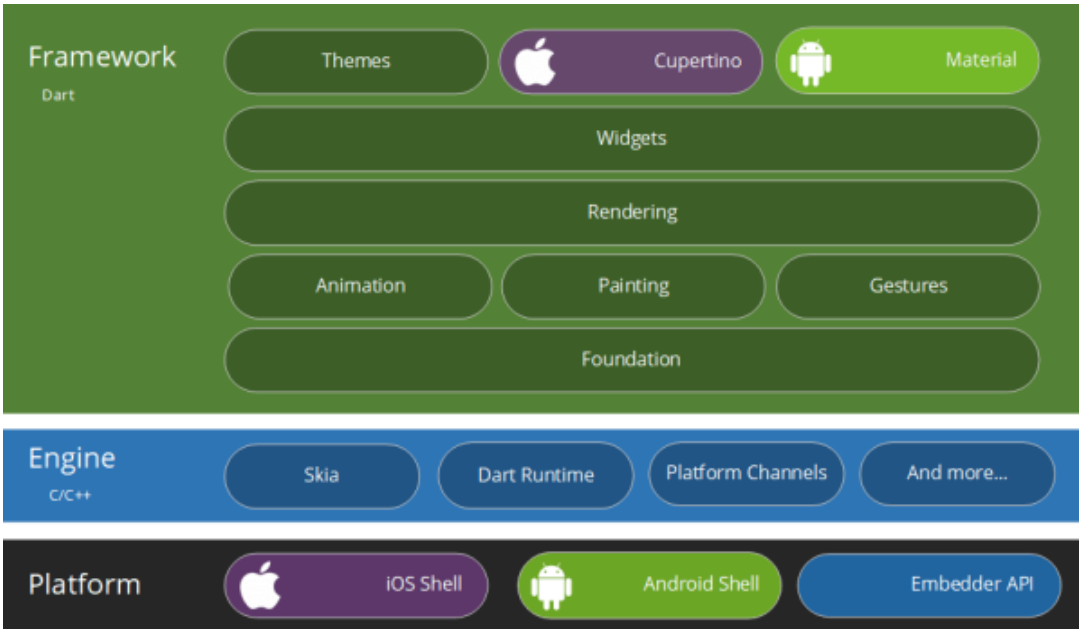
Como um *framework* de desenvolvimento multiplataforma, o Flutter atualmente suporta o desenvolvimento para dispositivos móveis e PCs, e tem planos de expandir

seu suporte para desenvolvimento *web*, conforme indicado em seu *roadmap* (Mainkar; Giordano, 2019; Flutter, 2023).

O desenvolvimento no Flutter é realizado utilizando a linguagem Dart, uma linguagem que segue o modelo da linguagem C, é orientada a objetos, baseada em classes, com um sistema de tipos opcional e suporte a heranças (Google, 2013; Bracha, 2015).

Diferentemente de outras linguagens multiplataforma, o Flutter não se baseia em uma lista de componentes com equivalentes nativos, mas sim possui sua própria *engine* de renderização. Esta *engine* é compilada através da *Android* NDK, e o código Dart é convertido em código nativo. Este processo utiliza o *Skia* para o desenho da interface, compilando os *Widgets* diretamente em código nativo (Mainkar; Giordano, 2019; Flutter, 2023). É possível ver na Figura 2 o diagrama da linguagem:

Figura 2 – Diagrama dos componentes do Flutter




Fonte: Jim, Simon (2020)

A escolha do Flutter em detrimento do React Native para o desenvolvimento de aplicativos móveis se justifica por várias razões. Primeiramente, a abordagem única do Flutter na construção de interfaces de usuário, baseada em um extenso conjunto de *widgets* pré-construídos e personalizáveis, facilita a criação de interfaces complexas e responsivas. Esta filosofia de “tudo é um *widget*” permite um desenvolvimento mais eficiente e consistente em diferentes plataformas,

economizando tempo e recursos valiosos (Bueno, 2021; Flutter, 2023).

Além disso, a capacidade do Flutter de compilar diretamente para código nativo oferece uma vantagem significativa sobre outras plataformas como o React Native. Isso melhora o desempenho e simplifica o desenvolvimento, eliminando a necessidade de gerenciar bases de código separadas para diferentes plataformas. A possibilidade de usar uma única base de código para criar aplicativos para *iOS*, *Android*, *web* e *desktop* é uma economia significativa de tempo e recursos, tornando o Flutter uma escolha atraente para desenvolvedores e empresas (Mainkar; Giordano, 2019; Flutter, 2023).

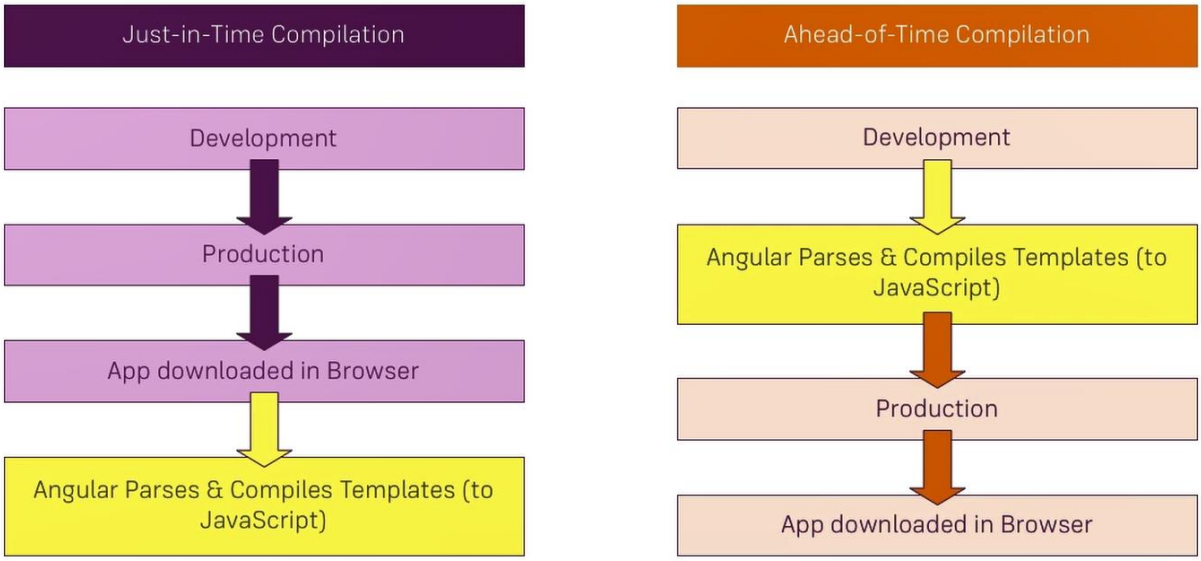
### 3.1.2 Características Técnicas

O Flutter se destaca no cenário de desenvolvimento de aplicativos móveis por suas características técnicas inovadoras, como a compilação *Just-in-Time* (JIT) e *Ahead-of-Time* (AOT), além da funcionalidade de *hot reload*. Essas funcionalidades marcam um avanço significativo tanto na otimização do processo de desenvolvimento quanto na elevação da experiência do usuário.

A compilação JIT, que ocorre durante a execução do aplicativo, desempenha um papel crucial na fase de desenvolvimento com Flutter. Esta técnica permite um ciclo de desenvolvimento mais dinâmico e uma iteração ágil, pois o código é compilado em tempo real. Esse recurso é extremamente vantajoso durante as fases de teste e depuração, onde alterações frequentes no código são uma constante. A compilação JIT promove uma experimentação eficaz e uma adaptação rápida do código, elementos fundamentais para um desenvolvimento eficiente e adaptável (Flutter, 2023).

Em contrapartida, a compilação AOT é utilizada na fase de preparação do aplicativo para o ambiente de produção. Neste método, o código Dart é transformado em código de máquina nativo antes mesmo da execução do programa. Esta abordagem é essencial para aprimorar o desempenho do aplicativo, proporcionando um lançamento mais rápido e uma execução mais fluida. A compilação AOT é eficaz pois converte o código em um formato que é diretamente executável pelo *hardware* do dispositivo, eliminando a necessidade de um interpretador durante a execução, o que é decisivo para a otimização do desempenho em dispositivos móveis (Windmill, 2020). A Figura 3 ilustra a comparação de compilação JIT e AOT.

Figura 3 – Diagrama de fluxo de compilação JIT e AOT do Flutter



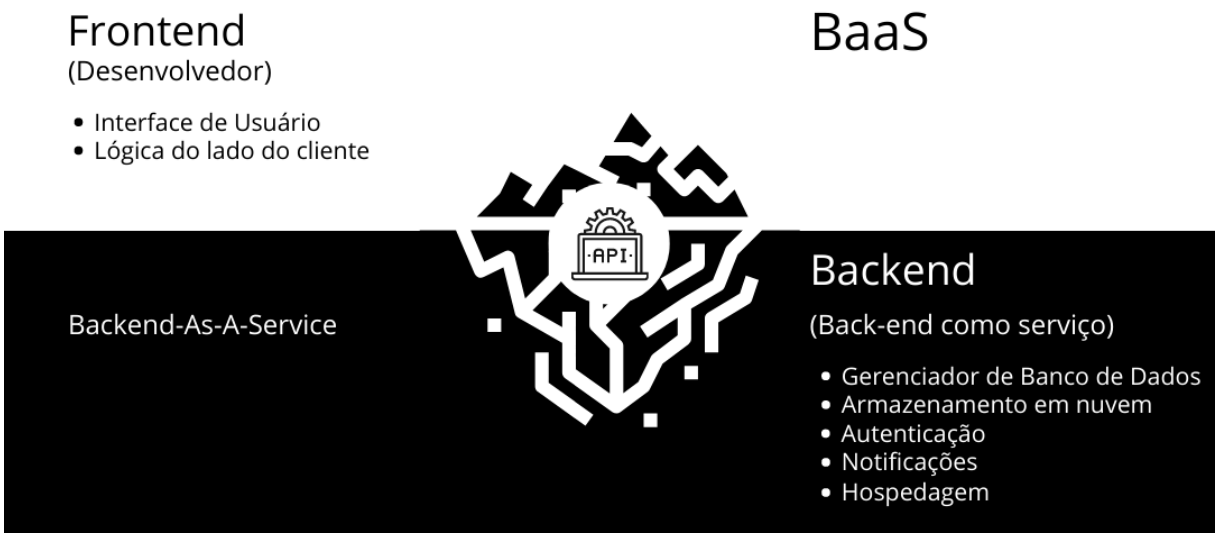

Fonte: Nadeem, (2023)

Adicionalmente, o Flutter se destaca por sua funcionalidade de *hot reload*, uma das mais inovadoras no contexto do desenvolvimento de aplicativos. Esta característica permite que os desenvolvedores visualizem as modificações realizadas no código quase que instantaneamente, sem a necessidade de reiniciar o aplicativo. O *hot reload* revoluciona o processo de desenvolvimento, proporcionando uma economia significativa de tempo e promovendo uma abordagem de experimentação e iteração mais ágil. As alterações feitas são imediatamente aplicadas e visíveis na aplicação em uso, facilitando de maneira considerável o ajuste da interface do usuário e a correção de *bugs* de forma eficiente e dinâmica (Flutter, 2023).

