UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
TRABALHO DE CONCLUSÃO DE CURSO

Alex Paulo Ferreira Damascena

Desenvolvimento de Framework iOS para aplicações controladas por Eye Tracking

Recife

2025

Alex Paulo Ferreira Damascena

Desenvolvimento de Framework iOS para aplicações controladas por Eye Tracking

Trabalho apresentado ao curso de Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Área de Concentração: Ciência da Computação

Orientador: Kiev Santos da Gama

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Ferreira Damascena, Alex Paulo.
Desenvolvimento de framework ios para aplicações controladas por eye tracking / Alex Paulo Ferreira Damascena. - Recife, 2025.
67p : il., tab.

Orientador(a): Kiev Santos da Gama
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2025.
Inclui apêndices, anexos.

1. Interação Humano-Computador. 2. Framework iOS. 3. Eye Tracking. I. Gama, Kiev Santos da. (Orientação). II. Título.

000 CDD (22.ed.)

ALEX PAULO FERREIRA DAMASCENA

Desenvolvimento de framework iOS para aplicações controladas por Eye Tracking

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em 04/04/2025 da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Aprovado em: 04/04/2025

BANCA EXAMINADORA

_____

Prof. Dr. Kiev Santos da Gama (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dra. Fernanda Madeiral Delfim (Examinador Interno)

Universidade Federal de Pernambuco

_____

Prof. Dr. Kiev Santos da Gama (Examinador Interno)

Universidade Federal de Pernambuco

AGRADECIMENTOS

A realização deste Trabalho de Conclusão de Curso só foi possível graças ao apoio e à dedicação de muitas pessoas, às quais gostaria de expressar minha mais sincera gratidão.

Primeiramente, agradeço à minha família, por estar sempre ao meu lado, oferecendo apoio incondicional, amor e compreensão ao longo de toda a minha trajetória acadêmica. Cada palavra de incentivo, gesto de carinho e exemplo de perseverança foram fundamentais para que eu pudesse superar os desafios e conquistar este momento tão especial. Um agradecimento especial à minha irmã, Isabelly Damascena, cuja trajetória de vida e determinação foram uma fonte constante de inspiração para mim. Seu exemplo mostrou-me que, com esforço e resiliência, é possível alcançar nossos objetivos, independentemente dos obstáculos.

Expresso também minha profunda gratidão aos meus amigos e colegas de curso, que tornaram essa caminhada acadêmica mais leve, enriquecedora e inesquecível. Compartilhamos momentos de aprendizado, superação e alegria que ficarão para sempre na memória. Em especial, dedico um agradecimento à minha amiga Sofia Melo de Lucena, cuja parceria ao longo desses cinco anos foi indispensável.

Agradeço, ainda, aos professores do curso, que desempenharam um papel crucial na minha formação, transmitindo conhecimento e incentivando o pensamento crítico. Sua dedicação e paciência foram indispensáveis para o meu crescimento acadêmico e profissional. Em especial, gostaria de mencionar meu orientador, Kiev Santos da Gama, por sua orientação, apoio e disponibilidade durante todas as etapas deste trabalho. Suas valiosas contribuições e ensinamentos foram essenciais para a conclusão deste projeto.

A todos vocês, deixo meu mais sincero agradecimento e a certeza de que cada um teve um papel especial nesta importante etapa da minha vida.

RESUMO

A Interação Humano-Computador é um campo de cada vez mais destaque em pesquisas na área da computação. Ela abrange uma ampla variedade de algoritmos e técnicas projetadas para melhorar a forma como humanos e dispositivos interagem. Entre essas inovações, o eye tracking, tecnologia de rastreamento ocular, vem ganhando destaque em inúmeros ramos da ciência, da sociologia à medicina. A partir de sensores, câmeras ou algoritmos de rastreamento facial, o eye tracking permite registrar e analisar os movimentos oculares, fornecendo informações como direção do olhar, tempo de fixação em uma área e detecção do momento exato que o globo ocular é fechado. No contexto de dispositivos móveis, o eye tracking permite a expansão da interação homem-máquina através do controle de interfaces sem a necessidade do toque, viabilizando interações como a navegação por gestos oculares. Entretanto, ainda existe uma carência de ferramentas que auxiliem a construção de aplicações controladas por eye tracking, onde a interface é adaptada e inclusiva. Este estudo aborda o desenvolvimento e a validação de um framework iOS para a criação de aplicações baseadas em eye tracking. O framework integra o rastreamento ocular à interface por meio de uma navegação baseada em áreas de interesse, utilizando tecnologias e algoritmos como ARKit, árvore de visualização e Depth First Search. A partir da condução de entrevistas semiestruturadas e de múltipla escolha, desafios foram evidenciados na interação com a ferramenta e sua documentação os quais foram fundamentais para detectar melhorias.

Palavras-chaves: Interação Humano-Computador, Eye Tracking, Framework iOS.

ABSTRACT

Human-Computer Interaction is an increasingly prominent field in computing research. It encompasses a wide range of algorithms and techniques designed to enhance the way humans and devices interact. Among these innovations, eye tracking technology has been gaining attention in various fields of science, from sociology to medicine. Using sensors, cameras, or facial tracking algorithms, eye tracking enables the recording and analysis of eye movements, providing information such as gaze direction, fixation time on a specific area, and detection of the exact moment the eyeball is closed. In the context of mobile devices, eye tracking expands human-machine interaction by enabling interface control without the need for touch, allowing interactions such as gaze-based navigation. However, there is still a lack of tools that support the development of eye-tracking-controlled applications with adaptive and inclusive interfaces. This study focuses on the development and validation of an iOS framework for creating eye-tracking-based applications. The framework integrates eye tracking into the interface through gaze-based navigation using technologies and algorithms such as ARKit, view tree, and Depth First Search. Through semi-structured and multiple-choice interviews, challenges in interacting with the tool and its documentation were identified, which were essential in determining areas for improvement.

Keywords: Human-Computer Interaction, Eye Tracking, iOS Framework.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADI	Área de Interesse
API	Application Programming Interface
AV	Árvore de Visualização
DFS	Depth First Search
ET	Eye Tracking
IHC	Interação Humano-Computador
PSSUQ	Post-Study System Usability
RO	Rastreadores Oculares
TA	Tecnologia Assistiva
WCAG	Web Content Accessibility Guidelines

SUMÁRIO

1 INTRODUÇÃO

A acessibilidade digital tem se tornado um tema cada vez mais relevante no desenvolvimento de tecnologias, impulsionando a criação de soluções inovadoras para ampliar a inclusão de pessoas com diferentes necessidades. A Tecnologia Assistiva (TA) desempenha um papel fundamental nesse cenário, possibilitando maior autonomia e qualidade de vida para indivíduos que enfrentam desafios na interação com dispositivos digitais e físicos. Além disso, o investimento em TA gera benefícios que se estendem para além do indivíduo, impactando positivamente as famílias, comunidades e a sociedade como um todo, ao fomentar o crescimento econômico e a participação social [1].

No cenário de tecnologias assistivas, o Eye Tracking (ET) surge como um mecanismo inovador para lidar com vários desafios existentes, entre eles o controle de dispositivos sem o uso das mãos. O ET utiliza o rastreamento dos olhos a partir de dispositivos - especializados ou não - e vem sendo amplamente utilizado em pesquisas científicas até aplicações comerciais, destacando-se como uma ferramenta de grande relevância em diversas áreas do conhecimento e da indústria [2]. Contudo, apesar de seu potencial e da diversidade de casos de uso, essa tecnologia ainda pode alcançar uma adoção mais generalizada, não se limitando a nichos específicos de cada campo em que se insere, e sem depender de equipamentos extras especializados para realizar o rastreamento ocular [3].

Considerando que é possível potencializar o acesso e o uso de ferramentas de rastreamento ocular no desenvolvimento de software, essa pesquisa traz um *framework*, chamado **UITracking**, capaz de facilitar a criação de interfaces controladas por ET para dispositivos de uso cotidiano, como smartphones e tablets, eliminando a necessidade de aparelhos adicionais. Essa abordagem tem o potencial de ampliar significativamente o alcance e a acessibilidade dessa tecnologia, permitindo que ela seja integrada em uma variedade de contextos e aplicações. A proposta deste estudo é oferecer aos desenvolvedores um *framework* abrangente, composto por um conjunto de ferramentas e APIs, que permita a integração de funcionalidades de rastreamento ocular e navegação em aplicativos de forma eficiente e simplificada [4]. O **UITracking** está disponível no GitHub.

A adoção deste *framework* pode abrir novas possibilidades no desenvolvimento de aplicativos voltados a públicos diversos, como pessoas com deficiências motoras. Além disso, oferece uma experiência inovadora no mercado de tecnologia móvel.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção visa fornecer uma descrição dos temas abordados no presente estudo. Inicialmente, os tópicos foram organizados de maneira a construir um conhecimento progressivo, partindo dos fundamentos teóricos até a implementação do *framework*. Foram abordados os seguintes conceitos: tecnologia assistiva, interação humano-computador, princípios e funcionamento do ET, o uso do ET em plataformas móveis e, por fim, o desenvolvimento do *framework* *UITracking* e sua validação.

2.1 TECNOLOGIA ASSISTIVA

As tecnologias assistivas desempenham um papel fundamental na promoção da acessibilidade e inclusão, proporcionando maior autonomia e qualidade de vida para pessoas [5]. Essas soluções abrangem dispositivos e softwares que auxiliam na comunicação, mobilidade e interação com ambientes digitais e físicos. Além de reduzir barreiras funcionais, elas fomentam a independência e a dignidade dos usuários, permitindo maior participação social e profissional [1]. O desenvolvimento dessas tecnologias tem avançado significativamente, impulsionado por pesquisas em usabilidade e interação humano-computador [6]. Assim, a adoção de tecnologias assistivas contribui para uma sociedade mais equitativa e acessível.

2.2 INTERAÇÃO HUMANO-COMPUTADOR

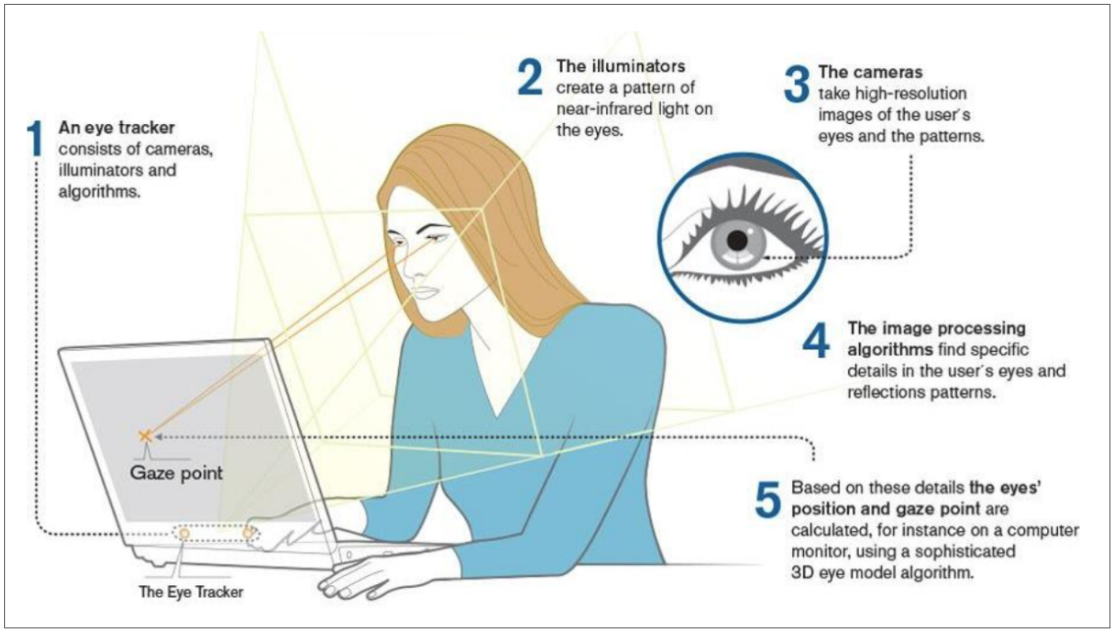
A Interação Humano-Computador (IHC) é um campo interdisciplinar que estuda como os humanos interagem com dispositivos e sistemas computacionais, buscando melhorar essa relação por meio de soluções tecnológicas. A partir desses estudos, é possível criar novos dispositivos que melhoram o desempenho e a experiência humana [6]. No contexto de plataformas móveis, expandir e aprimorar as técnicas de interação humano-computador torna-se cada vez mais necessário para atender um público mais amplo e diverso, como pessoas com deficiência e idosos [4].

2.3 EYE TRACKING

O rastreamento ocular, ou ET, é uma tecnologia que permite registrar e analisar os movimentos dos olhos. A concepção de controlar computadores por meio do olhar surgiu na década de 1960, porém, foi apenas em 1981 que Bolt apresentou um sistema capaz de realizar essa interação [7]. Em 2025, essa tecnologia está presente em diversos ramos da ciência, da psicologia à medicina, e surge com o objetivo de diminuir barreiras de acessibilidade para pessoas que apresentam dificuldade em utilizar equipamentos como teclado, mouse ou telas [3] [8] [4]. A partir de sensores ópticos e câmeras, os ROs capturam aspectos como direção do olhar, duração da fixação e trajetórias oculares [9]. Com pode ser visto na Figura 1, os ROs utilizam majoritariamente os métodos de reflexão da córnea e centro da pupila os quais um emissor de luz infravermelha direciona um feixe para os olhos, iluminando a pupila e gerando reflexos na superfície ocular. Essas reflexões, combinadas com a detecção do centro da pupila, permitem que as câmeras capturem os movimentos dos olhos [2] [7]. Entretanto, é válido ressaltar que a principal limitação do método de reflexão da córnea é a sensibilidade aos movimentos da cabeça, o que pode aumentar o número de erros na detecção [8].

A análise dos movimentos oculares permite identificar áreas de interesse e acionar comandos com base em ações do usuário [9]. Nesse contexto, o olhar se torna uma ferramenta para interação humano-computador, capaz de funcionar como uma fonte de entrada para dispositivos. Com isso, é possível criar alternativas de interação com interfaces gráficas modernas que dispensam o uso de equipamentos convencionais, como o mouse [10] [11].

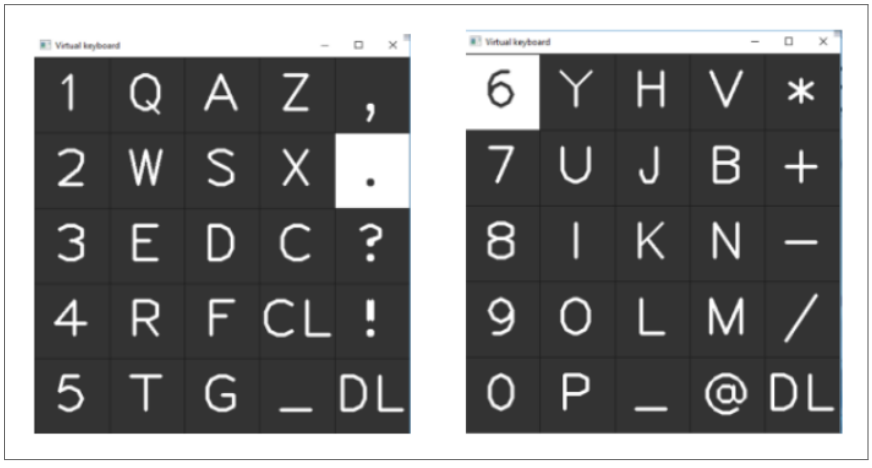
Figura 1 – Funcionamento de rastreadores oculares



Fonte: *Eye Tracking and Its Applications* - 2021 [9].

Um estudo, em 2019, explorou o uso do RO para controlar um teclado por meio de uma *interface*. O protótipo desenvolvido apresenta teclas em um formato retangular, iluminando-as sequencialmente à medida que o usuário move o globo ocular - ver Figura 2. A seleção da tecla ocorria por meio do fechamento dos olhos, acionando automaticamente a letra iluminada [11]. Esse sistema demonstrou como a interação humano-computador pode ser utilizada para realizar tarefas mais complexas, como escrever palavras, além de oferecer reflexões sobre novas possibilidades de uso para os ROs no desenvolvimento do *framework* UITracking.

Figura 2 – Interface de teclado controlada por RO



Fonte: Eye Gaze Controlled Virtual Keyboard - 2019 [11].

2.4 FUNDAMENTOS DO EYE TRACKING

Com o avanço dos ROs, tornou-se viável capturar e analisar os padrões visuais dos usuários durante a interação com sistemas digitais. Essa capacidade permite compreender como as pessoas processam informações visuais, identificando tanto dificuldades quanto oportunidades de aprimoramento. Para interpretar os dados obtidos por meio do RO, é fundamental compreender os principais componentes dessa análise. Com isso é possível obter *insights* sobre determinado artefato que utiliza o *eye tracking* para melhorar alguma interação humano-computador, como experiência do usário em interfaces. Estudos anteriores propuseram uma categorização do comportamento visual dos usuários, que auxilia na interpretação desses dados [2] [7].

- **Fixação** (*Fixation*): Olhar espacialmente estável que dura de 100 a 300 ms. Durante uma fixação, a atenção visual do participante está focada em uma área específica do estímulo e desencadeia processos cognitivos. A duração da fixação varia conforme a tarefa e as características do participante.

- **Sacada** (*Saccade*): movimentos oculares contínuos, rápidos e comuns, com duração de 40 a 50 ms, que ocorrem entre fixações, mas proporcionam percepção visual limitada.

- **Dilatação e contração da pupila** (*Pupil dilation and constriction*): a pupila é a abertura por onde a luz entra no olho, cuja dilatação é controlada pelo músculo da íris. Uma pupila maior pode indicar um aumento no esforço cognitivo.

- **Caminho de varredura** (*Scan path*): por meio das sacadas, os olhos fixam diferentes partes de um estímulo, formando uma série de fixações, ou Áreas de Interesse (ADI) visitadas, ordenadas cronologicamente.

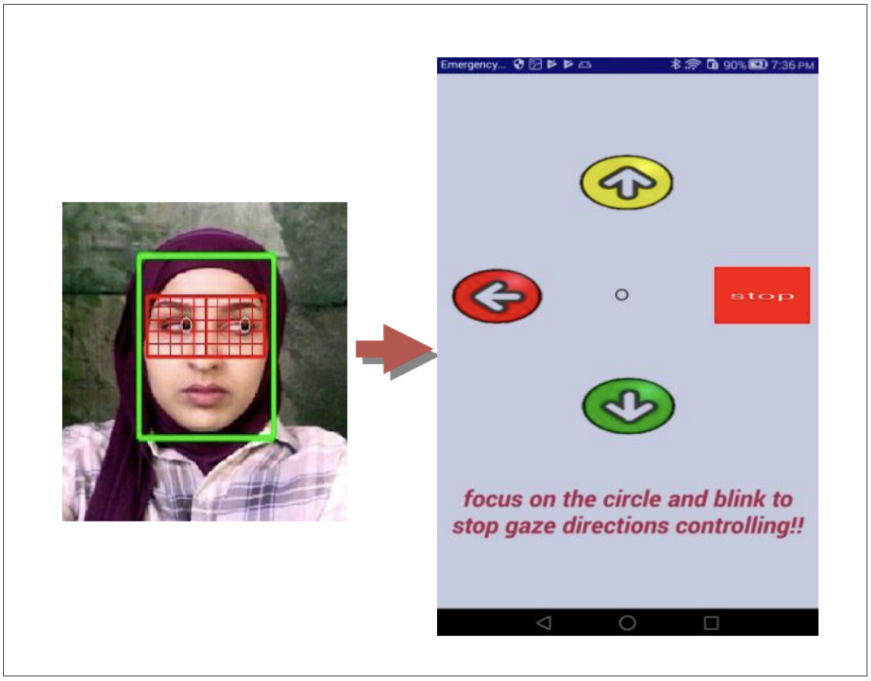
2.5 EYE TRACKING EM DISPOSITIVOS MÓVEIS

A implementação de ET em dispositivos móveis contribui para o avanço da tecnologia no design de *interfaces* adaptativas e inclusivas [12]. Um estudo, realizado em 2018, demonstrou a viabilidade do eye tracking em dispositivos móveis por meio de um aplicativo Android que permitia o controle da interface por meio da movimentação ocular, bem como a seleção de botões por piscadas. O sistema utilizou detecção facial e rastreamento da pupila através da

câmera frontal, sem a necessidade de equipamentos adicionais. Como pode ser visto na Figura 3, o aplicativo se mostrou eficiente para controlar uma *interface* simples, mesmo em dispositivos modestos, que apresentaram uma precisão de 66% em ambientes internos e 76% em externos, indicando que a taxa pode melhorar com câmeras de maior resolução. Ainda assim, os algoritmos de rastreamento ocular que dispensam o uso de hardware externo ainda enfrentam desafios significativos em termos de precisão e desempenho, uma vez que ao contrário de dispositivos com tela maior, como computadores, os smartphones e tablets apresentam limitações de espaço e movimentação, o que restringe sua capacidade de uso em aplicações com *interfaces* complexas. Dessa forma, os resultados obtidos com o uso de ET até então reforçam o potencial do eye tracking para o controle de interfaces em dispositivos móveis, bem como alertam a necessidade de um maior desenvolvimento da tecnologia para interfaces mais complexas, visando deixá-las mais acessíveis e interativas [4] [13].

Após a realização de buscas em plataformas como o GitHub, onde desenvolvedores hospedam *frameworks* voltados para o ecossistema Apple, não foi identificado nenhum projeto que forneça ferramentas específicas para a criação de *interfaces* adaptadas ao controle por *eye tracking*. As soluções existentes se limitam a oferecer funcionalidades para a captura de atributos como coordenadas do olhar (x, y), distância do usuário em relação ao dispositivo, sem fornecer suporte direto para a construção de *interfaces* otimizadas para essa forma de interação. Além disso, em setembro de 2024, com o lançamento do iOS 18, a Apple incorporou nativamente o controle do sistema operacional por meio do eye tracking [14]. No entanto, essa funcionalidade ainda apresenta limitações, especialmente no que se refere à compatibilidade com aplicativos que não foram projetados para incorporar essa tecnologia — o que, atualmente, corresponde à grande maioria das aplicações disponíveis.

Figura 3 – Exemplo de aplicação móvel baseada em Eye Tracking

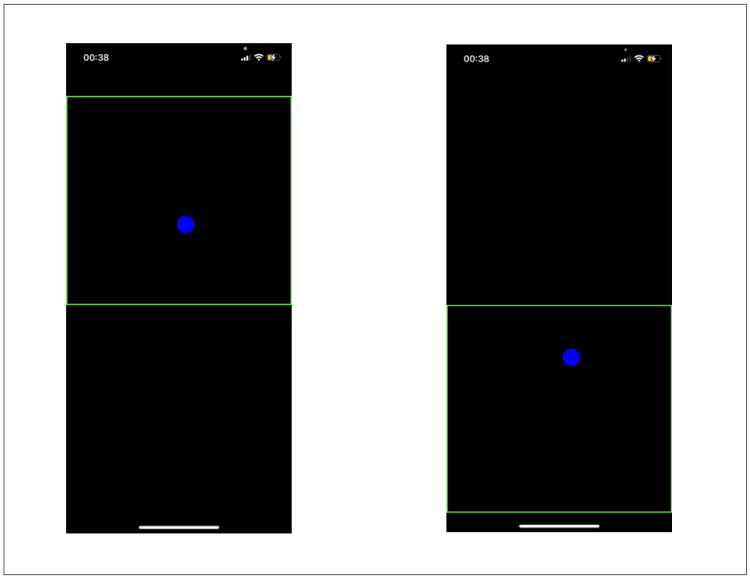


Fonte: Eye Tracking Based Mobile Application - 2018[4].

2.6 NAVEGAÇÃO POR ÁREA DE INTERESSE

Área de Interesse (ADI) é um conceito fundamental em estudos de eye tracking definido a partir da análise dos movimentos oculares [8]. A ADI é responsável por delimitar regiões específicas, que são relevantes para a tarefa ou para o indivíduo [2]. No contexto de interfaces digitais, a navegação por área de interesse pode ser concretizada por meio de mecanismos de foco e seleção como ilustrado na Figura 4. O foco se refere ao elemento visual que está ativo, geralmente indicado por uma borda ou uma mudança visual, como uma cor diferente. Já a seleção é o processo de destacar um ou mais itens em uma lista ou área da interface, permitindo que o usuário execute ações sobre eles [15]. A navegação por foco e seleção está alinhada às diretrizes do Web Content Accessibility Guidelines (WCAG), que estabelecem padrões para garantir que interfaces digitais sejam acessíveis a todos os usuários. A WCAG, em sua versão 2.1, define princípios para assegurar que a ordem de navegação seja previsível e lógica (Critério 2.4.3 – Ordem do Foco) e fornecer indicadores visuais claros para destacar o elemento atualmente em foco (Critério 2.4.7 – Foco Visível) [16]. Essas diretrizes são essenciais para usuários que dependem de tecnologias assistivas, como dispositivos de rastreamento ocular.

Figura 4 – Exemplo de navegação por área de interesse



Fonte: Elaborado pelo autor - 2025.

2.7 DESENVOLVIMENTO FRAMEWORK - UITRACKING

No contexto de desenvolvimento de software, um *framework* pode ser descrito como uma coleção de ferramentas que fornece uma estrutura pré-definida para o desenvolvimento de aplicações. Com essa estrutura, novas funcionalidades podem ser adicionadas dentro de um conjunto controlado de regras e padrões [17]. A plataforma iOS, desenvolvida pela Apple, oferece um ecossistema para o desenvolvimento de aplicações móveis e segue um modelo de arquitetura modular, no qual os desenvolvedores podem escolher quais *frameworks* e componentes podem ser usados conforme a necessidade de sua aplicação [18].

A escolha do ecossistema iOS para o desenvolvimento do *framework UITracking*, voltado para aplicações controladas por Eye Tracking, foi motivada por uma série de fatores técnicos e estratégicos. Em primeiro lugar, o acesso aos recursos Apple, como Xcode e dispositivos físicos para testes, desempenhou um papel fundamental para a escolha. Outro fator determinante foi a presença do sensor TrueDepth nos dispositivos iPhone, a qual, a partir do iPhone X, permite o mapeamento facial detalhado e oferece suporte nativo para detecção de movimentos oculares. Por fim, o domínio da linguagem Swift e do fluxo de desenvolvimento iOS foi um fator crucial, uma vez que evitou a necessidade de uma curva de aprendizado acentuada, permitindo maior eficiência na implementação do *UITracking*. O código está disponível em GitHub.

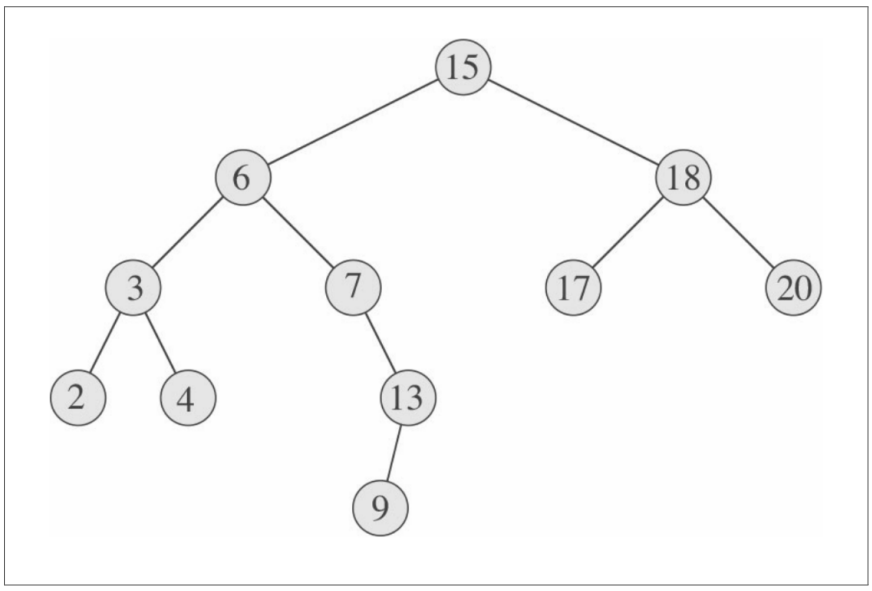
3 FERRAMENTAS

Esta seção apresenta as principais ferramentas utilizadas no desenvolvimento do *UITracking*. A seguir são apresentados os conceitos de AV, padrão estrutural Composite e algoritmo DFS que, juntos, possibilitam a manipulação de estruturas complexas de interface de forma recursiva e flexível. Além disso, o ARKit, o RxSwift e o UIKit desempenharam papéis fundamentais no desenvolvimento do *framework*, e suas descrições também serão apresentadas a seguir.

3.1 ÁRVORE DE VISUALIZAÇÃO

O conceito é utilizado no desenvolvimento de interfaces gráficas para organizar e gerenciar componentes visuais de maneira hierárquica. Essa abordagem permite compor interfaces complexas a partir de componentes mais simples, promovendo modularidade e reuso [19]. A AV utiliza árvores como estrutura de dados, na qual cada nó representa uma *view* individual ou um grupo de *views*. As árvores são estruturas hierárquicas amplamente utilizadas em algoritmos de ciência da computação, como em busca binária - Figura 5 - e são compostas por nós conectados que partem de um nó raiz e se ramificam em subníveis [20].

Figura 5 – Representação de uma árvore binária

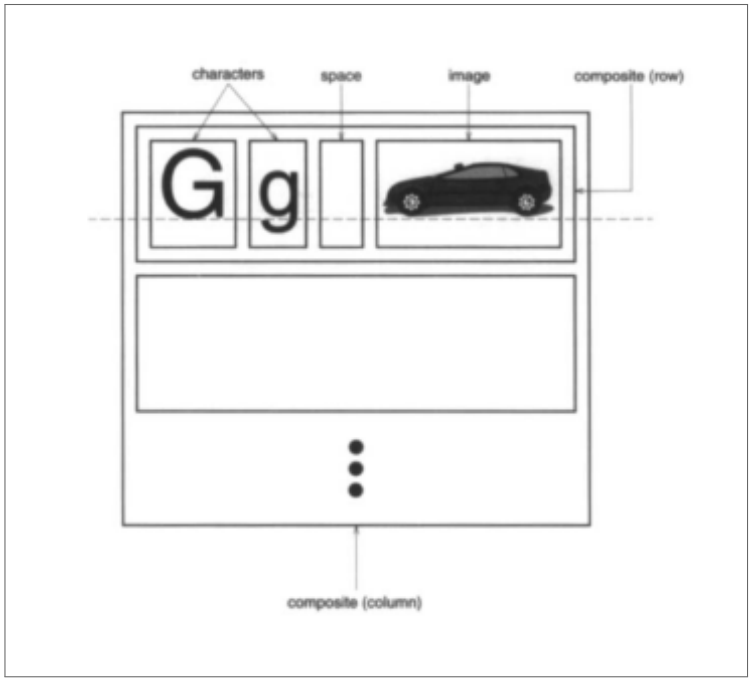


Fonte: Introduction to Algorithms - 2009 [20].