### *3.1.3 Firebase*

O *Firebase*, uma plataforma *Backend As A Service* (BaaS) mantida pela Google, representa uma solução inovadora e abrangente no desenvolvimento de aplicações, tanto *web* quanto móveis. Este modelo de serviço BaaS oferece uma infraestrutura e *backend* simplificados, eliminando a necessidade de desenvolvimento manual de várias soluções essenciais, como autenticação de usuário, armazenamento de dados, análise de dados e serviços de notificação (Andrade, 2021). A Figura 4 ilustra o que é um BaaS.

Figura 4 – Ilustração de um BaaS




Fonte: Andrade (2021).

A principal vantagem do *Firebase* reside em sua facilidade de uso e integração com diversas tecnologias. A plataforma dispõe de uma *Application Programming Interface* (API) intuitiva e amplamente documentada, facilitando significativamente o processo de desenvolvimento. Além disso, sua escalabilidade e confiabilidade são notáveis, permitindo o manejo eficiente de muitos usuários e garantindo uma experiência de uso estável. A arquitetura em nuvem do *Firebase* possibilita a execução de aplicativos em múltiplas plataformas, incluindo *Android*, *iOS* e *Web*, proporcionando uma abordagem versátil e integrada ao desenvolvimento de aplicações (Firebase, 2023).

Um aspecto crucial do *Firebase* é sua capacidade de oferecer recursos em tempo real. Isso significa que os aplicativos podem atualizar dados instantaneamente em todos os dispositivos conectados, facilitando a construção de aplicativos com sincronização de dados em tempo real. Esta característica é particularmente valiosa para aplicações que dependem de interações dinâmicas e atualizações contínuas, como jogos online e aplicativos de mensagens.

Além disso, o *Firebase* oferece uma gama de serviços integrados que abrangem desde o desenvolvimento até a análise de desempenho do aplicativo. Estes incluem o *Realtime Database*, que oferece um banco de dados NoSQL hospedado na nuvem, permitindo a sincronização de dados em tempo real entre os usuários. Este

serviço é essencial para aplicações que necessitam de atualizações instantâneas e interatividade, como aplicativos de mensagens instantâneas e jogos online. A estrutura de dados é organizada como um grande JSON, facilitando o armazenamento e a sincronização de dados entre os usuários de forma eficiente. Além disso, o *Realtime Database* fornece recursos robustos de segurança e regras de acesso, assegurando que os dados sejam acessíveis apenas por usuários autorizados e de maneira segura (Orlandi, 2018; Firebase, 2023).

Outro serviço importante é o *Firebase Cloud Messaging (FCM)*, que permite o envio de notificações e mensagens para os usuários de forma eficiente. Este serviço é crucial para manter os usuários engajados e informados sobre atualizações importantes, novos conteúdos ou outras comunicações essenciais. O FCM é altamente escalável, permitindo o envio de milhões de mensagens simultaneamente sem custo adicional. Além disso, oferece recursos avançados como segmentação de público, análise de desempenho de mensagens e personalização de notificações, tornando-o uma ferramenta valiosa para estratégias de marketing e comunicação (Firebase, 2023).

Por fim, o sistema de autenticação do *Firebase* proporciona uma solução completa e segura para a gestão de usuários. Ele suporta autenticação usando senhas, números de telefone, identidades de provedores populares como Google, Facebook e Twitter, e mais. Este sistema é fundamental para garantir a segurança e a privacidade dos usuários, oferecendo métodos de autenticação confiáveis e personalizáveis de acordo com as necessidades de cada aplicativo. A integração do sistema de autenticação com outros serviços do *Firebase*, como o *Realtime Database* e o FCM, permite uma experiência de usuário coesa e segura, essencial para o sucesso de qualquer aplicativo moderno (Firebase, 2023).

### *3.1.4 Google Calendar API*

A *Google Calendar API* é uma interface de programação de aplicativos desenvolvida pela Google, que se tornou uma ferramenta essencial no arsenal de desenvolvedores de *software* modernos. Esta API permite a interação direta com o *Google Calendar*, um serviço de calendário online amplamente utilizado, possibilitando a criação, modificação e exclusão de eventos de forma programática. Através dela, desenvolvedores podem integrar funcionalidades de calendário em suas

próprias aplicações, proporcionando uma experiência de usuário mais rica e personalizada (Google, 2023).

Uma das principais características do *Google Calendar API* é sua capacidade de fornecer acesso em tempo real aos eventos do calendário. Isso significa que as aplicações podem ler e escrever dados diretamente no *Google Calendar*, sincronizando eventos em tempo real. Além disso, a API suporta o envio de notificações e lembretes, facilitando a gestão de compromissos e a organização do tempo para os usuários. A integração com outros serviços do Google e aplicações de terceiros também é um ponto forte, permitindo uma maior interoperabilidade entre diferentes sistemas e plataformas (Google, 2023).

Do ponto de vista técnico, a *Google Calendar API* segue os princípios de *design* REST, utilizando protocolos de autenticação e autorização padrão, como *OAuth 2.0*, e suportando formatos de dados como JSON e XML. Esses aspectos técnicos garantem que a API seja segura, eficiente e fácil de integrar em diferentes ambientes de desenvolvimento (Google, 2023).

Apesar de suas muitas vantagens, a utilização da *Google Calendar API* não está isenta de desafios. Questões de segurança de dados são uma preocupação constante, especialmente ao lidar com informações sensíveis dos usuários. Além disso, os desenvolvedores devem estar atentos aos limites de uso impostos pela Google, para evitar interrupções no serviço. A necessidade de manter a compatibilidade com atualizações do serviço do *Google Calendar* também pode representar um desafio adicional (Google, 2023).

No contexto do aplicativo em questão, a *Google Calendar API* desempenha um papel fundamental na gestão eficiente dos eventos de calendário. Esta integração permite aos usuários realizarem uma variedade de ações, incluindo a criação, edição, exclusão e recuperação de eventos, seja de forma individual ou em conjunto. É importante destacar que a funcionalidade plena da API está disponível exclusivamente para usuários autenticados através de suas contas Google, garantindo uma experiência personalizada e segura. Essa integração com a *Google Calendar API* não apenas enriquece o aplicativo com recursos avançados de gerenciamento de tempo, mas também facilita significativamente a organização e o planejamento do usuário, contribuindo para uma experiência de uso mais eficiente e intuitiva.

## 3.2 Abordagens de Desenvolvimento

Nesta Seção, será apresentada as abordagens de desenvolvimento desde os *designs* elaborados para as interfaces do aplicativo, até os principais fluxos de trabalho que integram a inteligência artificial e a API de calendários do Google. Este segmento visa explorar como os elementos de *design* e as funcionalidades técnicas se combinam para criar uma experiência de usuário coesa e intuitiva, destacando a sinergia entre a tecnologia de inteligência artificial e as capacidades avançadas da *Google Calendar API* no contexto do aplicativo.

### 3.2.1 Figma

No desenvolvimento deste projeto, priorizamos uma abordagem centrada no usuário, focando na criação de uma interface intuitiva e uma experiência de usuário (UX) agradável. Utilizamos o Figma, uma ferramenta de *design* de interface, para iterar rapidamente os *designs* e protótipos do aplicativo. A simplicidade e usabilidade foram os pilares do processo de *design*, com ênfase em princípios como consistência, previsibilidade e facilidade de navegação. O Figma também permitiu ajustes rápidos durante o desenvolvimento, assegurando que o aplicativo atendesse às necessidades funcionais e proporcionasse uma experiência de usuário eficiente.

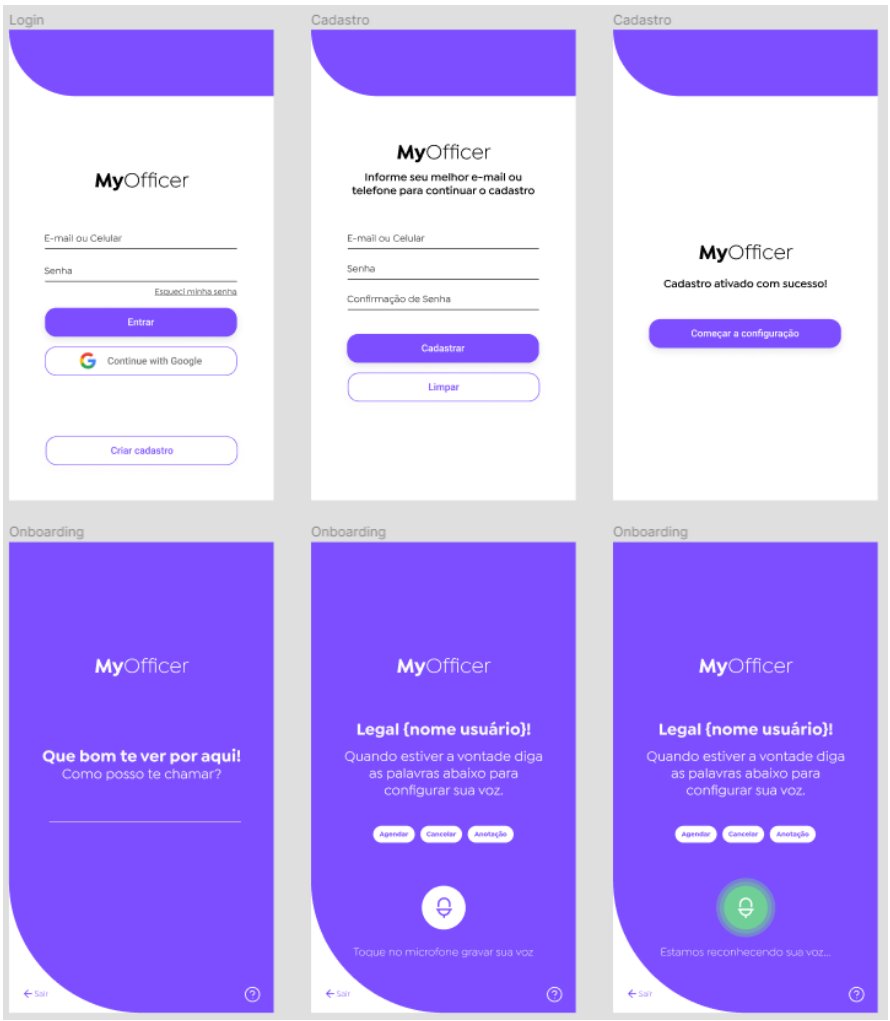
A estética visual foi cuidadosamente planejada para ser atraente e funcional, com uma paleta de cores harmoniosa, tipografia legível e uso estratégico de ícones. A navegação intuitiva do aplicativo, com um *layout* claro e botões de ação bem posicionados, visou minimizar a curva de aprendizado e melhorar a retenção do usuário. Nos próximos segmentos, apresentaremos prints dos *designs* realizados no Figma, ilustrando a evolução do *design* e como as decisões de UI/UX foram implementadas desde o início do projeto. A Figura 5 demostra as telas de *Login*, Cadastro e fluxo de *onboarding* do aplicativo.