Sistemas gráficos, como editores de desenho, frequentemente utilizam hierarquias de com-

ponentes visuais onde elementos, como botões ou textos, podem ser combinados para formar componentes maiores, como mostrado na Figura 6. A Uber, empresa de tecnologia que oferece serviços de transporte, utiliza AV em sua aplicação para gerenciar a complexidade das interfaces, as quais podem ser dinâmicas e reativas - interfaces que reagem a mudanças de estado. Isso facilita o controle do ciclo de vida de cada *view*, permitindo atualizações em componentes específicos de forma eficiente pela interface [19] [21].

Figura 6 – Composição de interface por componentes menores

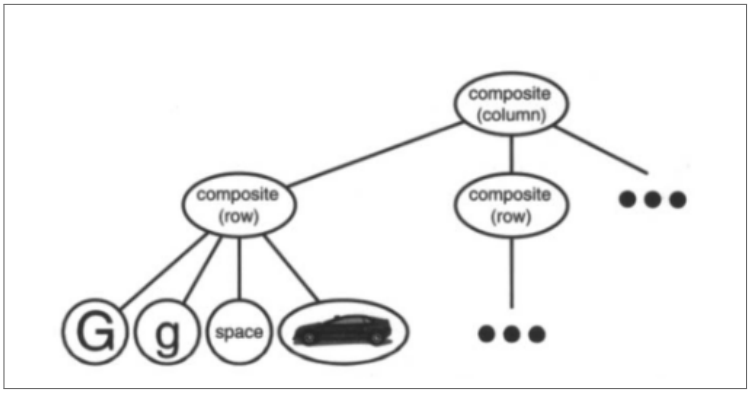


Fonte: Design Patterns: Elements of Reusable Object-Oriented Software - 1994 [21].

3.2 COMPOSITE

O padrão Composite é um design pattern estrutural que unifica o tratamento de objetos individuais e composições, organizando-os em uma estrutura hierárquica onde nós e as folhas compartilham a mesma interface - ver Figura 7. Dessa forma, o padrão promove a reutilização de código e facilita operações recursivas sobre os elementos, favorecendo a construção de interfaces flexíveis e reutilizáveis [21]. Essa abordagem é útil em sistemas que utilizam tecnologias onde as interações do usuário podem exigir a manipulação dinâmica de componentes visuais, como em RO. Neste estudo, a utilização do padrão composite foi fundamental para representar o conceito da AV citado anteriormente.

Figura 7 – Árvore gerada pelo Composite



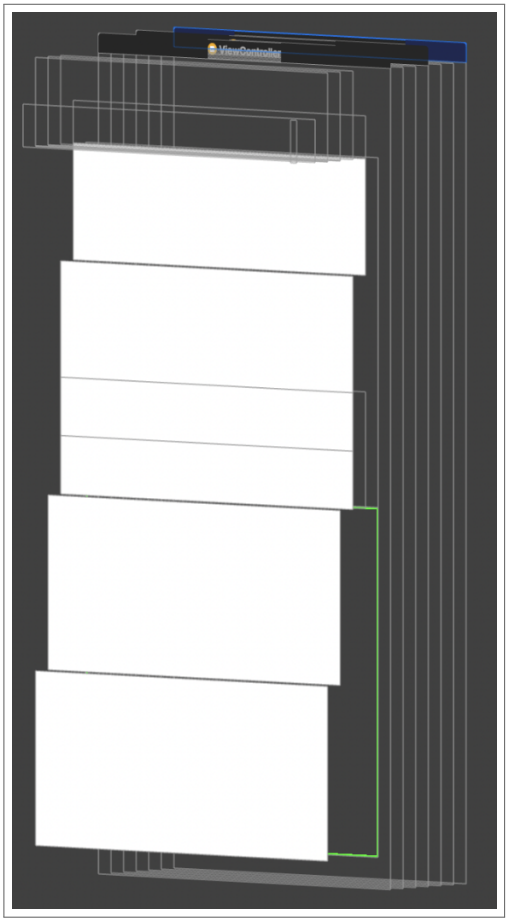
Fonte: Design Patterns: Elements of Reusable Object-Oriented Software - 1994 [21].

3.3 DEPTH FIRST SEARCH

A busca em profundidade é um algoritmo clássico para a travessia de grafos e árvores, explorando os nós de forma recursiva. Ele avança por um caminho até atingir o limite de profundidade antes de retornar e explorar outras ramificações [20]. No contexto de AV, o DFS percorre a estrutura de maneira sistemática, permitindo a identificação de elementos, extração de propriedades e execução de operações em níveis específicos da árvore. Ao utilizar DFS, cada elemento presente na estrutura é considerado para interação ou renderização, facilitando modificações dinâmicas sobre uma view [20] [21]. Abaixo segue os usos de DFS em árvores de visualização, bem como a Figura 8 que demonstra o uso em uma *interface*:

- **Interação Usuário-Interface:** Identificar quais *views* estão sob o foco do usuário e obter informações relevantes para o fluxo de iteração.
- **Atualização de Estados:** Propagar mudanças, como animações ou ajustes de layout, em componentes específicos da hierarquia.

Figura 8 – Uso do DFS na interface para identificar a *view* (nó) em foco e desenhar uma borda verde no nível hierárquico

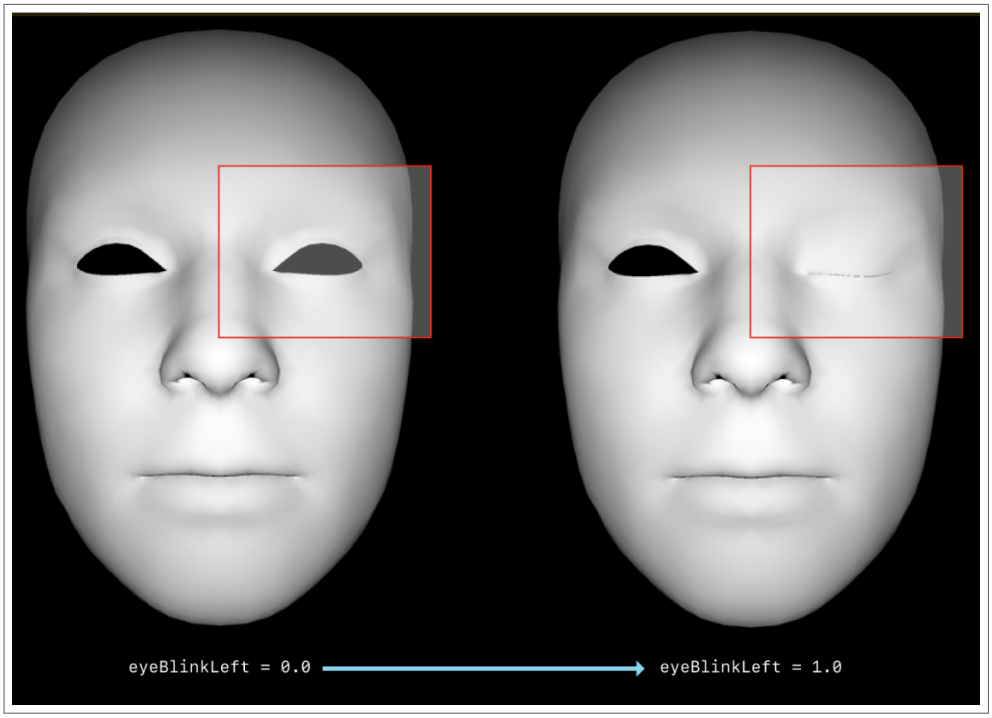


Fonte: Elaborado pelo autor a partir do ambiente de desenvolvimento *Xcode* - 2025.

3.4 ARKIT

O ARKit é um *framework* para desenvolvimento de experiências em realidade aumentada a qual a partir de sensores integrados é possível mapear o espaço tridimensional ao redor do usuário e permitir a projeção de objetos virtuais. Com a introdução do sensor TrueDepth a partir do *iPhone X*, os dispositivos iOS passaram a suportar nativamente o rastreamento facial, permitindo a detecção de expressões e seus movimentos [22]. Assim, por meio de sessões de rastreamento, é fornecido um conjunto de coeficientes de expressão facial, chamado *blendShapes*, além de outras propriedades, como *lookAtPoint* — um vetor tridimensional que estima o ponto no qual o olho está focado. O blendShapes é um dicionário, coleção de pares chave-valor, que descreve expressões faciais detectadas com base em movimentos, variando seus coeficientes de 0.0 (posição neutra) a 1.0 (máximo movimento) [23]. A Figura 13 demonstra a alteração do coeficiente de acordo com a expressão detectada.

Figura 9 – Mudança de coeficiente no olho esquerdo após o movimento



Fonte: Documentação Apple 2025 [23].

3.5 RXSWIFT

É uma biblioteca para a programação reativa em Swift, amplamente utilizada no desenvolvimento de aplicações iOS. Baseada no conceito de Reactive Extensions (Rx), ela permite que desenvolvedores lidem com eventos assíncronos e fluxo de dados de maneira declarativa. A programação reativa é bastante utilizada em cenários onde há interação constante entre o usuário e a interface, como é o caso de aplicações controladas por eye tracking. Ao utilizar o RxSwift, é possível observar eventos interpretados e reagir a essas mudanças de forma eficiente e organizada. Essa abordagem também melhora a clareza do código e facilita a manutenção do software, pois promove a separação de responsabilidades e evita o acoplamento excessivo entre os componentes [24].

3.6 UIKIT

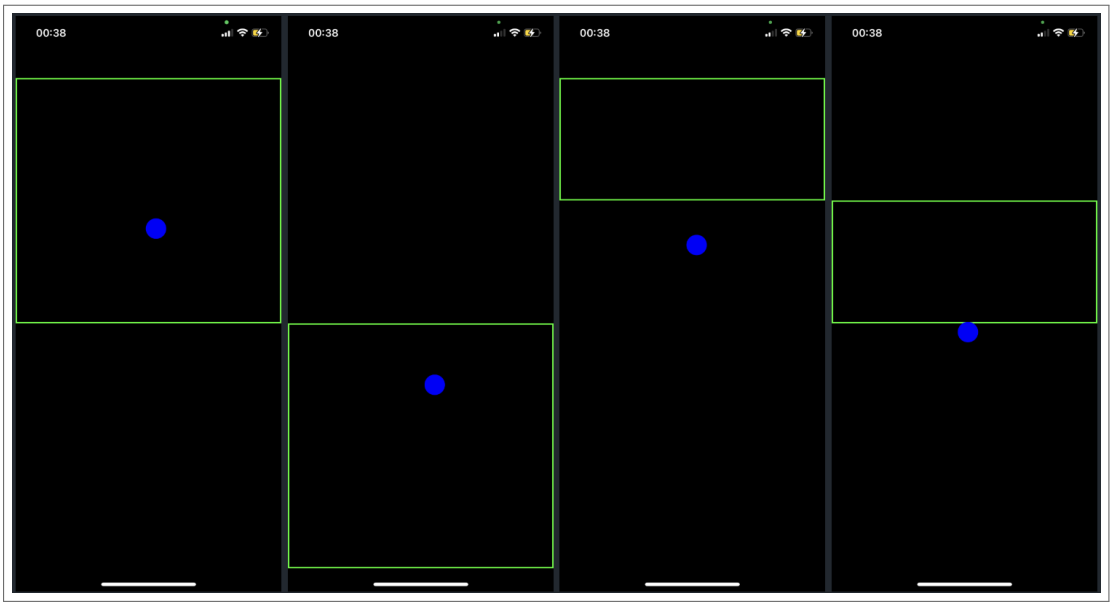
É um *framework* para a construção de interfaces gráficas em aplicativos iOS. Ele fornece um conjunto de componentes essenciais, como botões, tabelas, coleções, navegação e controles de gestos, permitindo que os desenvolvedores criem experiências interativas e responsivas.

Embora seja uma tecnologia consolidada e amplamente utilizada, sua abordagem programática e baseada em código pode representar desafios para iniciantes [25].

4 FRAMEWORK - UITRACKING

Dada a relevância de fornecer melhorias à interação humano-computador com dispositivos móveis, este trabalho apresenta uma ferramenta nativa para iOS, projetada para facilitar o desenvolvimento de aplicações controladas por eye tracking [26]. O objetivo principal é proporcionar aos desenvolvedores uma ferramenta para criar *interfaces* móveis, sem a necessidade de configurações complexas. Na Figura 10, é possível verificar um simples exemplo do que o *UITracking* será capaz de fazer através do controle do olho - navegar por área de interesse, selecionar uma área específica e associar um evento, como navegação para uma outra tela, a uma *view*. Ao fim, será disponibilizado a documentação - para melhor entendimento - e o código no GitHub.

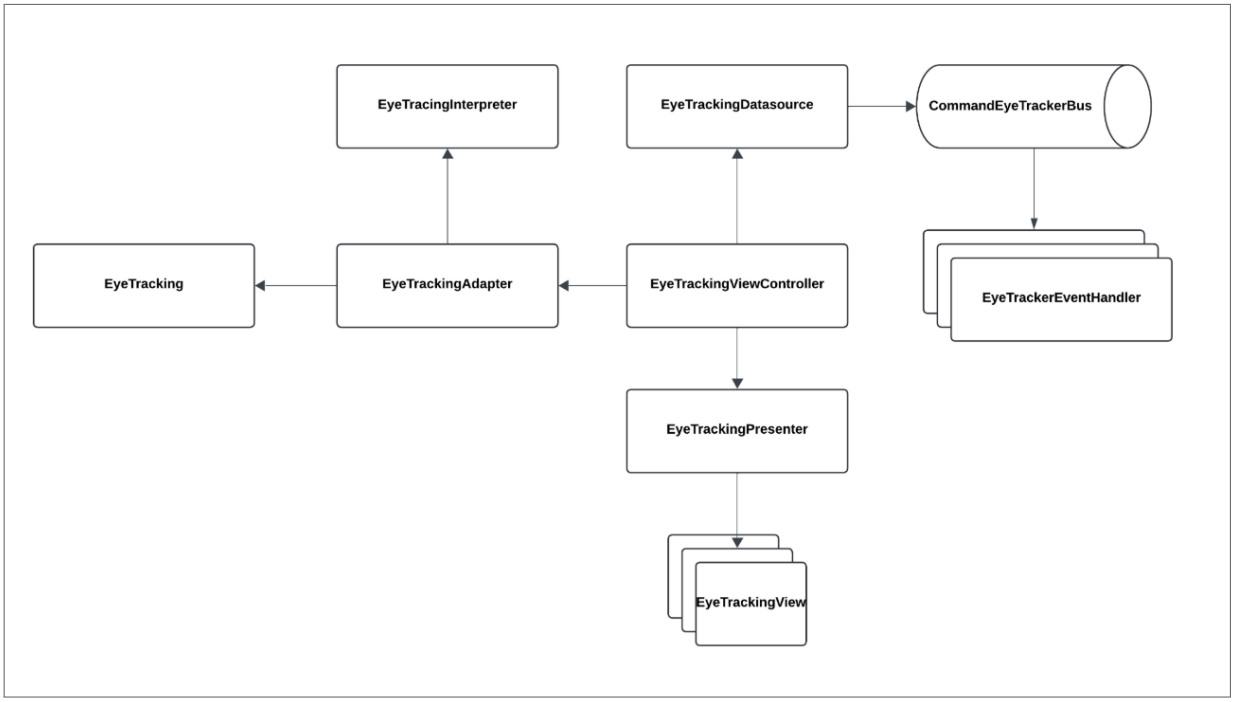
Figura 10 – Movimento entre seções



Fonte: Elaborado pelo autor 2025.

A partir de padrões já conhecidos por desenvolvedores iOS, como o *delegate*, *observer* e *composite*, foi construída uma arquitetura, como pode ser visto na Figura 11, que visa separar a aplicação em camadas bem definidas. A seguir, descrevemos brevemente a responsabilidade de cada componente da arquitetura:

Figura 11 – Diagrama representando a arquitetura e os principais componentes



Fonte: Elaborado pelo autor - 2025.

4.1 EYE TRACKING

O componente é responsável por processar dados crus relacionados ao rastreamento ocular em tempo real. Nesse momento, é utilizado o ARKit para capturar coordenadas x e y do ponto de fixação do globo ocular [22]. Uma outra responsabilidade do *EyeTracking* é gerenciamento do *lifecycle* das sessões de rastreamento, desde a inicialização até a finalização. Através do padrão *delegate*, o qual permite a comunicação entre dois objetos, é possível enviar mensagens em momentos críticos, como informações de x e y para outros componentes do sistema. [27].

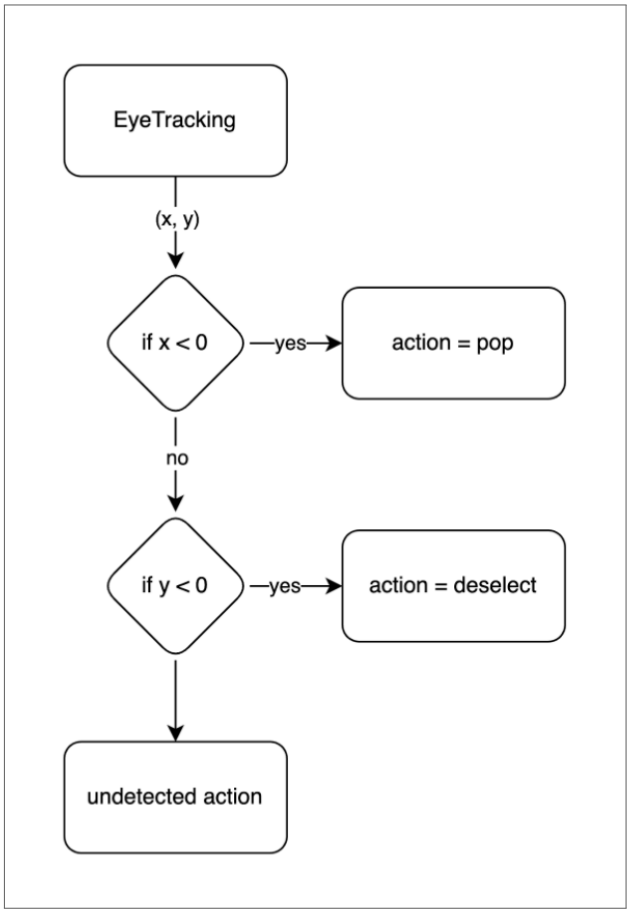
O componente *EyeTracking* utilizado neste estudo foi baseado em um pacote que simplifica o uso do ARKit, disponível no GitHub. Contudo, foram adicionadas novas funcionalidades tais como: detecção de piscadas oculares e o envio de mensagens para outros componentes do sistema.

4.2 EYE TRACING INTERPRETER

O componente é responsável por interpretar dados de rastreamento ocular gerados pelo componente *EyeTracking*. É nesse momento que é dado um valor semântico para os valores de x e y. Na figura 12, apresento o algoritmo responsável pela identificação de ações pelo

movimento ocular.

Figura 12 – Algoritmo de interpretação de (x,y) coletado pelo componente *EyeTracking*



Fonte: Elaborado pelo autor - 2025.

De um modo geral, segue as seguintes etapas:

- Dado um valor de (x,y) detectado pelo rastreamento ocular, caso o valor x identificado seja menor que 0, a ação de retornar à tela anterior é detectada.
- Logo em seguida, dado um valor de (x,y) detectado pelo rastreamento ocular, caso o valor de y identificado seja menor que 0, a ação de retornar o foco para o componente acima na hierarquia de *views* é detectada.
- Caso não se enquadre nesses dois últimos pontos, não foi detectado nenhuma ação a partir do movimento ocular.

4.3 EYE TRACKING ADAPTER

Um *adapter* é um padrão de design estrutural que permite que interfaces incompatíveis se comuniquem. Ele age como um "tradutor", adaptando a interface de uma classe para que ela possa ser usada por outra classe sem modificar o código original [21]. O *EyeTrackingAdapter* utiliza o *EyeTracingInterpreter* para adaptar as coordenadas (x,y) em ações e tem a responsabilidade de repassar a informação adaptada - as ações - para o outros componentes através do *delegate* [27]. Com isso, é possível segregar responsabilidades de cada componente.

4.4 EYE TRACKING VIEW CONTROLLER

É responsável por integrar a funcionalidade de rastreamento ocular, através do *EyeTrackingAdapter*, com os componentes responsáveis pela *interface*. Sua principal função é controlar a lógica de navegação, registrar informações geradas a partir de ações do usuário e garantir que as ações detectadas sejam refletidas na tela.

4.5 EYE TRACKING PRESENTER

Sua responsabilidade é gerenciar a interação entre a *view* e os processos da aplicação, garantindo que a interface e a lógica de dados permaneçam separadas. Dessa forma, promove um design modular, testável e adaptável, uma vez que permite a interação com estruturas de *views* diferentes, contanto que se adeque a forma que o *presenter* se comunica [28]. No presente estudo, o *presenter* foi utilizado para delegar as seguintes requisições:

- **Gerenciar a área de foco:** Responsável por solicitar a remoção e a criação da área de foco em um nível hierárquico da *view*.
- **Quantidade de *views* em um nível hierárquico:** Delega a consulta para determinar o número de nós filhos disponíveis em um nível hierárquico.
- **Consultar a última *view* do nível hierárquico:** Delega a verificação, com base na hierarquia atual e na interação do usuário, se a *view* atual é uma folha.
- **Carregar Eventos:** Identifica o evento associado a uma *view*.

4.6 EYE TRACKING DATASOURCE

O componente é responsável por armazenar e propagar as ações detectadas do usuário entre diferentes contextos, como em outros *view controllers*. Ele permite que as informações como interpretação das ações e profundidade do nível hierárquico da AV sejam mantidas, mesmo em outros controladores, para retornar ao estado anterior após remoção de um *view controller*.

4.7 COMMAND EYE TRACKER BUS

O Command Bus é um padrão de design usado para desacoplar a emissão de comandos da execução de sua lógica. Ele funciona como um intermediário entre o solicitante de uma ação (quem emite o comando) e o manipulador (quem executa o comando) [29]. O uso do *CommandEyeTrackerBus* tem o intuito de registrar, remover e executar eventos associados um nível hierárquico da AV. Com isso, é possível desacoplar a lógica dos eventos, através de manipuladores os quais apresentam o código a ser executado. Para tornar isso possível, foi utilizada a interface *EyeTrackerEventHandler* que padroniza a implementação de manipuladores, garantindo flexibilidade e extensibilidade ao permitir mudanças em tempo de execução.

4.8 EYE TRACKING VIEW

O componente é responsável por criar a estrutura hierárquica de *views* que representam zonas interativas. Com o auxílio do padrão estrutural *Composite*, foi possível criar estruturas complexas a partir de componentes *EyeTrackingView* mais simples [21]. Além disso, o componente também é responsável por implementar funcionalidades referentes a modificação visual da *interface*, como a adição e remoção de bordas, como também faz a busca recursiva de atributos e estados, como o nome do evento, sendo determinante no processamento das ações.

Abaixo segue a demonstração de como uma DFS foi utilizada para buscar atributos, como os eventos associados.

Código Fonte 1 – Busca em profundidade dos eventos associados a uma *EyeTrackingView*

```
1     /// Retrieves the event associated with a specific view at a given depth and
          section in the hierarchy.
```

```
    ///
    /// This method supports navigating through a hierarchical structure of `EyeTrackingView` instances,
    /// retrieving the event associated with a particular view at a specified section and depth.
    ///
    /// - Parameters:
    ///   - memoDepth: An array representing the path to the target view, used for recursive traversal.
    ///   - section: The section index of the current view in the `subTrackingViews` array.
    ///
    /// - Returns: A `String?` representing the event associated with the specified view, or `nil` if no event is found.
    public func getEvent(on memoDepth: [Int], and section: Int) -> String? {
        if memoDepth.isEmpty {
            // If memoDepth is empty, return the event for the current section.
            return subTrackingViews[section].event
        }

        var tempMemoDepth = memoDepth
        let indexComponent = tempMemoDepth.removeFirst()

        // Recursively traverse the subTrackingViews hierarchy to retrieve the event.
        return subTrackingViews[indexComponent].getEvent(on: tempMemoDepth,
                                                         and: section)
    }
```

Fonte: Elaborado pelo autor 2025

4.9 API - CRIAÇÃO DA ÁRVORE DE VISUALIZAÇÃO

Com o objetivo de reduzir a curva de aprendizado dos desenvolvedores e tornar a adoção mais intuitiva, foi criado uma Application Programming Interface (API). Nesse aspecto, foi desenvolvido um componente para facilitar a construção de AVs, contribuindo para uma melhor experiência de desenvolvimento e redução de erros. A API, através da estrutura *EyeTracking-TreeViewBuilder*, fornece ferramentas para a configuração da AV, oferecendo métodos de alto nível para a construção de componentes hierárquicos. Abaixo, descreve-se cada um desses métodos:

- ***addSection*:** responsável por adicionar um nível hierárquico na tela. A quantidade de

seções define em quantas vezes a tela será dividida.

- ***addView*:** responsável por adicionar uma *view* a uma seção na tela.

- ***addViewWithEvent*:** responsável por adicionar uma *view* associada a um evento.

Como podemos ver no código 3, a sintaxe torna a API expressiva, permitindo que os desenvolvedores foquem na estrutura lógica da interface sem precisar manipular referências diretamente.

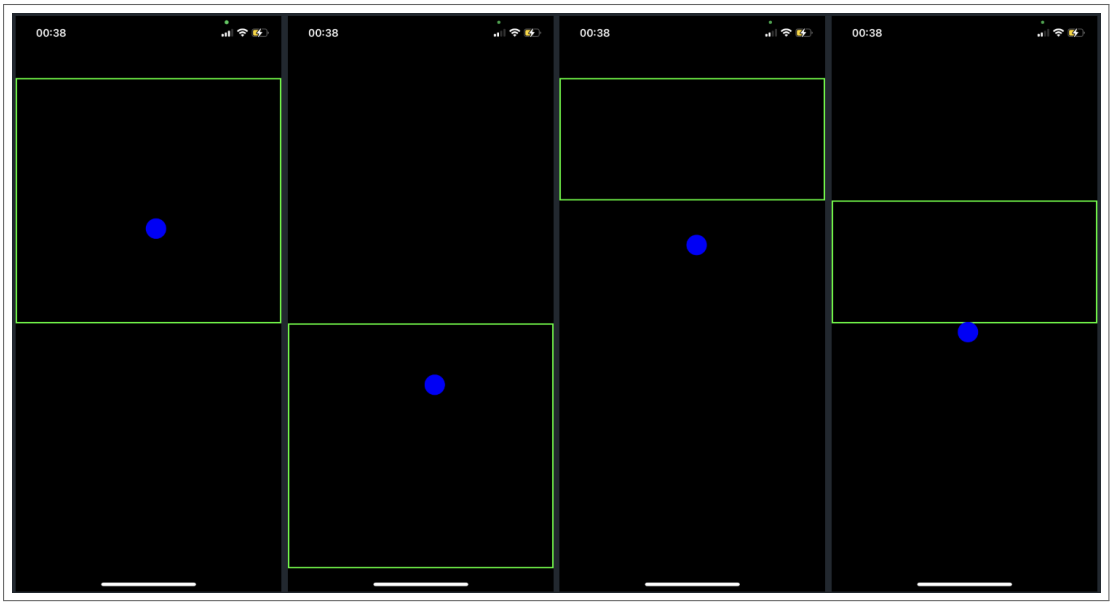
Código Fonte 2 – Construção de AV utlizando o *EyeTrackingTreeViewBuilder*

```
let view = EyeTrackingTreeViewBuilder()
            .addSection { node in
                node.addView(ExampleView())
                    .addViewWithEvent(ExampleView(), event: ButtonEventHandler.
                        event)
            }
            .addSection{ node in
                node.addView(ExampleView())
            }.build()
```

Fonte: Elaborado pelo autor 2025

Esse trecho de código presente em 3, pode ser traduzido nessa estrutura de *views*.

Figura 13 – Tela dividida em duas seções, a qual na primeira seção apresenta duas subdivisões. Uma das subdivisões apresenta um evento associado.



Fonte: Elaborado pelo autor 2025.

4.10 FERRAMENTA PARA ESTUDOS

A captura de dados brutos, como as coordenadas x e y detectadas pelo componente EyeTracking, apresentado na seção 4.1, pode fornecer informações relevantes para pesquisas que investigam padrões de olhar e interação visual. Esses dados podem ser úteis em estudos sobre comportamento humano, usabilidade e acessibilidade, permitindo uma compreensão mais aprofundada da forma como os usuários interagem com os dispositivos. Nesse contexto, o framework desenvolvido neste estudo oferece suporte à obtenção dessas informações por meio de métodos específicos de configuração, permitindo que os usuários acessem e analisem os dados capturados de forma eficiente.

Código Fonte 3 – Configuração para ativar logs no *framework*

```
datasource.tracking.setLogScreenPoint(status: .activate)
```

Fonte: Elaborado pelo autor 2025

Como pode ser visto na Figura 14, a váriavel *datasource* é encontrada dentro do escopo de um *view controller* que herda do *EyeTrackingViewController* presente na seção 4.4. Na Figura 15 há uma demonstração dos dados brutos obtidos através da detecção do rastreamento ocular, os quais apresentam um formato (x,y *timestamp*).

Figura 14 – Ativação de *logs*

```
class ViewController: EyeTrackingViewController {

    init(){
        let view = EyeTrackingTreeViewBuilder()
            .addSection { node in
                node.addView(EyeTrackingView())
            }
            .addSection { node in
                node.addView(EyeTrackingView())

            }.build()

        super.init(view: view)

        datasource.tracking.setLogScreenPoint(status: .activate)
    }
```

Figura 15 – *Logs* no *console*

```
Line: 20  Col: 65
Screen Point:  (178.98277580738068, 315.74847388267517) at timestamp:  2025-03-27
17:49:33 +0000
Screen Point:  (178.75478088855743, 315.56666684150696) at timestamp:  2025-03-27
17:49:33 +0000
Screen Point:  (177.82017141580582, 317.5361592769623) at timestamp:  2025-03-27
17:49:33 +0000
Screen Point:  (177.11012810468674, 319.74400305747986) at timestamp:  2025-03-27
17:49:33 +0000
Screen Point:  (178.64667624235153, 320.04008090496063) at timestamp:  2025-03-27
17:49:33 +0000
Screen Point:  (180.02487391233444, 319.3115699291229) at timestamp:  2025-03-27
17:49:33 +0000
Screen Point:  (181.28278613090515, 320.81472277641296) at timestamp:  2025-03-27
17:49:33 +0000

Message from debugger: killed
```

Fonte: Elaborado pelo autor - 2025.

4.11 DOCUMENTAÇÃO

A documentação desempenha um papel essencial no desenvolvimento de software, fornecendo informações estruturadas que facilitam a compreensão e adoção de uma tecnologia pelos desenvolvedores [30]. No contexto deste trabalho, foi elaborada uma documentação para explicar a forma que se pode interagir com o *UITracking*. A documentação foi disponibilizada

no GitHub e organizada de maneira a abranger desde a instalação até exemplos práticos de uso [26]. Ela aborda os seguintes tópicos principais:

- **Introdução:** Visão geral do framework e seu propósito.
- **Recursos:** Explicação sobre a integração com UIKit e recomendações de uso.
- **Instalação:** Passos detalhados para instalação.
- **Como Usar:** Guia sobre o conceito de AV e os principais componentes do framework, tais como: *EyeTrackingTreeViewBuilder*, *EyeTrackerEventHandler*, *EyeTrackingViewController* e *CommandBus*.
- **Exemplo de Código:** Demonstração prática de implementação.