Ao adentrarmos na análise das interfaces desenvolvidas, observamos primeiramente a tela de *Login*, que exemplifica a aplicação de nossos princípios de *design*, com um *layout* que prima pela simplicidade. A interface permite ao usuário a entrada via e-mail e senha, além de oferecer integração com o Google, o que reflete uma abordagem moderna e centrada no usuário, dando ênfase na versatilidade e na rapidez do acesso.

Em seguida, a tela de cadastro mantém a coerência visual, com campos claramente demarcados para inserção de senha e sua confirmação, minimizando erros de entrada e melhorando a eficiência do usuário, há uma opção de limpar as informações inseridas, permitindo correções fáceis e rápidas.

O processo de *onboarding* ilustra a jornada de acolhimento do usuário no primeiro uso do aplicativo. As telas são projetadas com uma comunicação amigável e direta, incentivando a interação imediata. A primeira tela de *onboarding* pede para o usuário inserir seu nome que será chamado dentro do aplicativo. Prosseguindo, a tela seguinte introduz a funcionalidade de comando de voz com uma interface clara, que destaca a opção de gravação de voz, acompanhada de instruções sucintas e um visual que destaca a ação esperada. Isso não apenas orienta o usuário, mas também encoraja a exploração das funcionalidades do aplicativo.

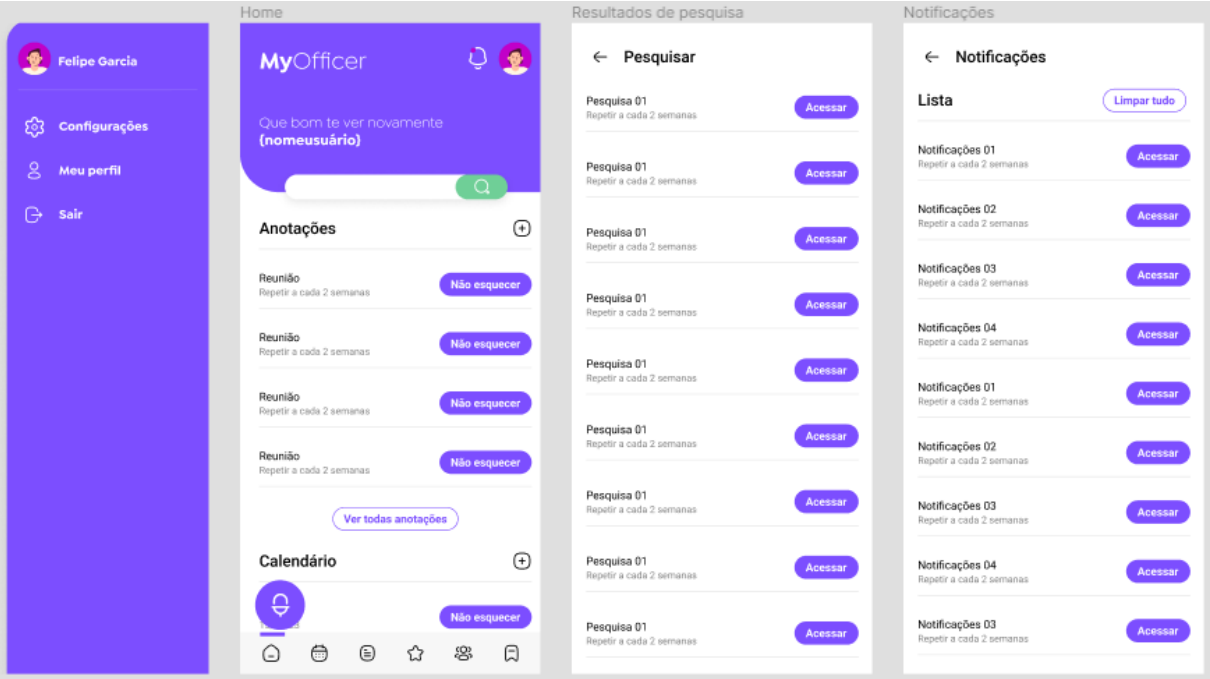
Figura 5 – Projeto do Figma da página de *Login* e Cadastro




Fonte: Autor

Após o processo de *onboarding*, o usuário é enviado diretamente para a tela da *Home*, assim como o usuário que fazer o *Login* direto no aplicativo, que são mostradas na ilustração abaixo. A Figura 6 demostra a tela principal, tela de pesquisa e notificações do aplicativo.

Figura 6 – Projeto do Figma da página de *Home*, Pesquisa e Notificações




Fonte: Autor

A tela principal (*Home*) serve como o *hub* central de navegação dentro do aplicativo *MyOfficer*, refletindo uma continuidade do *design* intuitivo. O foco no usuário é mantido com a funcionalidade de pesquisa rápida acessível, permitindo uma navegação eficiente às informações desejadas.

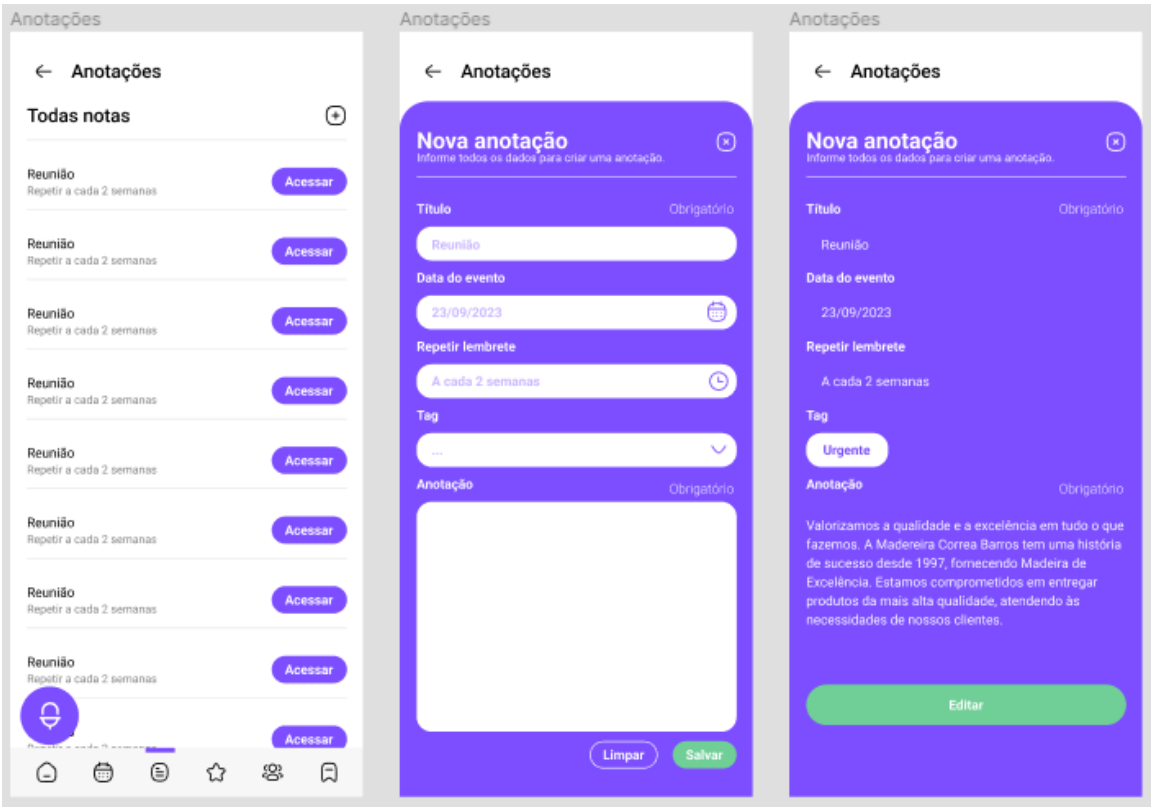
O *design* da tela *Home* é estrategicamente seccionado em áreas de interesse, como “Anotações”, ”Calendário” e mais pra baixo ainda teria “Lembretes”, são selecionados para aparecer na página, apenas as quatro anotações e lembretes mais recentes, para facilitar o manuseio do usuário, e na Seção de calendário trás todos os eventos que o usuário tem no dia, para melhor visualização. A opção de ver todas as anotações com um único toque exemplifica a facilidade de acesso e a valorização do tempo do usuário.

Nas telas de pesquisa e notificações, é observado uma lista consistente e organizada, que permite ao usuário identificar rapidamente itens individuais e interagir

com eles. A capacidade de limpar todas as notificações com um simples toque na opção na tela de notificações demonstra uma compreensão da necessidade de manter a interface do usuário livre de desordem, promovendo uma experiência mais tranquila e controlada.

O menu lateral, visível na tela da *Home*, oferece uma navegação secundária, possibilitando o acesso rápido às configurações do usuário, perfil e a opção de sair do aplicativo. A identificação do usuário, e a escolha de ícones intuitivos ao lado de cada opção, garantem que o usuário sempre saiba onde está no aplicativo e como navegar para outras seções importantes. A Figura 7 demostra o estilo de telas das funcionalidades, dando como exemplo a tela de anotações e as telas de criação e detalhamento da anotação do aplicativo.

Figura 7 – Projeto do Figma exemplo da página de Anotações, para criação e visualização




Fonte: Autor

O *design* da tela de anotações foi utilizado como base para a tela das outras funcionalidades parecidas, como os lembretes e *tags*, por isso não foi necessário fazer um *design* de cada um.

A tela de anotações apresenta uma lista de notas claramente delineada em itens, esta abordagem minimalista permite aos usuários um rápido discernimento de suas tarefas e compromissos recorrentes, assegurando que as informações críticas sejam facilmente acessíveis. Em cada item das anotações é possível ver seu título e uma parte do que será escrito dentro dela.

O processo de criação de uma nova anotação é simplificado, como visto no *design* de nova anotação, como a criação é para ser acessado de qualquer parte do aplicativo, utilizando a IA foi escolhido com que a página de criação apareça na parte inferior da tela, para ficar mais agradável. E como os *design*s foram utilizados de base para todas as funcionalidades, os campos nesse *design* não refletem exatamente os campos do aplicativo final. O *design* utiliza elementos familiares de interface do usuário, como listas suspensas e interruptores, para oferecer uma experiência de usuário consistente e intuitiva.