Essa organização foi projetada para oferecer uma curva de aprendizado intuitiva, na tentativa que desenvolvedores possam compreender rapidamente o funcionamento da ferramenta.

4.12 INTERAÇÃO DO USUÁRIO

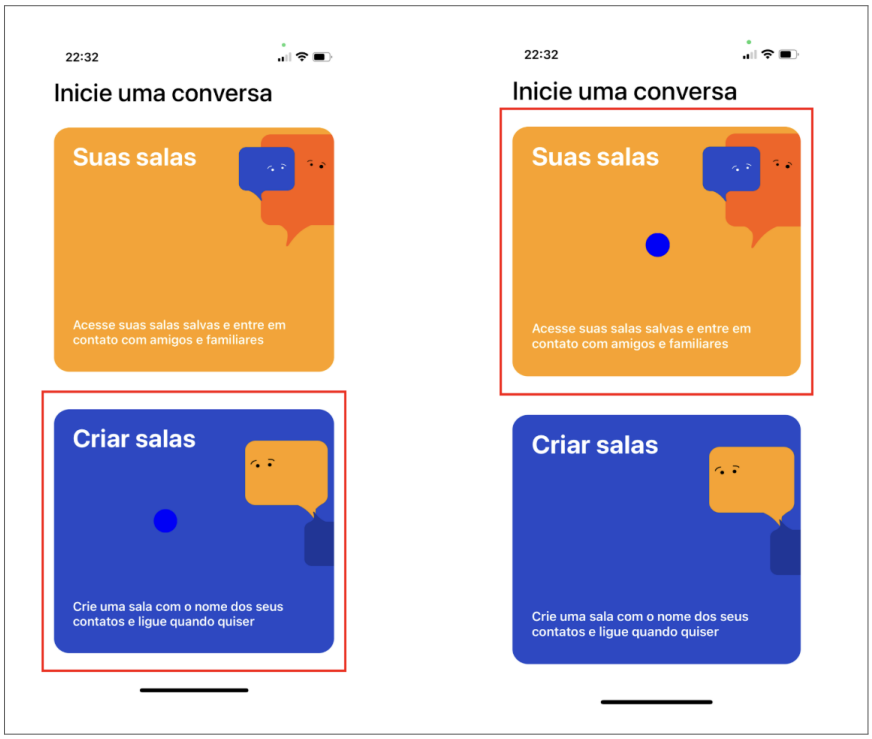
Ao criar aplicações com o *framework* presente neste estudo, é preestabelecido novas formas de interação com o dispositivo móvel. A seguir, detalha-se cada interação:

- **Ao piscar os olhos**, o usuário conseguirá selecionar um nível hierárquico em especifico ou acionar evento que determinada view está associada, como uma navegação para uma outra tela.
- **Ao olhar completamente para cima**, o usuário fará a deseleção de área em foco, voltando um nível hierarquico.
- **Ao olhar completamente para esquerda**, o usuário voltará para a tela anterior, caso exista um empilhamento de telas

4.13 CASO DE USO

Nesta seção, é apresentado um caso de uso que exemplifica como o framework proposto neste estudo pode ser usado para criar aplicações mais complexas, destacando cenários práticos de sua utilização. Na Figura 16, é exemplificado a na navegação por foco e seleção. A estrutura da interface é composta por dois níveis hierárquicos e nos subníveis foram adicionados eventos para simular o clique em um botão.

Figura 16 – Navegação por área de interesse em uma aplicação com AV mais complexa



Fonte: Elaborado pelo autor - 2025.

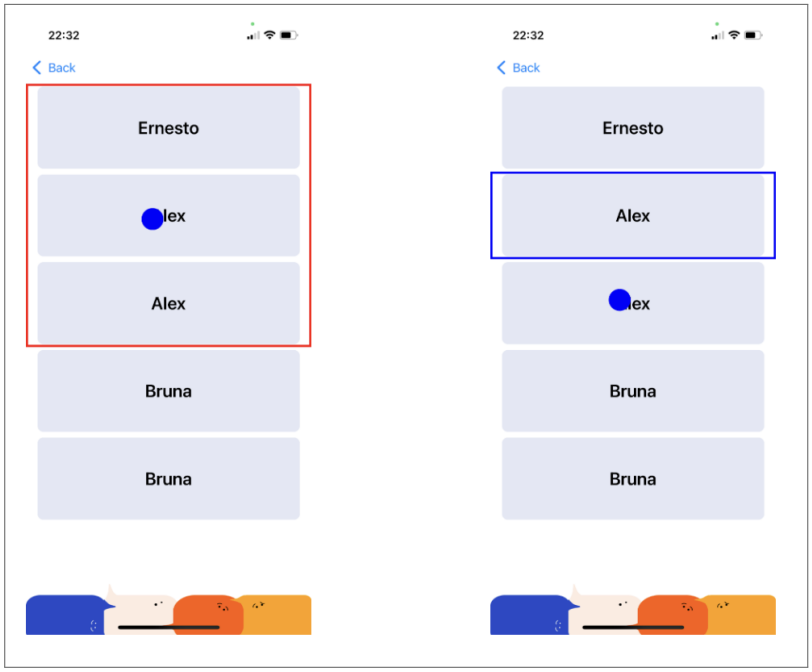
Após a seleção de um nível hierárquico, como mostrado na Figura 17, o sistema direciona o foco para o último elemento da hierarquia, permitindo ao usuário decidir entre realizar uma nova seleção ou, em caso de equívoco, retornar ao nível hierárquico superior.

Figura 17 – Estado após a seleção de um nível hierárquico

Fonte: Elaborado pelo autor - 2025.

Após a seleção do último nível hierárquico com evento associado de navegação entre telas, o sistema permite a transição para uma nova tela, como pode ser visto na Figura 18. O mesmo princípio de navegação por área de interesse é detectado, permitindo ao usuário novas interações. Caso o usuário deseje, é possível retornar à tela anterior.

Figura 18 – Transição de tela com o uso do *framework*



Fonte: Elaborado pelo autor - 2025.

A partir do caso de uso criado nesta seção, foi demonstrado que o framework desenvolvido neste estudo pode ser utilizado para criar aplicações mais complexas. Assim, os experimentos realizados reforçam o potencial do *UITracking* como uma base sólida para o desenvolvimento de interfaces controladas pelo olhar, ampliando seu escopo de aplicação no ecossistema iOS.

5 METODOLOGIA

Esta seção tem como objetivo apresentar o processo de validação do **UITracking**, desenvolvido para aplicações controladas por eye tracking em dispositivos iOS. Para isso, foi conduzida uma pesquisa de abordagem metodológica mista, visando identificar os principais desafios de usabilidade da solução. A investigação combinou a aplicação do PSSUQ para a análise quantitativa com entrevistas, utilizadas na análise qualitativa. A seguir, descreve-se em detalhes a metodologia adotada.

5.1 ENTREVISTAS COM DESENVOLVEDORES

A realização de entrevistas permite a obtenção de insights profundos sobre a perspectiva e a experiência dos participantes sobre um artefato [31]. Nesse contexto, entrevistas são fundamentais em estudos exploratórios, pois possibilitam uma compreensão mais detalhada dos fenômenos analisados, especialmente em contextos onde a interação do usuário com novas tecnologias é um fator crítico [32]. Nesse cenário, o método qualitativo, realizado através de entrevistas semiestruturadas, é essencial para a descoberta de fatores que influenciam a adoção do *framework UITracking* e, também, é importante para entender como os desenvolvedores interagem com a API proposta.

Para obter uma análise mais abrangente, este estudo combina entrevistas semiestruturadas e questionários de múltipla escolha, utilizando tanto abordagens qualitativas quanto quantitativas. A coleta de dados quantitativos garante uniformidade nas perguntas e respostas, permitindo comparações estatísticas entre os participantes [32]. Essa abordagem mista visa garantir um entendimento mais completo dos desafios enfrentados pelos desenvolvedores e fornecer métricas objetivas que auxiliem na validação do *framework* proposto. Dessa forma, compreender as dificuldades e percepções dos desenvolvedores ao utilizar o *UITracking* é um passo crucial para garantir sua adoção.

A coleta de dados ocorreu exclusivamente de forma online devido a praticidade e possibilidade de ser executada com pessoas de outros estados e países.

1. **Entrevistas Semiestruturadas**

    a) **Público:** Desenvolvedores iOS com diferentes níveis de proficiência em UIKit.

b) **Estruturação:** Introdução, contextualização do *framework* e da experiência do entrevistado, execução de tarefas com níveis diferentes de dificuldade, desafios enfrentados e sugestão de melhorias.

2. **Formulário de Múltipla Escolha**

a) **Público:** Desenvolvedores iOS que participaram da entrevista semiestruturada.

b) **Estruturação:** Avaliação pós-uso do *framework* com foco em três dimensões principais: utilidade do sistema, qualidade da informação e qualidade da interface.

5.1.0.1 Entrevistas Semiestruturadas

A retenção de conhecimento em programação depende das experiências e percepções individuais de cada desenvolvedor. Com o intuito de investigar como essas vivências moldam o aprendizado e superação dos desafios na adoção de uma novo *framework*, foi escolhida a Entrevista Semiestruturada. A partir disso, foi selecionado um grupo de desenvolvedores iOS, com diferentes níveis de proficiência em UIKit, para analisar fatores que influenciam a adoção, sobretudo dificuldades encontradas ao utilizar o *UITracking*. Nesse contexto, em pesquisas qualitativas, a formulação de perguntas deve priorizar as modalidades de investigação "como", "o que", "de que forma"e "por que", permitindo uma exploração mais profunda dos fenômenos estudados sem necessariamente pressupor relações de causa e efeito [31]. Isso é importante porque promove a flexibilidade necessária para adaptar a conversa às respostas do entrevistado. Dessa forma, o *template* das entrevistas pode ser encontrado em A e foi estruturado com base nos seguintes eixos:

- **Experiência do Desenvolvedor:** Para mapear o histórico do entrevistado em desenvolvimento iOS, estabelecendo um contexto para compreender sua trajetória e nível de conhecimento.

- **Contextualização do *Framework*:** Explicação prévia do propósito e conceitos, através da documentação disponibilizada no Github, para garantir que todos os participantes tenham um entendimento básico antes de utilizá-lo.

- **Execução de Tarefas:** Aplicação de desafios práticos para avaliar o entendimento e usabilidade do UITracking. O detalhamento das tarefas está em 5.1.0.1.1

- **Desafios e Dificuldades:** Identificação de barreiras técnicas e conceituais que dificultam a adoção do *UITracking*.
- **Sugestões de Melhorias:** Coleta de feedback sobre possíveis ajustes na API, documentação e usabilidade. Servirá, também, como indicador para construção de novas funcionalidades.

5.1.0.1.1 Desafios Propostos para Avaliação do UITracking

Durante a etapa de entrevistas, os participantes foram convidados a executar uma série de tarefas práticas com o objetivo de avaliar o entendimento e usabilidade do *UITracking*. As tarefas foram sequenciais e progressivamente mais complexas, com foco em testar a compreensão da estrutura de *views*, manipulação de eventos e navegação entre telas. A seguir, são descritas as tarefas aplicadas:

1) **Construção de Interface com Múltiplas Seções:** O participante deverá implementar uma interface dividida em duas seções visuais distintas, cada uma composta por uma única view, utilizando as estruturas disponibilizadas pelo *UITracking*. O objetivo dessa tarefa é avaliar a compreensão dos desenvolvedores sobre a organização hierárquica de *views* e o uso das abstrações fornecidas, como *EyeTrackingTreeViewBuilder*, *EyeTrackingViewController* e *EyeTrackingView*.

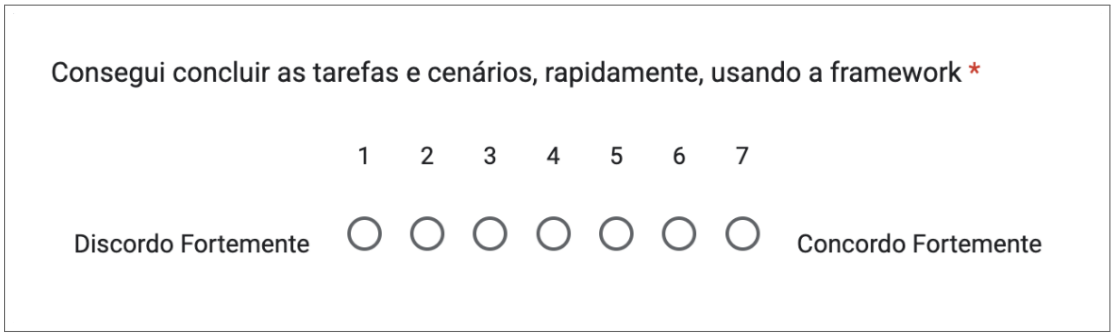
2) **Registro de Eventos:** Com base na interface implementada na tarefa anterior, esta etapa tem como objetivo tornar a view da primeira seção interativa. A tarefa consistirá em adicionar um evento a uma *view* e que ele seja devidamente disparado através do fechamento do globo ocular, resultando na exibição de uma mensagem no *console* do ambiente de desenvolvimento. Essa tarefa buscou avaliar a interação dos participantes em lidar com o tratamento de eventos do UITracking, principalmente no uso do componente *EyeTrackerEventHandler* e conceito de *Command Bus*.

3) **Navegação entre Telas:** A terceira e última tarefa consistiu na modificação do evento definido no segundo desafio, substituindo-o pela implementação de uma navegação para uma nova tela. Esta nova tela deveria conter três seções, sendo cada uma composta por duas views. O objetivo desta etapa foi avaliar tanto a capacidade de manipular a navegação entre telas quanto a confirmação do conhecimento adquirido na construção de *views* hierárquicas, utilizando o *UITracking*.

5.1.0.2 Formulário de Múltipla Escolha

O uso de questionários padronizados é uma prática amplamente adotada na avaliação da usabilidade de sistemas, pois permite a obtenção de dados quantitativos que complementam análises qualitativas [33]. Entre os instrumentos mais utilizados, destaca-se o PSSUQ, um questionário desenvolvido para medir a satisfação dos usuários após a interação com um sistema computacional. Nesse método, os questionários utilizam a escala Likert que varia de 1 (fortemente concordo) a 7 (fortemente discordo) permitindo uma análise detalhada da percepção dos usuários sobre o sistema testado - ver Figura 19. O PSSUQ é composto por 16 perguntas que avaliam três dimensões principais: utilidade do sistema, qualidade da informação e qualidade da interface. Dessa forma, a aplicação do PSSUQ no contexto deste estudo teve como objetivo quantificar a usabilidade do *framework* desenvolvido, fornecendo métricas que complementam as percepções qualitativas obtidas nas entrevistas, bem como garante a obtenção de dados estruturados, reduzindo vieses subjetivos [34].

Figura 19 – Pergunta presente no PSSUQ realizada no Formulário de Múltipla Escolha - escala Likert invertida



Fonte: Elaborado pelo autor - 2025.

5.1.1 Amostragem

A seleção dos participantes foi feita por conveniência, uma estratégia comum em pesquisas qualitativas devido à sua rapidez, baixo custo e facilidade de acesso aos participantes [35]. Nesse cenário, a seleção dos entrevistados tentou abranger desenvolvedores iOS com diferentes níveis de experiência, bem como proficiência em UIKit - iniciante, intermediário e avançado. A escolha desse grupo se justifica pelo fato de que a API proposta se integra diretamente ao ecossistema iOS.

A amostra da pesquisa foi composta por 10 desenvolvedores e trouxe um recorte diverso

para compreender como diferentes perfis interagem com o *UITracking*. O recrutamento foi realizado por meio de redes sociais, e-mails e contato com profissionais conhecidos. O número de participantes foi definido com base no princípio da saturação teórica, ou seja, quando novas entrevistas deixaram de acrescentar informações significativas ao estudo [36]. Por fim, a participação foi totalmente voluntária, garantindo que os desenvolvedores contribuíssem espontaneamente com o estudo, sem qualquer tipo de obrigação ou incentivo financeiro.

5.1.2 Análise de dados

5.1.2.1 Entrevista Semiestruturada

Para analisar os dados coletados durante a entrevista, foi utilizada a análise temática indutiva. O raciocínio indutivo pode ser empregado para descobrir padrões, categorias e temas emergentes a partir dos dados brutos. Com isso, seguiu-se as seguintes etapas após a coleta dos dados [37]:

1. In Vivo Coding
2. Analytic Memos
3. Categorização

5.1.2.1.1 In Vivo Coding

Durante a transcrição das entrevistas, foi utilizada a técnica de coding - um método de descoberta para identificar os significados das seções individuais dos dados - para organizar e categorizar as informações. Apesar de existir diversos tipos de coding, foi utilizado o In Vivo Coding, o qual coleta palavras e frases exatas dos participantes, o que é ideal para capturar diretamente a percepção e as emoções dos entrevistados através de expressões utilizadas, como, por exemplo “senti dificuldade”, “não entendi bem”. A partir disso, é possível padronizar, classificar e, posteriormente, organizar cada dado em categorias emergentes para análise posterior [37].

5.1.2.1.2 Analytic Memos

Após algumas entrevistas, foi utilizada a técnica de analytic memo, que é uma ferramenta para refletir sobre os dados durante a análise. À medida que foram realizadas algumas codificações, foi possível escrever memos analíticos sobre os padrões emergentes, bem como as dificuldades observadas. Esses memos, posteriormente, ajudaram a organizar o pensamento e a contextualizar as observações, facilitando a transição para a construção de teorias ou conclusões mais amplas a partir dos dados [37].

5.1.2.1.3 Categorização

Após a aplicação do In Vivo Coding, a categorização dos dados permitiu a identificação de padrões em suas falas, com foco especial nas dificuldades relatadas pelos participantes. Esse processo facilitou a interpretação dos significados subjacentes às respostas e permitiu uma análise estruturada dos principais desafios enfrentados [37]. A abordagem utilizada foi baseada no estudo de Myllärniemi et al. [38], que investigou fatores que podem influenciar a adoção e o uso de *frameworks*. A seguir, são detalhadas as categorias utilizadas como referência neste trabalho:

- **Capacidades da API:** Recursos e funcionalidades que a API oferece, sendo frequentemente utilizada para justificar a escolha de um determinado *framework*.
- **Informações insuficientes:** Falta de informações que facilitam o entendimento do *framework*.
- **Informações não direcionadas e não organizadas:** Desenvolvedores perdem tempo procurando informações específicas, o que reduz a eficiência.
- **Exemplos de códigos e vídeos:** Exemplos de código e vídeos fornecem demonstrações visuais e práticas de como o *framework* funciona, facilitando a compreensão e o aprendizado.
- **APIs especializadas:** A falta de APIs especializadas impede que desenvolvedores com diferentes habilidades e objetivos usem o *framework* de forma eficiente.

5.1.2.2 Formulário de Múltipla Escolha

No fim das entrevistas, foi solicitado o preenchimento do formulário de múltipla escolha, o qual apresenta o objetivo de complementar a pesquisa qualitativa, bem como estabelecer comparações estatísticas entre os participantes. Para isso, as seguintes avaliações foram consideradas:

- Análise descritiva das médias do PSSUQ
- Triangulação de Dados - Análise das Categorias

5.1.2.2.1 Análise descritiva das médias do PSSUQ

Após o fim da coleta dos dados através do formulário, calcula-se a média de respostas para cada uma das três dimensões do PSSUQ: utilidade do sistema, qualidade da informação e qualidade da Interface. Isso é importante uma vez que ajudará a identificar áreas onde os participantes estão mais ou menos satisfeitos com o *framework*. Abaixo segue valores de referência para a versão 3 do PSSUQ para cada dimensão, bem como o índice geral [34]. É relevante enfatizar que quanto mais próxima de 1 for a média, melhor será o resultado.

- **Utilidade do Sistema:** 2.80
- **Qualidade da Informação:** 3.02
- **Qualidade da Interface:** 2.49
- **Geral:** 2.82

A partir dos diferentes perfis de desenvolvedores presentes na entrevista, como desenvolvedores com diferentes níveis de experiência, pode-se comparar as médias entre esses grupos para identificar se há diferenças significativas na percepção do sistema. Isso pode ajudar o presente estudo a entender se a usabilidade varia com a experiência ou familiaridade com o desenvolvimento iOS.

5.1.2.2.2 Triangulação de Dados - Análise das Categorias

Neste momento, será feita uma triangulação entre as informações obtidas da análise temática indutiva das entrevistas e os resultados quantitativos do PSSUQ. Essa etapa busca

identificar convergências e divergências entre os dados qualitativos e quantitativos, permitindo uma compreensão mais completa da experiência do *framework*, ressaltando os desafios e dificuldades. Dessa forma, é possível validar e complementar as percepções dos desenvolvedores com uma análise estatística objetiva.

6 RESULTADOS

Nesta seção são apresentados e discutidos os principais resultados obtidos da análise realizada pelo estudo.

6.1 ENTREVISTAS SEMIESTRUTURADAS

6.1.1 Participantes

Primeiramente, foi realizada uma categorização - iniciante, intermediário, avançado - dos entrevistados, elencando fatores como: anos de experiência em desenvolvimento e anos em iOS.

Tabela 1 – Perfil dos entrevistados

Entrevistado	Anos de Experiência	Anos em iOS	Categoria
P1	2	0.5	Iniciante
P2	1	1	Iniciante
P3	3	3	Intermediário
P4	3	2	Intermediário
P5	3	3	Intermediário
P6	3	2	Intermediário
P7	4	2	Intermediário
P8	4	2	Intermediário
P9	10	3.5	Avançado
P10	8	8	Avançado

Como pode ser visto na Tabela 1, foram feitas 10 entrevistas com pessoas de níveis variados de experiência em desenvolvimento de software, bem como experiência em iOS. Para categorizar os entrevistados, foram utilizados os critérios abaixo:

- **Iniciante:** Menos de 2 anos de experiência em iOS.
- **Intermediário:** De 2 anos até 3 anos de experiência em iOS.
- **Avançado:** Mais de 3 anos de experiência em iOS ou vasta experiência geral em desenvolvimento combinada com experiência em iOS.

Portanto, notou-se a predominância de pessoas que se enquadram como intermediárias sob o critério estabelecido, como pode ser visto na Tabela 5. Isso permite inferir que a maioria dos entrevistados já possui uma base no ecossistema iOS, bem como contato com mais conceitos de desenvolvimento, o que reduz a necessidade de aprendizado desses fundamentos durante a avaliação. Dessa forma, os participantes puderam focar diretamente na compreensão do funcionamento do *framework*, favorecendo uma análise mais precisa da sua usabilidade e da clareza da API. Além disso, a presença de participantes iniciantes e avançados permite uma análise mais abrangente das dificuldades enfrentadas em diferentes níveis de experiência. Enquanto os iniciantes podem encontrar barreiras relacionadas à familiaridade com o ecossistema iOS e aos conceitos fundamentais de desenvolvimento, os participantes avançados tendem a avaliar o *UITracking* sob a perspectiva da eficiência, flexibilidade e integração com práticas mais complexas.

Tabela 2 – Porcentagem por Categorias

Categoria	**Quantidade**	**Porcentagem**
Iniciante	2	20%
Intermediário	6	60%
Avançado	2	20%

6.1.2 Categorização das Respostas

A análise qualitativa dos dados coletados nas entrevistas foi conduzida utilizando a técnica de In Vivo Coding, garantindo que as categorias emergiram diretamente das falas dos participantes [37]. Esse método permitiu identificar os termos e expressões mais recorrentes usados pelos desenvolvedores ao descrever suas interações com o *UITracking*, sobretudo as dificuldades, refletindo com precisão suas percepções sobre a usabilidade. Segue na Tabela 3 exemplos da extração das transcrições:

Tabela 3 – Trechos Codificados das Entrevistas

Entrevistado	**Transcrição**	**Código Extraído**
P4	"Senti falta de um direcionamento das interações possíveis na documentação, tipo piscar os olhos, olhar pra cima."	"falta de um direcionamento das interações"
P1	"Não sei se quem vai utilizar isso está habituado com o conceito do command bus."	"habituado com o conceito"

Após a análise utilizando a técnica de In Vivo Coding, foi possível confirmar a presença de fatores previamente estabelecidos pelo estudo Myllärniemi et al [38], além de identificar, também, categorias que elucidam dificuldades na utilização do UITracking pelos entrevistados. O In Vivo Coding e a categorização realizada pode ser encontrada em C.

Tabela 4 – Categorias e Referências

Categoria	**Referência de Origem**
Capacidades da API	Myllärniemi et al
Informações insuficientes	Myllärniemi et al
Exemplos de códigos e vídeos	Myllärniemi et al
Informações não direcionadas e não organizadas	Myllärniemi et al
Conceitualização	Este estudo
Experiência em iOS	Este estudo
Interações com dispositivo	Este estudo
Aprendizado Interativo	Este estudo

Como pode ser visto na Tabela 4, essas foram as categorias identificadas a partir das entrevistas. Inicialmente, havia 5 categorias preestabelecidas. No entanto, após a análise das entrevistas realizadas, apenas 4 dessas categorias foram detectadas de forma consistente nos dados. Além disso, outras 4 categorias adicionais foram identificadas, as quais o detalhamento será apresentado a seguir:

- **Conceitualização:** Conceitos teóricos gerais, como eventos, *handlers*, etc. A exemplo temos "Tive dificuldade inicial no entendimento de *event handler*, mas não sinto que é algo extremamente complexo para entender" (P4) extraído da entrevista.
- **Experiência em iOS:** A falta de experiência no ecossistema iOS pode gerar dificuldades, especialmente no uso do UIKit. Isso foi detectado em falas como: "Diria que o gap

principal seria a questão do UIKit em si" (P7), "Senti maior dificuldade com UIKIt mesmo" (P4).

- **Interações com dispositivo:** Interação com o dispositivo móvel, devido a não familiaridade com o *eye tracking*. Falas como: "Essa parte de voltar olhando pra esquerda é muito massa, no caso seria massa ter um user guide" (P10) comprovam isso.

- **Aprendizado Interativo:** Iniciantes estão acostumados a documentações mais interativas. Isso pode ser pecebido em "Você é iniciante, nunca utilizou tal coisa. Ter um comece por aqui" (P1) e "Se tivesse algo mais interativo, o passo a passo, seria mais fácil para eu entender" (P2).

6.1.2.1 Análise das Dificuldades

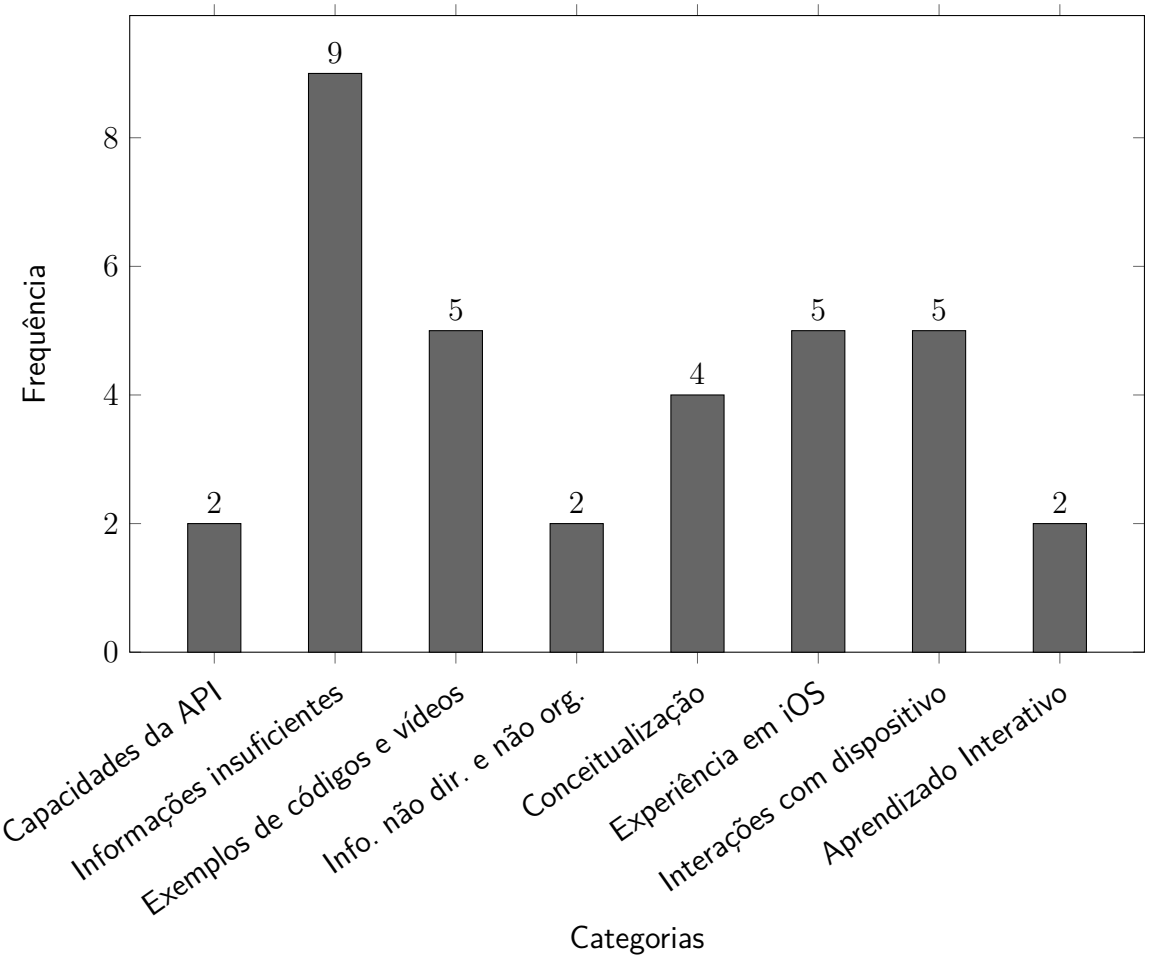
A Figura 20 mostra a frequência da cada categoria nas entrevistas realizada por este estudo. Destaca-se "Informações Insuficientes" como a categoria mais referenciada, representando 90% (9 participantes) dos entrevistados. Isso se deve a falta de informação na documentação referente as funções internas dos componentes disponibilizados, como *EyeTrackingView*. Tal constatação se correlaciona com o estudo de Myllärniemi et al [38] que apontam a documentação como um grande facilitador para a adoção de novos *frameworks*.

Seguido a isso, temos as categorias de "Exemplos de Códigos e Vídeos", "Experiência em iOS" e "Interações com Dispositivo", cada uma com 50% (5 participantes). Apesar da presença de diversos exemplos na documentação, os entrevistados ainda sentiram a necessidade de exemplos mais específicos, como a construção completa de uma tela, aumentando a frequência de "Exemplos de Códigos e Vídeos". Já a categoria "Experiência em iOS" teve uma alta frequência, refletindo a dificuldade de utilizar o UIKit, que exige um nível técnico mais avançado devido sua estrutura de desenvolvimento mais complexa. Por fim, a categoria "Interações com Dispositivo" também apresentou alta frequência, pois a documentação criada se concentrou exclusivamente na parte técnica, sem fornecer demonstrações práticas de como usar o *eye tracking* no dispositivo móvel.