E a última tela de *design* representa os detalhes da anotação criada, destaca a funcionalidade de texto expansível, permitindo aos usuários inserirem informações detalhadas. E nessa tela é possível editar pelo botão que irá abrir a tela de criação, mas com os campos preenchidos para edição, reforçando a natureza dinâmica da gestão de informações no aplicativo.

### 3.2.2 Análise dos Serviços

Nesta Seção, apresentarei alguns de diagramas de sequência detalhados, ilustrando os fluxos operacionais da inteligência artificial com reconhecimento de voz no sistema. Esses diagramas oferecem uma visão clara e estruturada de como a tecnologia de reconhecimento de voz é implementada e interage dentro do ambiente do aplicativo. Além disso, fornecerei uma explanação aprofundada sobre a integração com a *Google Calendar API*, destacando os aspectos técnicos e funcionais dessa conexão e como ela enriquece a funcionalidade geral do sistema.

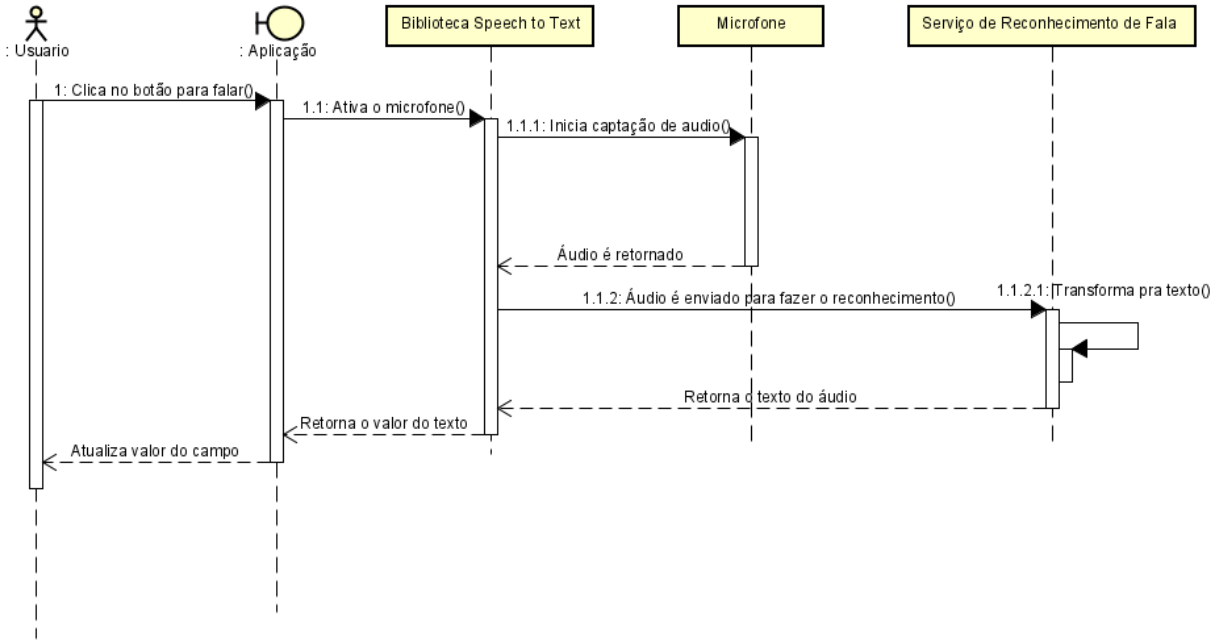
#### 3.2.2.1 Diagrama de Sequência preenchendo campo com IA

Dentro do escopo da aplicação em desenvolvimento, a automação de tarefas como a criação de anotações, eventos, lembretes e etiquetas é mediada por uma inteligência artificial avançada. Essa automação é acionada por uma interação

simplificada, na qual o usuário inicia o processo ao pressionar um botão destinado a ativar o sistema de IA. Subsequentemente, o usuário verbaliza comandos específicos que são interpretados pela aplicação para a execução da tarefa desejada. O *software*, por sua vez, promove a interação com a interface de criação, preenchendo os campos necessários de maneira autônoma. Posteriormente, o usuário tem a opção de refinar os dados inseridos utilizando comandos de voz para a seleção ou inserção de informações adicionais nos campos pertinentes.

O diagrama sequencial ilustrado abaixo detalha o processo interativo supracitado, que é fundamentado na utilização da biblioteca *Speech to Text*, uma ferramenta nativa do *framework* Flutter. A Figura 8 ilustra o diagrama de sequência de preenchimento de campo utilizando o reconhecimento de voz.

Figura 8 – Diagrama de Sequência do fluxo de preenchimento de campos utilizando IA




Fonte: Autor

Na Figura apresentada, identifica-se a participação de um ator primário e múltiplos objetos que integram o sistema. O ator, representando o usuário final, interage diretamente com o *software*, que desempenha o papel duplo de interface gráfica e *backend*, facilitando o acesso às bibliotecas subjacentes. A biblioteca *Speech to Text* foi essencial para facilitar o complicado processo de processamento da fala. Operando como um intermediário entre o aplicativo e o dispositivo de captação

de áudio, essa biblioteca gerencia o fluxo de dados sonoros, convertendo-os em texto com o auxílio de um serviço de reconhecimento de fala. Este serviço, especializado na transcrição de áudio para texto, recebe os dados captados, processa a fala e retorna o texto transcrito. O dispositivo de captação de áudio, o microfone, é responsável pela gravação das entradas de voz fornecidas pelo usuário.

No fluxo operacional descrito pelo diagrama, a interação se inicia com o usuário ativando o sistema de inteligência artificial através da interface, que aciona a biblioteca *Speech to Text* para começar a gravação de áudio. O microfone capta a voz do usuário e transmite os dados sonoros de volta para a biblioteca, que por sua vez os envia ao serviço de reconhecimento de fala. Este serviço executa a conversão da entrada de áudio em texto, o qual é imediatamente retornado à aplicação. A aplicação, recebendo o texto transcrito, procede ao preenchimento dos campos da interface gráfica, permitindo que o usuário observe em tempo real a transcrição de sua fala em texto.

#### 3.2.2.2 Diagrama de Sequência fazendo busca com IA

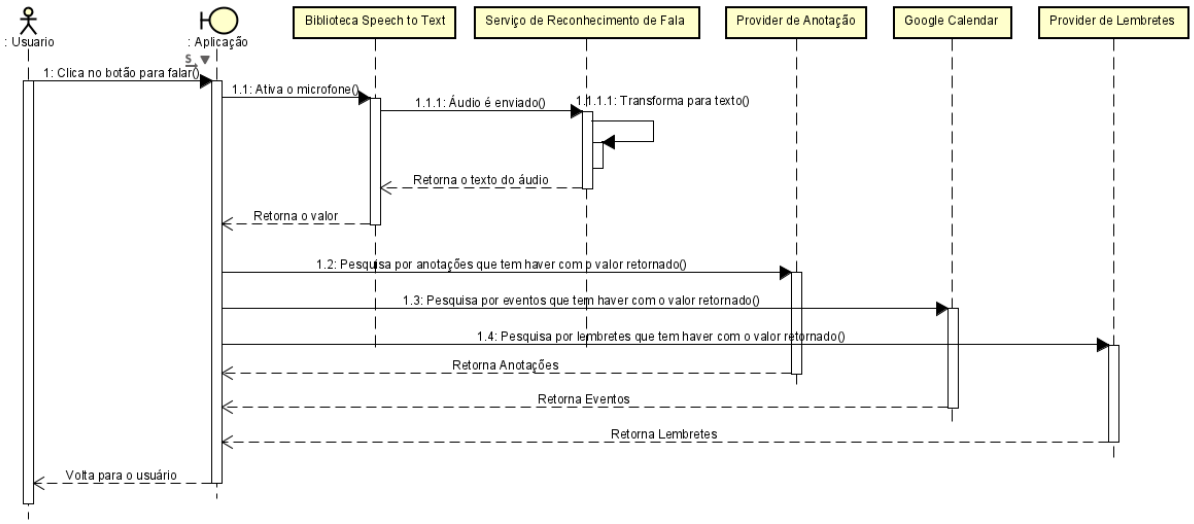
Ainda dentro do escopo que focaliza a implementação de inteligência artificial em processos de busca e recuperação de informação dentro de uma aplicação, foi desenvolvido um mecanismo robusto que capitaliza sobre as capacidades da IA para realizar buscas refinadas. Este mecanismo permite que o usuário, através de um comando de voz, inicie uma busca interativa na aplicação por anotações, eventos e lembretes pertinentes à consulta vocalizada. Este processo é intuitivo e eficiente, permitindo ao usuário simplesmente verbalizar o conteúdo desejado e, em resposta, o aplicativo efetua a recuperação dos dados correspondentes.

O diagrama subsequente proporciona uma visualização esclarecedora deste processo, delineando cada etapa necessária para a realização dessa operação de busca assistida por IA. A Figura 9 ilustra o diagrama de sequência de busca utilizando o reconhecimento de voz.

O diagrama em questão compartilha similaridades estruturais com o antecedente, apresentando o usuário como o agente iniciador da interação com a aplicação. A aplicação, por sua vez, desempenha um papel bifronte: como interface de comunicação direta com o usuário e como um sistema *backend* que invoca bibliotecas e executa operações internas, incluindo buscas em bases de dados e chamadas a APIs externas. A biblioteca *Speech to Text* é novamente empregada

como o elemento essencial para o gerenciamento da entrada de áudio e subsequente processamento em texto. Complementarmente, o serviço de reconhecimento de voz é responsável por transcrever o áudio em um formato textual.

Figura 9 – Diagrama de Sequência do fluxo de busca de itens utilizando a IA




Fonte: Autor

O diagrama em questão compartilha similaridades estruturais com o antecedente, apresentando o usuário como o agente iniciador da interação com a aplicação. A aplicação, por sua vez, desempenha um papel bifronte: como interface de comunicação direta com o usuário e como um sistema *backend* que invoca bibliotecas e executa operações internas, incluindo buscas em bases de dados e chamadas a APIs externas. A biblioteca *Speech to Text* é novamente empregada como o elemento essencial para o gerenciamento da entrada de áudio e subsequente processamento em texto. Complementarmente, o serviço de reconhecimento de voz é responsável por transcrever o áudio em um formato textual.

Distinguindo-se do fluxo previamente descrito, este diagrama incorpora novos objetos: o provider de anotação, que realiza consultas específicas para anotações dentro do banco de dados; o provider de lembretes, que faz o mesmo para lembretes; e o *Google Calendar API*, que é consultada para extrair informações sobre eventos relevantes. Estes componentes são fundamentais para a execução de buscas direcionadas e a apresentação de resultados pertinentes ao usuário.

Em relação ao fluxo de operações, o processo inicia-se com a ativação da biblioteca *Speech to Text* pelo usuário, que por sua vez ativa o microfone para captar

o comando vocal. Este áudio é transmitido ao serviço de reconhecimento de voz, onde é transcrita a fala em texto. Quando a aplicação recebe este texto, ela identifica-o como um comando de busca e procede com a apresentação da interface de pesquisa, ao mesmo tempo em que inicia a recuperação dos dados. De forma paralela, a aplicação transmite o texto transcrito aos providers de anotação e lembretes e ao *Google Calendar API*, que então filtram e retornam informações baseadas nos critérios especificados pelo usuário, como o título e a descrição dos itens buscados.

O resultado dessa busca é um conjunto refinado de dados que são então apresentados ao usuário, completando assim um ciclo interativo que destaca a eficiência e a inteligência do sistema proposto.

#### 3.2.2.3 Diagrama de Sequência criando evento com o *Google Calendar*

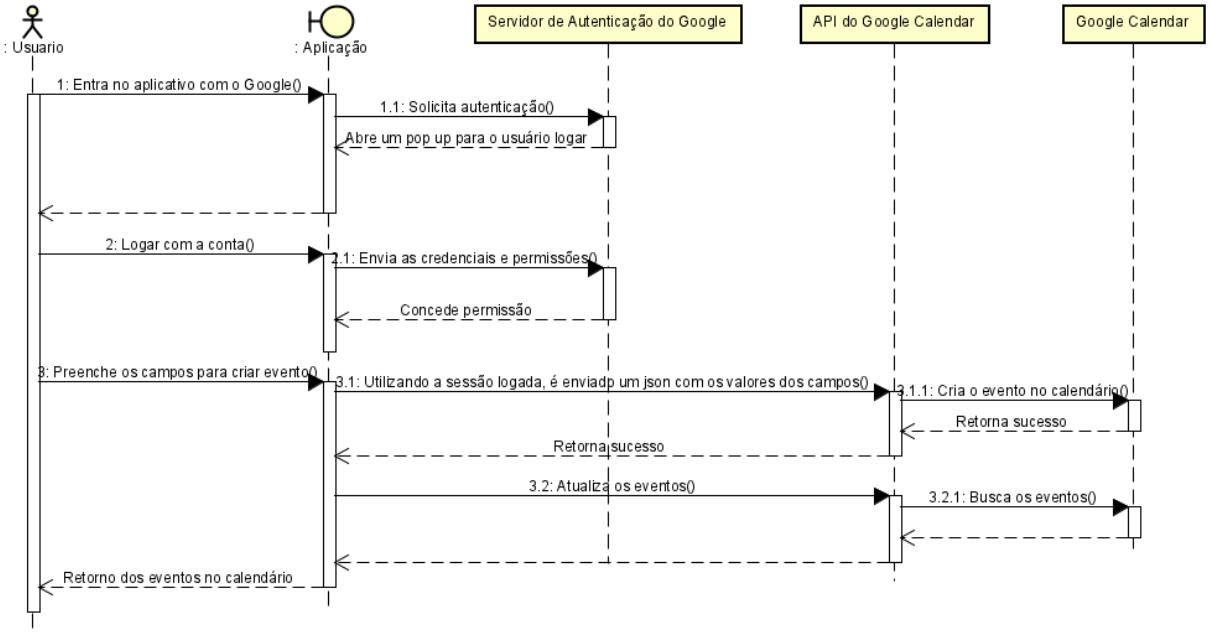
No contexto da mobilidade digital contemporânea, a aplicação desenvolvida em Flutter não somente capitaliza as vantagens intrínsecas à sua plataforma de criação, mas também se alavanca nos serviços robustos fornecidos pelo Google. Incorporar a autenticação via Google e a interação com a API do *Google Calendar* foi uma escolha estratégica, permitindo aos usuários uma maneira fluida e integrada de gerenciar eventos em seus dispositivos móveis. Através desta integração, a aplicação permite não apenas a criação e o armazenamento de eventos no calendário do usuário, mas também a busca e a organização de tais eventos, enriquecendo a funcionalidade do aplicativo e aprimorando a experiência do usuário com uma solução de gerenciamento de tempo eficaz. A Figura 10 ilustra o diagrama de sequência de gerenciamento de eventos.

O diagrama subsequente oferece uma visão granular das etapas envolvidas no processo de interação com os serviços do Google, elucidando desde o momento de autenticação até a criação e consulta de eventos no calendário do usuário. O diagrama evidencia o usuário como o protagonista dessa interação, delineando sua jornada desde o ponto de entrada no aplicativo até a realização de tarefas específicas dentro dele.

A aplicação, atuando como o nexo entre o usuário e os serviços do Google, desdobra-se tanto como interface de usuário quanto como processador de *backend*. O Servidor de Autenticação do Google é destacado como um bastião de segurança e confiabilidade, validando as credenciais do usuário e proporcionando um portal para

o acesso ao ecossistema do Google. Em particular, a API do *Google Calendar* é enfatizada pelo seu papel crucial na manipulação dos serviços de calendário, permitindo a inserção e recuperação de dados de eventos de forma transparente e eficiente.

Figura 10 – Diagrama de Sequência do fluxo de gerenciamentos de eventos utilizando o *Google Calendar API*




Fonte: Autor

Iniciando-se com a entrada do usuário no aplicativo via uma conta do Google, o diagrama mostra a sequência de eventos desencadeada pela escolha de uma conta, seguida pela autenticação e subsequente permissão concedida ao aplicativo. Uma vez autenticado, o usuário procede para a criação de um evento, preenchendo os campos necessários que o aplicativo transforma em um pacote de dados no formato JSON. Este pacote é então transmitido para a API do *Google Calendar*, que executa a ação de criar o evento no *Google Calendar* associado ao usuário. Completando o ciclo, o aplicativo recebe e apresenta os eventos atualizados no calendário do usuário, finalizando o processo com uma exibição dos eventos sincronizados e organizados de forma conveniente e acessível dentro do aplicativo.

# 4 DESENVOLVIMENTO

Neste Capítulo, exploraremos em profundidade os aspectos cruciais do desenvolvimento do aplicativo. Abordaremos a arquitetura e modelagem do banco de dados, detalhando como ele foi estruturado e integrado ao sistema. Além disso, discutiremos a implementação e aplicação da inteligência artificial com reconhecimento de voz, enfatizando sua funcionalidade e interação com os demais componentes do aplicativo. Por fim, faremos uma análise minuciosa das principais telas do aplicativo, descrevendo cada uma delas para ilustrar como contribuem para a experiência geral do usuário e a eficácia do aplicativo.

## 4.1 Modelagem do Banco de dados

Neste segmento do trabalho, exploraremos a estrutura fundamental que sustenta a funcionalidade e eficiência do aplicativo: a modelagem do banco de dados. A escolha de um banco de dados NoSQL não foi aleatória, mas sim uma decisão deliberada, alinhada com os objetivos específicos do projeto. Esta Seção detalha a modelagem do banco de dados NoSQL adotado, elucidando a estrutura de dados, a descrição de cada campo e a lógica subjacente à sua organização. Além disso, discutiremos as razões pelas quais um banco de dados NoSQL foi preferido em detrimento de outras abordagens de banco de dados, destacando como suas características únicas se alinham com as necessidades e desafios específicos enfrentados durante o desenvolvimento do aplicativo.

A modelagem de banco de dados é um componente crítico no desenvolvimento de qualquer aplicativo, pois define como os dados são armazenados, organizados e acessados. Uma modelagem bem planejada não apenas facilita a eficiência e a escalabilidade do aplicativo, mas também garante a integridade e a segurança dos dados.

A seguir, apresentaremos a estrutura do banco de dados NoSQL utilizado em formato de json, detalhando cada componente e explicando como eles se encaixam no contexto mais amplo do aplicativo. A Figura 11 demonstra a modelagem feita no banco de dados *Realtime Database*.

Figura 11 – Visão geral da modelagem do banco de dados por JSON


```
{
    "users": {
        "user-uid1": {
            "anotations": {
                "anotations-uid": {
                    "tag": "tag-uid2",
                    "text": "Escrever resenha da faculdade, mostrar aplicativo para o orientador, finalizar trabalho final",
                    "title": "Coisas para fazer",
                    "updatedAt": 1698895524793
                }
            },
            "name": "João Silva",
            "email": "joaosilva@example.com",
            "notification": {
                "notification-uid": {
                    "description": "Lembrete para me lembrar da reunião com os devs",
                    "title": "Reunião com os Devs",
                    "updatedAt": 1698895524793
                }
            },
            "reminders": {
                "reminders-uid": {
                    "description": "Lembrete para me lembrar da reunião com os devs",
                    "tag": "tag-uid1",
                    "time": "14:00",
                    "title": "Reunião com os Devs",
                    "updatedAt": 1698895524793
                }
            },
            "tags": {
                "tag-uid1": {
                    "color": 4280391411,
                    "title": "Trabalho"
                },
                "tag-uid2": {
                    "color": 4293467747,
                    "title": "Faculdade"
                }
            }
        }
    }
}
```


Fonte: Autor

A modelagem adotada no banco de dados do aplicativo foi cuidadosamente projetada, centrando-se em torno de uma chave principal chamada “*users*”. Esta chave é o ponto de partida para definir os usuários no sistema, com cada usuário sendo identificado por um '*user-uid*', uma chave única gerada automaticamente pelo *Firebase*. Sob esta chave, encontramos os dados essenciais do usuário, destacando-se dois atributos principais: “name” e “email”, que representam, respectivamente, o nome e o e-mail do usuário.

Além desses atributos fundamentais, o banco de dados abriga outros conjuntos de dados significativos: “*anotations*”, “*reminders*”, “*notifications*” e “*tags*”. No conjunto '*anotations*', as anotações feitas pelos usuários são armazenadas. Cada anotação, assim como os usuários, é diferenciada por um UID próprio, e dentro dela, encontramos seus atributos como o “*title*”, que refere ao título da anotação, *“text”* que constitui o corpo da anotação no qual o usuário irá escrever o que ele quer anotar,

*“updatedAt”* é um atributo usado principalmente para ordenar os itens na interface do usuário e “*tags*”, que categoriza a anotação, armazenando apenas o UID da tag correspondente.

O conjunto “*reminders*” é dedicado aos lembretes criados pelos usuários. Sua estrutura é bastante similar à das anotações, incluindo “*title*”, *“description”* (que nas anotações é chamado de *“text”*), *“updatedAt”* para a ordenação, e *“tag”* para a categorização, armazenando o UID específico da *tag*. Além disso, possui um novo atributo, *“time”*, que indica o momento em que o lembrete deve ser entregue ao usuário na forma de notificação.