As categorias "Capacidades da API", "Info. não dir. e não org." e "Aprendizado Interativo" tiveram uma menor citação entre os entrevistados, em 20% (2 participantes) cada. Destaca-se que todos os entrevistados classificados como iniciantes mencionaram a categoria de "Aprendizado Interativo", mostrando que essa abordagem pode ajudar a adoção da *framework* por

iniciantes. Além disso, a categoria "Capacidades da API" teve uma maior concentração de desenvolvedores avançados, sugerindo um pensamento crítico mais apurado para avaliar os recursos oferecidos pelo *UITracking* e sua integração com projetos que já desenvolveram anteriormente.

Figura 20 – Gráfico de Frequência de Pessoas



A identificação das categorias relacionadas às dificuldades de adoção do *UITracking* representa um passo importante para compreender os principais obstáculos enfrentados por desenvolvedores durante sua utilização. No entanto, mais do que apenas mapear os desafios, é fundamental propor caminhos para mitigar esses problemas e melhorar a experiência de adoção. Nesse cenário, será apresentada sugestões de melhorias com base nas categorias identificadas, tanto aquelas que correspondem a problemas já reconhecidos em estudos anteriores quanto às novas categorias emergentes observadas nas entrevistas realizadas. Abaixo segue o detalhamento:

- **Capacidades da API:** Adicionar estratégias que favoreçam a personalização e a extensibilidade da API, tais como: integração com projetos existentes, novas expressões

faciais para disparar eventos e navegação lateral por ADI. Para permitir a integração com aplicações existentes, recomenda-se a definição de protocolos ainda mais genéricos que abstraiam os pontos de entrada da API, possibilitando a substituição ou extensão de comportamentos sem a necessidade de alterar o código original. Além disso, a introdução de variáveis de configuração via construtor ou métodos *setters* no componente *EyeTracking* permitiria ao desenvolvedor ajustar qual expressão facial acionaria os eventos, um exemplo disso seria a substituição de piscadas pela abertura da mandíbula. Por fim, a navegação lateral por meio da ADI pode ser implementada por meio da adição de novos atributos em *EyeTrackingView* e *EyeTracingInterpreter*, responsáveis por fornecer informações sobre o estado horizontal do olhar e interpretar movimentos laterais dos olhos, respectivamente.

- **Informações Insuficientes:** Ampliar a documentação do UITracking, oferecendo descrições mais detalhadas dos componentes disponíveis e de suas funcionalidades, incluindo explicações sobre métodos internos, construtores, parâmetros esperados e exemplos de uso. Um exemplo dessa limitação são os componentes *EyeTrackingView* e *EyeTracking-ViewController*, que apresentaram ausência de orientações claras sobre como integrá-los corretamente na aplicação e descrições superficiais de seus métodos internos, respectivamente. Isso, consequentemente, gerou dúvidas durante a adoção inicial do framework.

- **Exemplos de códigos e vídeos:** Adicionar exemplos de código mais específicos, como a tela de cadastro de usuário de um aplicativo. Isso pode facilitar o entendimento por parte dos desenvolvedores, ao apresentar situações concretas de uso. Além disso, incluir vídeos pode contribuir significativamente para o entendimento visual do funcionamento da framework, ajudando a ilustrar comportamentos e interações que nem sempre são facilmente compreendidos apenas com código. Também se destaca a importância de adicionar exemplos de código progressivos, permitindo que o desenvolvedor acompanhe, de forma didática, a montagem de uma interface completa a partir de partes menores e mais compreensíveis.

- **Informações não direcionadas e não organizadas:** Recomenda-se a realocação do conceito de Árvore de Visualização para uma seção específica da documentação, em vez de mantê-lo dentro da seção *How to Use*, a fim de melhorar a clareza e a organização do conteúdo.

- **Conceitualização:** Incluir uma seção dedicada à explicação de conceitos fundamentais para pessoas que ainda não tiveram contato prévio com determinados temas abordados pelo UITracking. Essa seção deve apresentar definições claras e objetivas, auxiliando no entendimento inicial antes da exploração de aspectos mais técnicos do UITracking.

- **Experiência em iOS:** Oferecer tutoriais introdutórios com exemplos práticos que mostrem a relação entre os componentes do UITracking com elementos base do UIKit. Além disso, pode-se considerar o uso de camadas de abstração para reduzir a necessidade de interação direta com UIKit em tarefas comuns.

- **Interações com dispositivo:** É essencial demonstrar de forma clara como o framework pode ser utilizado diretamente no dispositivo móvel, especificando as interações disponíveis ao usuário final. Essa demonstração pode ser realizada de diferentes maneiras, como por meio de um vídeo no YouTube, um GIF ilustrativo na documentação ou outros formatos visuais que tornem as interações mais compreensíveis.

- **Aprendizado Interativo:** Um caminho para mitigar a ausência de recursos interativos é utilizar o Playground da Apple com exemplos práticos. O Playground é um ambiente interativo de codificação que permite visualizar em tempo real os resultados do código Swift, sendo amplamente utilizado para ensino de forma lúdica e prática. Ele permite que iniciantes explorem o uso do UITracking de forma guiada e interativa [39].

6.1.2.2 Análise das Tarefas Executadas

Durante as entrevistas, foi possível evidenciar aspectos positivos que destacam o potencial e a usabilidade do *framework*. 70% dos entrevistados conseguiram executar as tarefas propostas de forma independente, esbarrando apenas em problemas do ecossistema iOS. Os 30% restantes, compostos por participantes iniciantes ou com experiência limitada em UIKit, demonstraram uma maior dependência, buscando esclarecimentos em momentos específicos durante a execução das tarefas. Embora haja necessidade de melhorias, 80% dos entrevistados destacaram aspectos positivos na documentação, especialmente no que se refere ao uso de imagens, que facilitaram a compreensão dos conceitos como AV. Além disso, 60% dos participantes afirmaram que a curva de aprendizado diminuiu com exemplos de código.

A API de construção de telas se mostrou intuitiva, permitindo uma adaptação relativamente

rápida, mesmo para aqueles com pouca experiência no UIKit. A utilização do *EyeTrackingTreeViewBuilder* recebeu elogios por sua clareza e facilidade de uso, tornando viável a criação de interfaces mais complexas sem grande dificuldade. Além disso, 60% dos entrevistados relataram que não acharam complexo adicionar eventos após a fase inicial de familiarização.

Outro ponto relevante detectado foi a flexibilidade do *framework*, evidenciada pelo fato que desenvolvedores mais avançados conseguiram construir interfaces mais complexas - concebido a partir das experiências espontâneas de desenvolvedores que começaram a desbravar o *framework*. Dessa forma, embora existam desafios a serem aprimorados, os aspectos positivos ressaltados indicam que o *framework* oferece uma base promissora para o desenvolvimento de aplicações controladas por *eye tracking*.

Outro ponto relevante detectado foi a flexibilidade do *framework*, evidenciada pelo fato de que desenvolvedores mais avançados conseguiram construir interfaces mais complexas — algo percebido a partir das experiências espontâneas desses desenvolvedores ao explorarem o *framework*. Dessa forma, embora existam desafios a serem aprimorados, os aspectos positivos ressaltados indicam que o *framework* oferece uma base promissora para o desenvolvimento de aplicações controladas por *eye tracking*.

6.2 FORMULÁRIO DE MÚLTIPLA ESCOLHA

Esta seção visa trazer os resultados obtidos da pesquisa quantitativa que tem o objetivo de complementar os dados qualitativos [33].

6.2.1 Participantes

Utilizando os mesmos critérios presentes na seção 6.1.1, têm-se os seguintes perfis de desenvolvedores que preencheram o formulário.

Tabela 5 – Porcentagem por Categorias

Categoria	**Quantidade**	**Porcentagem**
Iniciante	2	20%
Intermediário	6	60%
Avançado	2	20%

6.2.2 Análise das Respostas

Para avaliar a usabilidade do *framework* desenvolvido, utilizou-se o PSSUQ para medir a satisfação dos usuários ao usar um sistema. Os resultados foram comparados ao valor de referência presente em 5.1.2.2.1 deste estudo.

6.2.2.1 Análise descritiva das médias do PSSUQ

A Tabela 6 apresenta os resultados do PSSUQ para cada entrevistado, incluindo a pontuação obtida nas dimensões de utilidade do sistema, qualidade da informação, qualidade da interface e avaliação geral. A análise descritiva das médias para cada uma dessas dimensões foi realizada para avaliar a percepção dos participantes em relação ao *framework*, levando em consideração seu nível de experiência (iniciante, intermediário e avançado).

Tabela 6 – Resultado do PSSUQ para cada entrevistado

Entrevistado	Categoria	Utilidade do Sistema	Qualidade da Informação	Qualidade da Interface	Geral
P1	Iniciante	1.83	4.5	1.33	2.81
P2	Iniciante	3.16	3.5	3.0	3.25
P3	Intermediário	1.83	1.16	1.33	1.43
P4	Intermediário	1.83	1.5	1.0	1.43
P5	Intermediário	2.16	2.5	2.33	2.31
P6	Intermediário	1.5	2.16	2.0	1.81
P7	Intermediário	1.5	1.33	2.0	1.56
P8	Intermediário	2.0	3.33	2.66	2.56
P9	Avançado	1.16	2.66	2.0	1.87
P10	Avançado	1.33	1.66	1.0	1.37

A média das respostas para a dimensão "Utilidade do Sistema" varia entre 1.16 e 3.16, com a maior pontuação observada no entrevistado P2 (3.16). Embora apresente bons índices em comparação com a referência do PSSUQ, é possível fazer algumas inferências. Iniciantes apresentaram uma média mais alta em comparação aos outros grupos, uma vez que a percepção de utilidade pode ser influenciada pelo processo de familiarização com a ferramenta e pela compreensão técnica das funcionalidades.

A dimensão "Qualidade da Informação" apresentou uma maior dispersão nas respostas, com as pontuações variando de 1.16 (P3) a 4.5 (P1). A qualidade da informação depende diretamente de diversos fatores presentes na documentação, como clareza, organização, relevância e profundidade das explicações. Nesse cenário, para iniciantes é notável a necessidade de uma documentação mais detalhada e estruturada para facilitar a compreensão do funcio-

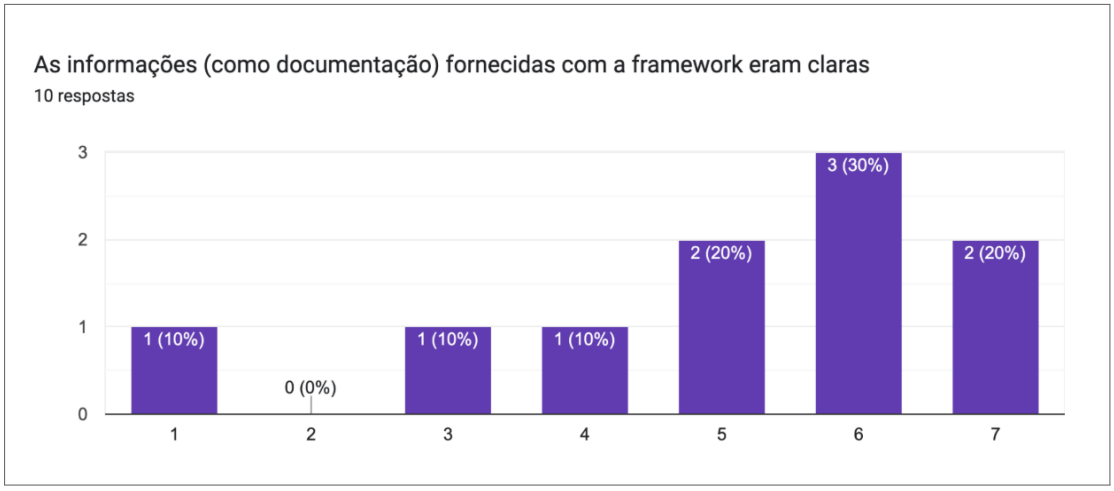
namento do sistema e suas funcionalidades. Além disso, percebe-se também algumas médias mais elevadas em pessoas mais experientes, mostrando a necessidade de melhorias na documentação. De um modo geral, a qualidade da informação pode ter sido impactada pela falta de uma abordagem diferenciada nas seções da documentação, que não conseguiu equilibrar a profundidade e a clareza necessária para cada nível de experiência.

Por fim, a dimensão "Qualidade da Interface" apresentou médias variando entre 1.0 e 3.0. No entanto, é evidente que a maioria das médias obtidas foi baixa, o que indica que a interface criada pelo *EyeTrackingViewBuilder* e as outras inferfaces conseguiram cumprir sua função de abstrair toda a complexidade na criação de telas controladas por *eye tracking*, independentemente do nível de experiência do usuário.

6.2.2.2 Triangulação de Dados - Análise das Categorias

A partir das entrevistas semiestruturadas e do PSSUQ, é possível identificar uma relação entre as dificuldades apontadas pelos participantes e os resultados das avaliações de usabilidade. A Tabela 20 revela que as dificuldades mais comuns estavam relacionadas à qualidade da informação fornecida pelo *framework*, com destaque para a falta de informações claras (categoria "Informações insuficientes"). Essas dificuldades foram reforçadas pelas avaliações do PSSUQ, que intensificaram os *insights* obtidos nas entrevistas. Isso é particularmente evidente na dimensão "Qualidade da Informação", onde os entrevistados, especialmente os iniciantes, apresentaram notas mais altas, o que indica uma percepção de menor qualidade. A Figura 21 mostra a distribuição das notas da documentação.

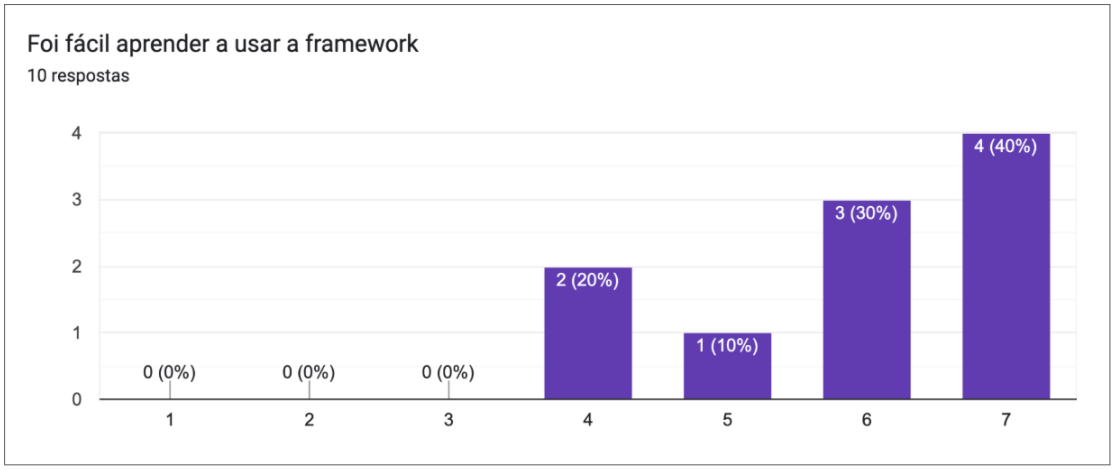
Figura 21 – Distribuição de notas sobre a documentação - Escala Likert Invertida



Fonte: Elaborado pelo autor - 2025.