Quanto ao conjunto “*notifications*”, este é responsável por armazenar as notificações que o sistema envia para o usuário. Estas notificações são guardadas para que o usuário possa consultá-las quando desejar. A estrutura deste conjunto é simplificada, contendo atributos para o título, descrição e a data de criação da notificação, que auxiliam na ordenação dentro do aplicativo.

Finalmente, o conjunto “*tags*” também possui uma estrutura simplificada. Cada *tag* é separada por um UID gerado automaticamente pelo *Firebase* e é utilizada nos conjuntos de anotações e lembretes. As *tags* incluem um título e um atributo “*color*”, que recebe um valor numérico representando sua cor no sistema.

É importante frisar que os eventos do calendário não são armazenados dentro do aplicativo. Todo o gerenciamento desses eventos é feito através da API do *Google Calendar*, o que sublinha a sua importância fundamental para o funcionamento eficaz do aplicativo.

A seleção de um banco de dados NoSQL para este projeto foi uma decisão estratégica, fundamentada em várias considerações técnicas. A primeira e mais significativa é a flexibilidade inerente ao NoSQL. Em um ambiente de desenvolvimento dinâmico, onde as exigências e especificações podem evoluir rapidamente, a capacidade de adaptar a estrutura de dados sem grandes complicações é um ativo valioso. Esta flexibilidade permite uma resposta ágil às mudanças, mantendo o desenvolvimento eficiente e alinhado com os objetivos do projeto.

Outro aspecto crucial é a escalabilidade oferecida pelo NoSQL, à medida que o aplicativo expande seu alcance e base de usuários, o banco de dados NoSQL facilita o gerenciamento desse crescimento. Esta escalabilidade assegura a continuidade do desempenho otimizado do aplicativo, mesmo sob carga crescente, garantindo uma experiência de usuário consistente e confiável.

Além disso, a natureza do NoSQL em lidar com uma variedade de formatos de dados é particularmente benéfica para este projeto, o aplicativo lida com uma gama diversificada de dados, desde informações de usuário até lembretes e anotações. O NoSQL proporciona uma maneira eficiente de armazenar e acessar esses dados, independentemente de sua estrutura.

Por fim, a eficiência do NoSQL em termos de velocidade de leitura e escrita de dados é um fator decisivo para a escolha desta tecnologia. Em um contexto em que a rapidez de resposta é fundamental, o desempenho superior do NoSQL em operações de banco de dados é um diferencial importante, contribuindo para a agilidade e a fluidez da experiência do usuário.

## 4.2 Inteligência Artificial e Reconhecimento de Voz

A escolha da tecnologia de inteligência artificial correta é um aspecto crucial no desenvolvimento de aplicativos, pois ela influencia diretamente a eficiência e a inovação do produto. No início deste projeto, várias opções de IA para reconhecimento de voz foram consideradas, incluindo a *Cloud Speech-to-Text API* da Google e a *Whisper API* da OpenAI. No entanto, ambas as soluções eram baseadas em um modelo de pagamento por uso, o que as tornava menos viáveis para um projeto acadêmico devido ao custo associado a cada requisição.

Alternativas mais acessíveis foram então exploradas, focando em soluções compatíveis com Flutter. Entre elas, destacam-se o *Picovoice*, uma plataforma versátil de IA e reconhecimento de voz, e o *Alan AI*, especializado em integrações de comando de voz. Além destes, considerou-se também a biblioteca *Speech to Text*, um *plugin* que facilita a conversão de fala em texto, aproveitando os recursos de reconhecimento de voz do dispositivo para transcrever a fala do usuário em tempo real.

A decisão final recaiu sobre a biblioteca *Speech to Text*, principalmente devido à sua capacidade de transcrever áudio em português para texto sem custos adicionais. Esta característica era essencial para o projeto, visto que as outras plataformas disponíveis na época suportavam apenas o reconhecimento de voz em inglês. A escolha dessa biblioteca refletiu a necessidade de uma solução eficaz e econômica, alinhada com os objetivos e limitações do projeto acadêmico.

A seguir, apresenta-se a implementação da biblioteca *Speech to Text* em um

aplicativo, exemplificando o emprego da tecnologia de reconhecimento de voz assistido por inteligência artificial no contexto do desenvolvimento de *software*. O fragmento de código em exame ilustra a aplicação prática desta tecnologia.

Figura 12 – Visão geral do código utilizado para a IA reconhecer e preencher os dados em campos do formulário

```
child: GestureDetector(
  onTapDown: (detail) async {
    if (!isListening) {
      var avaliable = await speechToText.initialize();
      if (avaliable) {
        modalSetState(() => isListening = true);
        speechToText.listen(
          localeId: 'pt_BR',
          onResult: (val) => modalSetState(() {
            String result = val.recognizedWords;
            if (result.toLowerCase().contains('título')) {
              campo = 'título';
              valor = result.replaceFirst('título', '').trim();
            } else if (result.toLowerCase().contains('anotação')) {
              campo = 'anotação';
              valor = result.replaceFirst('anotação', '').trim();
            }else if (result.toLowerCase().contains('tag')) {
              campo = 'tag';
              valor = result.replaceFirst('tag ', '').trim();
              _selectedTag = listTag.firstWhere(
                    (tag) => valor.toLowerCase().contains(tag.title.toLowerCase()),
              );
            }else if (result.toLowerCase().contains('salvar')) {
              if (_formKey.currentState!.validate()) {
                if (anotation == null) {
                  _createAnotation(controllerTitulo.text,
                      controllerTexto.text, _selectedTag);
                } else {
                  _editAnotation(controllerTitulo.text,
                      controllerTexto.text, _selectedTag, anotation);
                  Navigator.pop(context);
                }
              }
            }

            if (campo == 'título') {
              controllerTitulo.text = valor;
            } else if (campo == 'anotação') {
              controllerTexto.text = valor;
            }
          }),
```

Fonte: Autor

Na Figura 12, observamos o esqueleto geral do código, que se vale da inteligência artificial para preencher campos de um formulário, tanto para criação quanto para edição de entradas. O código emana de um *GestureDetector Widget*, que

responde ao evento *onTapDown*. Este evento é ativado quando o usuário toca na tela, instigando a biblioteca a iniciar o reconhecimento vocal. A ativação se dá após a inicialização bem-sucedida da biblioteca *Speech to Text*, permitindo que o objeto *SpeechToText* inicie a captação auditiva das falas do usuário.

Figura 13 – Código para ativar a IA e reconhecer os campos

```
speechToText.listen(
  localeId: 'pt_BR',
  onResult: (val) => modalSetState(() {
    String result = val.recognizedWords;
    if (result.toLowerCase().contains('título')) {
      campo = 'título';
      valor = result.replaceFirst('título', '').trim();
    } else if (result.toLowerCase().contains('anotação')) {
      campo = 'anotação';
      valor = result.replaceFirst('anotação', '').trim();
    }else if (result.toLowerCase().contains('tag')) {
      campo = 'tag';
      valor = result.replaceFirst('tag ', '').trim();
      _selectedTag = listTag.firstWhere(
            (tag) => valor.toLowerCase().contains(tag.title.toLowerCase()),
      );
```

Fonte: Autor

Prosseguindo, a Figura 13 demonstra a ativação da IA e reconhecimento dos campos para preenchimento dos valores, nela configuramos a biblioteca para reconhecer e interpretar áudio em português brasileiro ('pt_BR'). Posteriormente, adentramos a função que detém a lógica para identificar comandos específicos. Nota-se que cada condicional if corresponde a um comando que determina o campo a ser preenchido. Por exemplo, se o início da fala do usuário contém a palavra "título", o sistema compreende a intenção de preencher o campo de título, capturando o texto subsequente como valor para ele. Este mecanismo é replicado para os demais campos no contexto de criação de uma anotação.

Figura 14 – Código para salvamento dos dados para criação ou edição de anotações

```
}else if (result.toLowerCase().contains('salvar')) {
  if (_formKey.currentState!.validate()) {
    if (anotation == null) {
      _createAnotation(controllerTitulo.text,
          controllerTexto.text, _selectedTag);
    } else {
      _editAnotation(controllerTitulo.text,
          controllerTexto.text, _selectedTag, anotation);
      Navigator.pop(context);
    }
  }
}
```

Fonte: Autor

A Figura 14 demonstra a funcionalidade de salvamento: o sistema procede com a validação dos campos preenchidos. Conforme essa validação, ele executa a criação ou edição de uma anotação, dependendo da existência prévia da entrada no sistema.

Figura 15 – Código para aplicar no campo o valor reconhecido

```
if (campo == 'título') {
  controllerTitulo.text = valor;
} else if (campo == 'anotação') {
  controllerTexto.text = valor;
}
```

Fonte: Autor

A Figura 15 o processo de atualização em tempo real do conteúdo do campo, conforme o usuário vocaliza as entradas. Esta ação imediata fortalece a interatividade da aplicação, refletindo as alterações diretamente na interface do usuário.

Figura 16 – Código para desativar a IA

```
onTapUp: (detail) {
  modalSetState(() => isListening = false);
  speechToText.stop();
},
```

Fonte: Autor

A Figura 16 demonstra a desativação do processo de reconhecimento de voz, quando o usuário cessa a interação com o botão de ativação, a biblioteca é invocada para cessar o reconhecimento vocal, concluindo o processo de transcrição pela inteligência artificial.

## 4.3 Definição das telas do produto

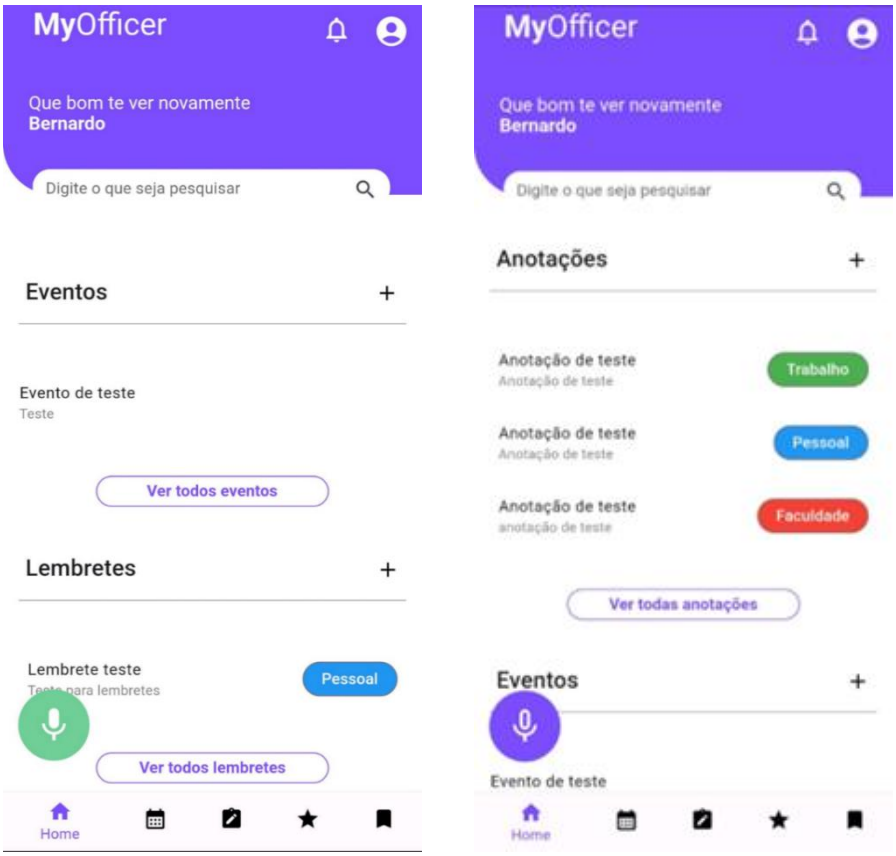
Neste segmento, apresentamos as telas chave do aplicativo, ilustrando o resultado pós-desenvolvimento e explicando o propósito e a função de cada uma delas.

### 4.3.1 *Home*

A tela inicial do aplicativo é essencial para definir o tom da experiência do usuário. Nas imagens abaixo, será mostrado o *design* e então detalhado a funcionalidade da tela *Home*, evidenciando como ela serve como portal para as principais funcionalidades do aplicativo.

A Figura 17 demonstra a tela principal do aplicativo, ela é a porta de entrada do aplicativo, projetada para proporcionar ao usuário uma visão geral imediata e acessar facilmente suas notas e lembretes mais recentes. Neste *hub* central, o ícone do sino permite visualizar notificações, enquanto o ícone do perfil abre a barra lateral para mais opções. A funcionalidade de busca é intuitivamente incorporada, permitindo filtros rápidos para localizar itens específicos, essa tela será melhor detalhada na próxima Seção. Além disso, a tela destaca as últimas atividades do usuário, como notas atualizadas e eventos do dia, com botões convenientes para acessar listas completas ou adicionar novos itens.

Figura 17 – Tela da *Home* do aplicativo




Fonte: Autor

### 4.3.2 Filtro de busca

A Figura 18 ilustra a tela de pesquisa do aplicativo que é projetada para eficiência, permitindo que o usuário localize rapidamente o que necessita. Ao inserir um termo no campo de busca da tela *Home* ou pedir para a IA em qualquer outra tela, o sistema aplica um filtro e exibe os resultados relevantes, incluindo anotações, eventos e lembretes relacionados. Se nenhum item corresponder ao critério de busca, a seção correspondente não será exibida, evitando assim a desordem visual. Por exemplo, na busca pela palavra "teste", a tela apresenta apenas os itens associados a esse termo, demonstrando a precisão e a relevância do sistema de filtragem.

Figura 18 – Tela de pesquisa do aplicativo

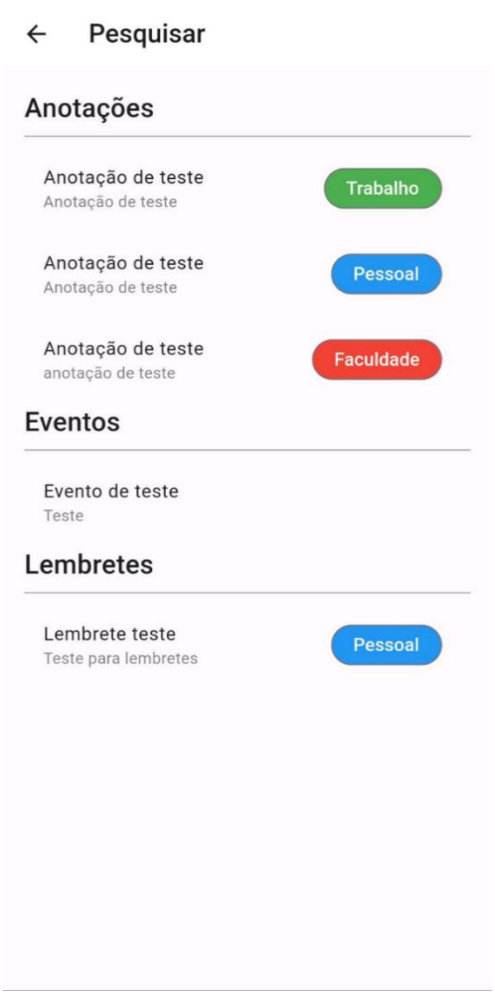
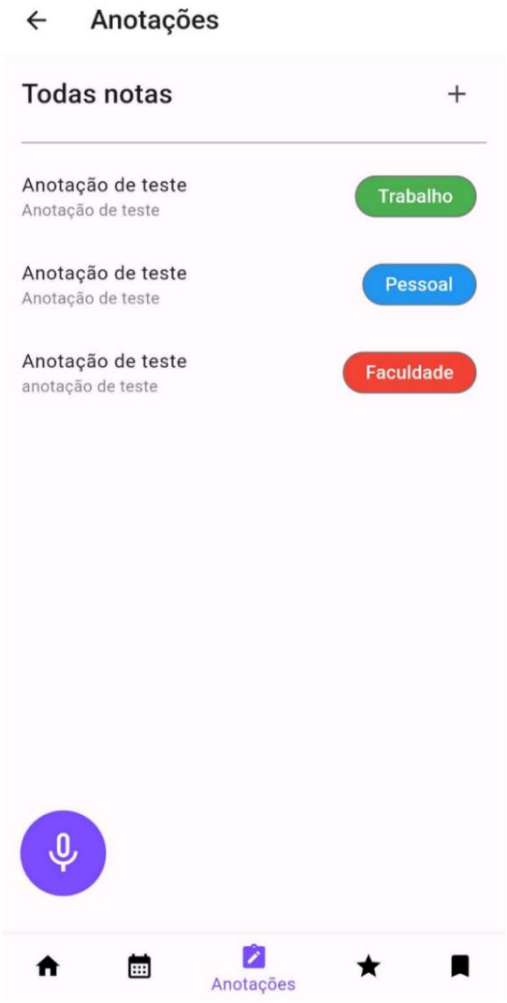



Figura 19 – Tela de anotações do aplicativo




Fonte: Autor

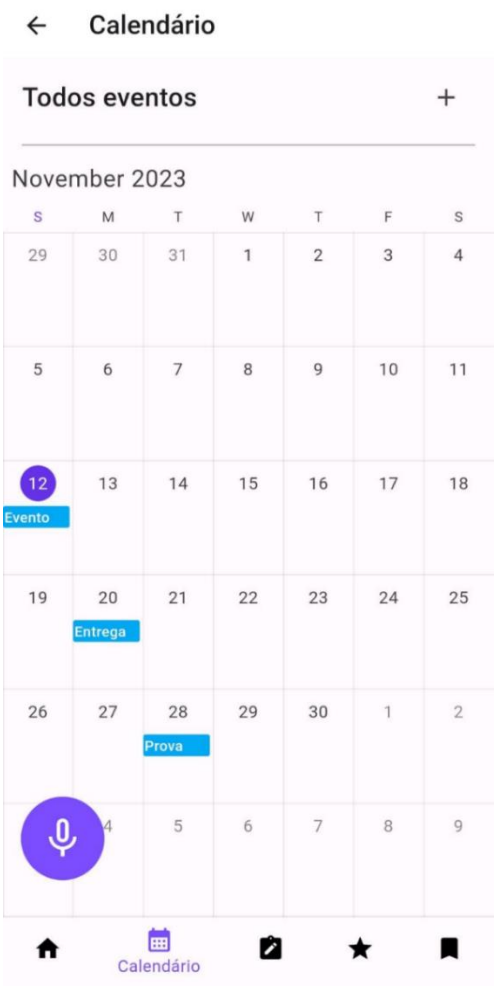
### 4.3.3 Anotações

A Figura 19 representa a tela em de anotações do aplicativo que é dedicada ao gerenciamento das notas do usuário, listando todas as que foram criadas. Com a facilidade de adicionar novas anotações utilizando o mesmo ícone encontrado na página '*Home*', a interface se mantém familiar e intuitiva. Selecionar uma nota específica revela detalhes completos, permitindo uma revisão aprofundada e a edição de cada entrada individualmente. Esta funcionalidade centraliza a organização das informações do usuário em um local conveniente e acessível.

Outras seções do aplicativo, como a de lembretes, seguem o mesmo esquema intuitivo e mantêm a arquitetura coesa da página, assegurando uma experiência de usuário consistente e fluída em todo o aplicativo.

### 4.3.4 Calendário

Figura 20 – Tela do caledário do aplicativo




Fonte: Autor

A Figura 20 ilustra uma tela que exibe um calendário interativo, centrado no gerenciamento dos eventos do usuário. Cada compromisso é visível na data correspondente, facilitando a visualização e o planejamento, a tela também destaca o dia atual, e um simples toque em qualquer data permite visualizar e gerenciar eventos específicos, de maneira similar à funcionalidade encontrada na tela de anotações.

### 4.3.5 Criação

Figura 21 – Tela de criação de anotação do aplicativo


Fonte: Autor

A Figura 21 demonstra o processo de cadastro para a criação de uma nova anotação. Conforme mencionado anteriormente, foi optado por uma modal que surge na parte inferior da tela, ao invés de uma página inteira, para manter a fluidez da navegação no aplicativo. Esta escolha de *design* se alinha com a funcionalidade de acessar a tela de cadastro por meio da assistente virtual integrada, que pode ser acionada em qualquer ponto do aplicativo.

Os campos para entrada de dados são simplificados, há o campo para o título da anotação, o campo de *tags* que permite ao usuário categorizar suas anotações com *tags* previamente definidas, e o espaço para o corpo da anotação, onde o usuário pode registrar livremente suas informações. Todos estes campos são otimizados para o uso com a assistente virtual de reconhecimento de voz, o usuário simplesmente

nomeia o campo e dita o conteúdo desejado.

A tela de criação para outros itens no aplicativo, como eventos e lembretes, segue uma estrutura semelhante, adaptando-se para incluir campos adicionais específicos, como data e hora, sempre mantendo a coerência e a facilidade de uso graças à possibilidade de interação por voz.

### 4.3.6 Detalhes do item

Figura 22 – Tela de detalhes da anotação do aplicativo


Fonte: Autor

A Figura 22 ilustra a visualização detalhada de uma anotação após sua criação. Assim como o cadastro, os detalhes são apresentados em um modal que emerge na parte inferior da tela, proporcionando uma visão completa e organizada das informações registradas, como a *tag* de categorização, o título e o conteúdo redigido pelo usuário. Esta tela também inclui um recurso prático, o botão para ativar o modo de edição, permitindo ao usuário alterar qualquer aspecto da anotação previamente inserida.

# 5 MELHORIAS E TRABALHOS FUTUROS

No desenvolvimento contínuo do aplicativo, foi pensado uma série de melhorias e novas funcionalidades que prometem enriquecer a experiência do usuário e expandir as capacidades da plataforma. Uma das adições mais significativas seria a implementação de grupos para compartilhamento de anotações, eventos e lembretes. Esta funcionalidade permitiria aos usuários criar grupos onde qualquer item adicionado seria acessível a todos os membros, facilitando a colaboração e o compartilhamento de informações em ambientes como grupos de estudo ou equipes de trabalho, essa abordagem coletiva transformaria o aplicativo em uma ferramenta mais interativa e colaborativa, ampliando seu uso para além de um simples organizador pessoal.

Além disso, o aprimoramento da inteligência artificial é um aspecto crucial para o futuro do aplicativo. Atualmente, utilizamos uma solução de IA gratuita para reconhecimento de voz, mas explorar serviços pagos ou desenvolver um modelo próprio de IA poderia oferecer avanços significativos. Um modelo personalizado de IA, treinado especificamente para as necessidades do aplicativo, poderia melhorar a precisão do reconhecimento de voz e expandir a gama de comandos disponíveis, tornando a interação com o aplicativo mais fluida e natural.

Outra melhoria importante seria a possibilidade de adicionar mais de uma *tag* para anotações, lembretes e eventos, ajudaria na organização e na busca de conteúdo dentro do aplicativo. Isso permitiria uma categorização mais detalhada e personalizada, melhorando a experiência de gerenciamento de informações dos usuários.

Por fim, a integração com serviços populares de mensagens, como o *WhatsApp*, para enviar mensagens usando a inteligência artificial, representaria um avanço significativo na interatividade do aplicativo, essa funcionalidade permitiria aos usuários enviar mensagens diretamente do aplicativo, utilizando comandos de voz, o que tornaria a comunicação mais rápida e eficiente. Essas melhorias e adições, ao serem implementadas, não apenas aumentariam a utilidade do aplicativo, mas também reforçaram seu papel como uma ferramenta essencial no dia a dia dos usuários.

# 6 CONCLUSÃO

O objetivo central deste projeto foi desenvolver um aplicativo de gerenciamento de tarefas que não apenas facilitasse a organização pessoal dos usuários, mas também inovasse na interação por meio de uma inteligência artificial assistente. Este aplicativo foi concebido para ser uma solução abrangente para o planejamento diário, oferecendo funcionalidades como criação e gerenciamento de lembretes, anotações e eventos, todos integrados com a eficiência e a conveniência da tecnologia de reconhecimento de voz. A integração com a *Google Calendar API* adicionou uma camada adicional de funcionalidade, tornando o aplicativo uma ferramenta poderosa para a gestão do tempo e compromissos.

Através deste trabalho, exploramos profundamente o potencial das tecnologias de banco de dados NoSQL, inteligência artificial para reconhecimento de voz, e a integração com APIs poderosas como a do *Google Calendar*. Cada escolha tecnológica e cada linha de código foram passos em direção à realização de um aplicativo que não só atende às necessidades de organização pessoal dos usuários, mas também oferece uma experiência interativa e intuitiva.

Este projeto também serviu como um campo de testes para práticas de *design* de interface do usuário e experiência do usuário (UI/UX), enfatizando a importância de uma abordagem centrada no usuário. A utilização do Figma no processo de *design* permitiu uma experimentação e iteração rápidas, resultando em uma interface que é tanto esteticamente agradável quanto funcional.

Em conclusão, este trabalho de conclusão de curso não é apenas o fim de uma etapa acadêmica, mas também um marco significativo em uma jornada contínua de aprendizado e inovação. O desenvolvimento deste aplicativo de gerenciamento de tarefas com integração de inteligência artificial representa um passo importante na exploração das possibilidades que a tecnologia moderna oferece. Ele estabelece uma base sólida para futuras explorações e desenvolvimentos no campo do *software*, abrindo caminho para novas ideias e aprimoramentos. Este projeto é um testemunho do poder da educação e da pesquisa em transformar conceitos em realidades tangíveis, beneficiando não apenas o desenvolvedor, mas também a comunidade mais ampla.

# REFERÊNCIAS

AMERSHI, S. *et al*. ***Guidelines for Human-AI Interaction*****.** Disponível em: <https://www.microsoft.com/en-us/research/publication/guidelines-for-human-ai-interaction/>.

ANDRADE, A. P. **O que é Firebase?** Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-firebase>.

BRACHA, G. ***The Dart Programming Language*****.** [S. l.]: Addison-Wesley Professional, 2015.

BUENO, C. E. O. **Desenvolvimento de um Aplicativo Utilizando o *Framework* Flutter e Arquitetura Limpa.** Disponível em: <https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/1861/1/TCC%202%20-%20CARLOS.pdf>.

CARVALHO NETO, F. *et al*. **Aplicações Críticas Habilitadas pela Tecnologia 5G: Oportunidades, Tendências e Desafios.** Disponível em: <https://sol.sbc.org.br/livros/index.php/sbc/catalog/view/120/535/820-1>.

CASTILLO LÓPEZ, M. I. *et al.* ***Applying User Experience and User-Centered Design Software Processes in Undergraduate Mobile Application Development Teaching***. Disponível em: <https://ijcionline.com/paper/12/12523ijci10.pdf>.

CHAN, W. *et al*. ***Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition*****.** Disponível em: <https://research.google/pubs/pub44926/>.

CHARNIAK, E.; MCDERMOTT, D. ***Introduction to Artificial Intelligence*****.** [S. l.]: Springer, 2018.

DAVIS, K. H.; BIDDULPH, R.; BALASHEK, S. ***Automatic Recognition of Spoken Digits*****.** Disponível em: <https://pubs.aip.org/asa/jasa/article-abstract/24/6/637/618458/Automatic-Recognition-of-Spoken-Digits?redirectedFrom=fulltext>.

FLUTTER. **Documentação oficial do Flutter**. Disponível em: <https://docs.flutter.dev/resources/faq>.

GONZALEZ, M.; LIMA, V. L. S. **Recuperação de Informação e Processamento da Linguagem Natural.** Disponível em: <https://www.academia.edu/download/43022678/minicurso-jaia2003.pdf>.

GOOGLE. ***Dart Programming Language***. Disponível em: <https://dart.dev>.

GOOGLE. **Documentação oficial do *Firebase*.** Disponível em: <https://firebase.google.com/docs?hl=pt-br>.

GOOGLE. **Documentação oficial do *Google Calendar API*.** Disponível em: <https://developers.google.com/calendar?hl=pt-br>.

GOOGLE. ***Firebase Authentication*.** Disponível em: <https://firebase.google.com/docs/auth?hl=pt-br>.

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

GOOGLE. ***Firebase Realtime Database*.** Disponível em: <https://firebase.google.com/docs/database?hl=pt-br>.

HAUGELAND, J. ***Artificial Intelligence: The Very Idea*.** [S. l.]: Bradford Book, 1985.

HINTON, G. *et al*. ***Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups*.** Disponível em: <https://ieeexplore.ieee.org/document/6296526>.

JADCZYK, T. *et al*. ***Artificial Intelligence Can Improve Patient Management at the Time of a Pandemic: The Role of Voice Technology***. Disponível em: <https://www.jmir.org/2021/5/e22959>.

KUMARI, V. *et al*. ***Scalability and Sustainability in Chatbot and Mobile Application Development*.** Disponível em: <https://ieeexplore.ieee.org/document/10048882>.

KURZWEIL, R. ***The Singularity is Near: When Humans Transcend Biology*.** [S. l.]: Penguin Books, 2005.

MAINKAR, P.; GIORDANO, S. ***Google Flutter Mobile Development Quick Start Guide*.** [S. l.]: Packt Publishing, 2019.

NADEEM, M. ***What is AOT vs. JIT compilation, and how it works?*.** Disponível em: <https://medium.com/taager-tech-blog/angular-what-is-aot-vs-jit-compilation-and-how-it-works-a52c81bc58f>.

NOVIA, D.; MATAHARI, T.; SETIAWAN, K. ***The Design Studies of User Interface and User Experience on the Mobile Application of Cimahi Mall***. Disponível em: <https://journal.untar.ac.id/index.php/ijassh/article/view/26389>.

ORLANDI, C. ***Firebase*: serviços, vantagens, quando utilizar e integrações.** Disponível em: <https://blog.rocketseat.com.br/firebase/>.

PINTO, S. C. S. **Processamento de Linguagem Natural e Extração de Conhecimento.** Disponível em: <https://estudogeral.uc.pt/bitstream/10316/35676/1/Processamento%20de%20Linguagem%20Natural%20e%20Extracao%20de%20Conhecimento.pdf>.

POOLE, D.; MACKWORTH, A.; GOEBEL, R. ***Computational Intelligence: A Logical Approach*.** [S. l.]: Oxford University Press, 1998.

RABINER, L.; JUANG, B. H. ***Fundamentals of Speech Recognition*.** [S. l.]: Pearson College Div, 1993.

RUSSELL, S.; NORVIG, P. ***Artificial Intelligence: A Modern Approach*.** [S. l.]: Pearson, 2016.

SANTOS, D. **Introdução ao processamento de linguagem natural através das aplicações.** Disponível em: <https://www.linguateca.pt/Diana/download/Santos2001Aplicacoes.pdf>.

SAON, G. *et al*. ***English Conversational Telephone Speech Recognition by Humans and Machines*.** Disponível em: <https://arxiv.org/abs/1703.02136>.

TOBING, G. B. R. L. *et al.* ***The Ietmpact of Sales Promotion, User Interface and User Experience Design on Shopee App Users' Repurchase Intentions***. Disponível em: <https://ojs.lib.unideb.hu/IJEMS/article/view/12424>.

VICHARE, S. *et al*. ***Intelligent Application of Voice Recognition and Personal Secretariat for Effective Business Management*.** Disponível em: <https://ieeexplore.ieee.org/document/10180496>.

VIEIRA, R. **Terminologia textual e Linguística de corpus.** Disponível em: <https://www.academia.edu/download/50033978/linguagensespecializadasemcorpora.pdf#page=184>.

VIEIRALVES, L. E. S. **Análise de Linguagem Multiplataforma com foco em Flutter.** Disponível em: <https://tconline.feevale.br/tc/files/0001_5195.pdf>.

WINDMILL, E. **Flutter in Action.** [S. l.]: Manning, 2020.

YAO, Q. ***The Application of Artificial Intelligence Voice Recognition on Helping Elders Use Mobile Phones More Easily*.** Disponível em: <https://iopscience.iop.org/article/10.1088/1742-6596/1824/1/012007>.