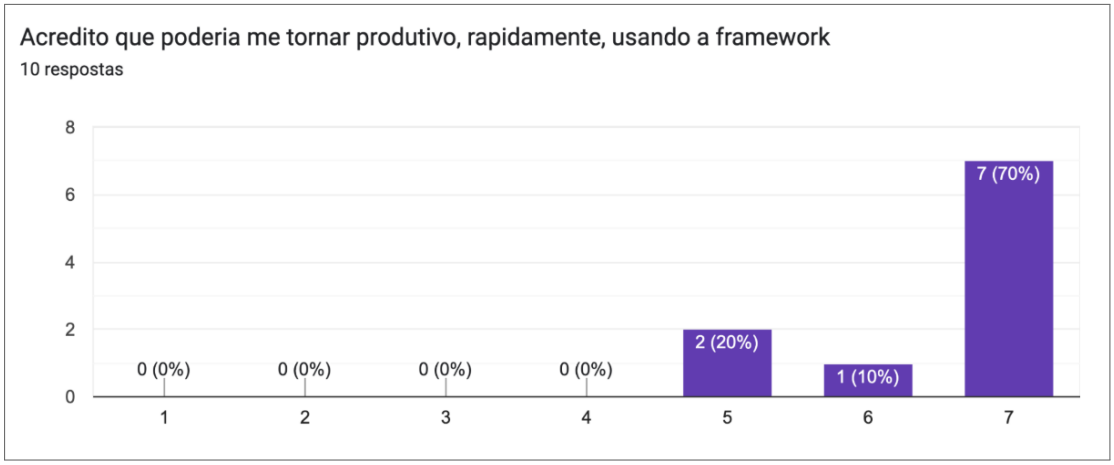
A triangulação dos dados revelou que a Categoria "Conceitualização" emergiu nas entrevistas semiestruturadas como um desafio inicial para os desenvolvedores. No entanto, a análise das pesquisas qualitativas confirmou que, apesar dessa dificuldade inicial, a assimilação dos novos conceitos ocorreu de forma relativamente fácil. Esse achado indica que, à medida que os desenvolvedores se familiarizam com os conceitos e a interface, a curva de aprendizado reduz significativamente, como ilustrado na Figura 22 e Figura 23.

Figura 22 – Facilidade de aprender - Escala Likert Invertida



Fonte: Elaborado pelo autor - 2025.

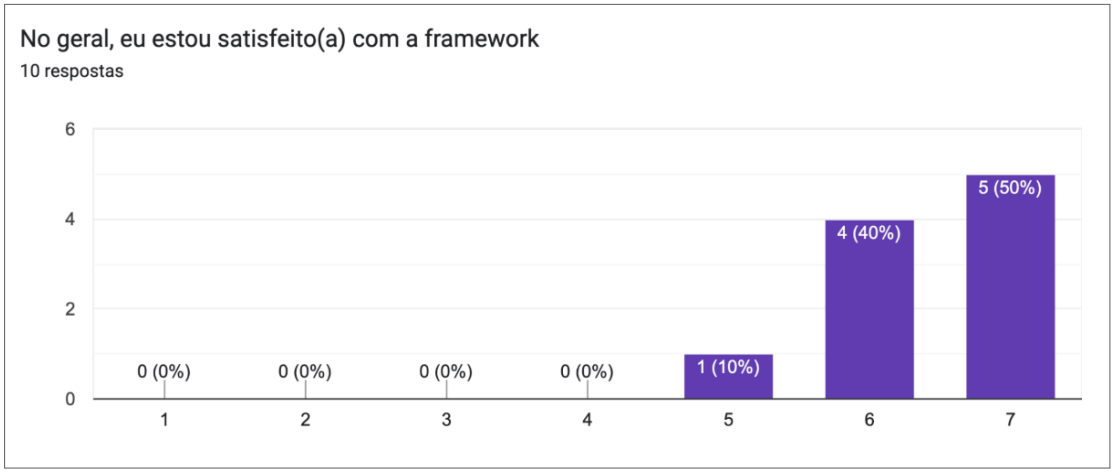
Figura 23 – Produtividade - Escala Likert Invertida



Fonte: Elaborado pelo autor - 2025.

Como se pode ver na Figura 24, a maioria dos respondentes expressou um alto nível de satisfação, indicando que o *framework* atendeu de forma satisfatória às suas expectativas e necessidades. Nas entrevistas semiestruturadas, também foi possível identificar um nível de satisfação positivo por parte dos desenvolvedores. Isso ficou evidente em comentários, como "Muito massa! Tu acha que vai rolar integrar com projetos existentes? Ia ser massa colocar isso lá onde trabalho" P(10). Esse tipo de resposta sugere que a solução proposta desperta curiosidade e expectativa quanto à sua aplicabilidade prática do *UITracking*, tanto em projetos pessoais quanto em projetos do trabalho. Embora existam áreas para melhorias, como mencionado por alguns desenvolvedores, a análise geral sugere que o *UITracking* cumpre bem seu papel de oferecer uma solução eficiente para a construção aplicações com *eye tracking*. Este *feedback* é valioso para as futuras iterações do *framework*, que podem se beneficiar de ajustes para aumentar ainda mais a satisfação dos usuários.

Figura 24 – Distribuição nível de satisfação - Escala Likert Invertida



Fonte: Elaborado pelo autor - 2025.

6.3 LIMITAÇÕES DO TRABALHO

Durante o desenvolvimento deste trabalho, se teve a preocupação de minimizar ao máximo a influência de vieses que pudessem comprometer, de alguma maneira, a imparcialidade dos resultados. Um viés contextual se dá quando fatores externos ou irrelevantes influenciam as respostas ou decisões. Nesse aspecto, ao utilizar uma abordagem de seleção convencional que prioriza contatos conhecidos e acessíveis, há a possibilidade de que as respostas durante as entrevistas não representem, de forma ampla, a diversidade de perspectivas existentes, limitando a generalização dos resultados obtidos [40].

Mesmo que a condução da pesquisa foi pautada pela busca por neutralidade e objetividade, é necessário reconhecer essas limitações supracitadas. Dessa forma, uma sugestão para trabalhos futuros é a **ampliação da amostra**, uma vez que um recrutamento ainda mais diversificado pode fornecer uma visão mais abrangente sobre os desafios e oportunidades na adoção do framework presente neste estudo.

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou a proposta de uma ferramenta que possibilita o uso de *eye tracking* para controlar *interfaces* em projetos UIKit. Ao analisar a ferramenta através de entrevistas, verificou-se o *UITracking* dentro da comunidade e a importância de ferramentas que promovam acessibilidade e inclusão no desenvolvimento iOS. Os resultados da análise qualitativa e das avaliações do PSSUQ demonstram que a usabilidade do *UITracking* é satisfatória, apesar dos desafios iniciais enfrentados pelos desenvolvedores. As dificuldades mais citadas estão relacionadas aos conceitos utilizados e documentação, especialmente a insuficiência das informações e à estruturação dos conteúdos. No entanto, a curva de aprendizado mostra uma tendência de melhoria conforme os desenvolvedores ganham familiaridade com o *UITracking*, ressaltando uma experiência positiva no desenvolvimento de aplicações controladas por *eye tracking*.

A categorização das dificuldades enfrentadas durante o uso da ferramenta permitiu identificar oportunidades claras de aprimoramento. Essas melhorias, organizadas em 6.1.2.2, abrangem desde ajustes técnicos na API até estratégias pedagógicas de aprendizado e comunicação mais efetiva com os desenvolvedores. Com isso, para trabalhos futuros, sugere-se a continuidade da evolução do *UITracking* com base nas sugestões levantadas, priorizando:

- **A ampliação e reorganização da documentação**, com foco em detalhamentos técnicos, exemplos progressivos de código e estruturação mais clara dos tópicos. Documentações mais completas e bem organizadas reduzem a curva de aprendizado e evitam erros comuns de uso.
- **Integração com projetos existentes,** oferecerendo suporte para a incorporação do framework em aplicações previamente desenvolvidas. Isso simplifica a adoção e reduz a necessidade de refatoração do código base.
- **A expansão das formas de interação disponíveis**, visando maior acessibilidade, como a personalização de expressões faciais para disparo de eventos e a navegação lateral baseada em áreas de interesse.

Por fim, este trabalho representa um passo inicial, mas significativo, na direção de tornar o desenvolvimento de interfaces acessíveis mais prático e integrado ao ecossistema iOS. A proposta do *UITracking* demonstrou seu potencial tanto em termos técnicos quanto em usabilidade, e os resultados obtidos reforçam a viabilidade de soluções baseadas em *eye tracking*

no contexto de aplicações UIKit. Espera-se que as contribuições aqui apresentadas incentivem novas iniciativas de pesquisa e desenvolvimento voltadas à acessibilidade digital, promovendo uma experiência mais inclusiva para todos os usuários. A continuidade dessa linha de investigação, aliada às melhorias sugeridas, pode consolidar o *UITracking* como uma ferramenta de referência no desenvolvimento de aplicações controladas pelo olhar.

REFERÊNCIAS

[1] ORGANIZATION, W. H.; (UNICEF), U. N. C. F. *Global Report on Assistive Technology*. Geneva: World Health Organization, 2022. Licence: CC BY-NC-SA 3.0 IGO.

[2] SHARAFI BONITA SHARIF, Y.-G. G. Z. A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering*, v. 25, 06 2020.

[3] KRAFKA, K. et al. *Eye tracking for everyone*. [S.l.]: arXiv preprint arXiv:1606.05814, 2016.

[4] TABAN, R. A.; CROOCK, D. M. S.; KORIAL, A. E. Eye tracking based mobile application. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, www.ijarcet.org, v. 7, n. 3, p. 246–250, 2018. ISSN 2278-1323.

[5] PELKA, B.; BüHLER, C. Empowerment by digital media of people with disabilities. three dimensions of support. In: . [S.l.: s.n.], 2014. ISBN 978-3-319-08595-1.

[6] VISHWAKARMA, P. et al. Classification of hci and issues and challenges in smart home hci implementation. In: _____. [S.l.]: Wiley, 2021. cap. 2, p. 23–61. ISBN 9781119791607.

[7] HOLMQVIST, K.; ANDERSSON, R. *Eye-tracking: A comprehensive guide to methods, paradigms and measures*. [S.l.]: Oxford University Press, 2017. ISBN ISBN-13: 978-1979484893.

[8] BUSJAHN BONITA SHARIF, T. et al. Eye tracking in computing education. In: *Proceedings of the Tenth Annual Conference on International Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2014. p. 3–10. ISBN 9781450327558. Disponível em: <https://doi.org/10.1145/2632320.2632344>.

[9] R LARANYA C R, H. C. P. R. S. K. K. S. Eye tracking and its applications. *International Advanced Research Journal in Science, Engineering and Technology (IARJSET)*, v. 8, n. 8, August 2021.

[10] KARKANIS, I. *Exploring Eye Tracking Techniques On Smartphones*. Dissertação (Mestrado) — Uppsala universitet, Institutionen för informationsteknologi, 2015. Disponível em: <http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-260665>.

[11] CHAKRABORTY DIPA ROY, M. Z. R. S. R. P. Eye gaze controlled virtual keyboard. *International Journal of Recent Technology and Engineering (IJRTE)*, v. 8, n. 4, November 2019. ISSN 2277-3878.

[12] PARSONS, T.; HORGAN, D.; MATTHEWS, J. Eye-tracking for accessibility in mobile devices. *Journal of Assistive Technology*, v. 8, n. 3, 2013.

[13] POOLE, A.; BALL, L. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In: GHAOUI, C. (Ed.). *Encyclopedia of Human-Computer Interaction*. [S.l.]: Idea Group Inc., 2006. p. 211–219.

[14] Apple Inc. *iOS 18 makes iPhone more personal, capable, and intelligent than ever*. 2024. Accessed: 2025-03-23. Disponível em: <https://www.apple.com/br/newsroom/2024/06/ios-18-makes-iphone-more-personal-capable-and-intelligent-than-ever/>.

[15] INC., A. *Focus and Selection*. 2025. https://developer.apple.com/design/human-interface-guidelines/focus-and-selection?changes=l_4_2_8_5__3. Acesso em: março 2025.

[16] W3C. *Understanding WCAG 2.0: Navigation Mechanisms*. 2025. https://www.w3.org/TR/UNDERSTANDING-WCAG20/navigation-mechanisms.html. Acesso em: março 2025.

[17] FREEMAN, A. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries*. [S.l.]: Addison-Wesley, 2015.

[18] INC., A. *Bundles and Frameworks*. 2025. https://developer.apple.com/documentation/xcode/bundles-and-frameworks. Acesso em: 17 mar. 2025.

[19] UBER. *New Rider App Architecture*. 2025. https://www.uber.com/en-BR/blog/new-rider-app-architecture/. Acesso em: 11 jan. 2025.

[20] CORMEN, T. H. et al. *Introduction to Algorithms*. 3rd. ed. [S.l.]: MIT Press, 2009.

[21] GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley, 1994.

[22] INC., A. *ARKit Documentation*. 2025. https://developer.apple.com/documentation/arkit. Acesso em: março 2025.

[23] INC., A. *ARFaceAnchor.BlendShapeLocation.eyeBlinkLeft*. 2025. `https://developer.apple.com/documentation/arkit/arfaceanchor/blendshapelocation/eyeblinkleft`. Acesso em: março 2025.

[24] REACTIVEX. *RxSwift | Documentation*. 2025. `https://reactivex.io/`. Acesso em: 15 jan. 2025.

[25] INC., A. *UIKit Documentation*. 2025. `https://developer.apple.com/documentation/uikit`. Acesso em: março 2025.

[26] DAMASCENA, A. *UITracking: iOS framework for eye tracking-based applications*. 2025. Acessado em 2025-03-22. Disponível em: <https://github.com/apfdamascena/UITracking>.

[27] BUCK, E.; YACKTMAN, D. Cocoa design patterns. In: _____. Boston, MA: Addison-Wesley Professional, 2009. cap. 15. ISBN 978-0321535023.

[28] MARTIN, R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Upper Saddle River, NJ: Prentice Hall, 2017. ISBN 978-0134494166.

[29] PHAN, D. M. *Command Bus Pattern*. 2020. Acessado em: 23 mar. 2025. Disponível em: <https://ducmanhphan.github.io/2020-12-02-command-bus-pattern/>.

[30] SOMMERVILLE, I. Software documentation. *Software engineering*, v. 2, p. 143–154, 2001.

[31] CRESWELL, J. W. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. [S.l.]: SAGE Publications, 2014.

[32] BRYMAN, A. *Social Research Methods*. 4. ed. [S.l.]: Oxford University Press, 2014.

[33] SAURO, J.; LEWIS, J. R. *Quantifying the User Experience: Practical Statistics for User Research*. 2nd. ed. [S.l.]: Morgan Kaufmann, 2016.

[34] TREND, U. U. *PSSUQ (Post-Study System Usability Questionnaire)*. 2025. Acesso em: 16 fev. 2025. Disponível em: <https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/>.

[35] BALTES, S.; RALPH, P. Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering*, Springer, v. 27, n. 4, p. 94, 2022.

[36] GUEST, G.; BUNCE, A.; JOHNSON, L. How many interviews are enough? an experiment with data saturation and variability. *Field Methods*, SAGE Publications, v. 18, n. 1, p. 59–82, 2006.

[37] SALDANA, J.; LEAVY, P. *Fundamentals of Qualitative Research*. 1st. ed. [S.l.]: Oxford University Press, 2011. (Understanding Qualitative Research).

[38] MYLLäRNIEMI, V. et al. Development as a journey: factors supporting the adoption and use of software frameworks. *Journal of Software Engineering Research and Development*, v. 6, n. 6, 2018. Disponível em: <https://doi.org/10.1186/s40411-018-0050-8>.

[39] Apple Inc. *Xcode Playgrounds*. 2024. `https://www.apple.com/br/swift/playgrounds/`. Acesso em: 09 abr. 2025.

[40] MURPHY, N. *Types of Bias*. 2025. Accessed: 2025-03-24. Disponível em: <https://cpdonline.co.uk/knowledge-base/safeguarding/types-of-bias>.

ANEXO A – TEMPLATE ENTREVISTA SEMIESTRUTURADA

Template Base - Entrevista Semiestruturada

Objetivo

O objetivo foi compreender como desenvolvedores iOS interagem com o framework proposto, avaliando aspectos como documentação, usabilidade e facilidade de adoção. Cabe destacar que essas constituíram as perguntas principais, mas outras questões complementares também foram incluídas com o intuito de aprofundar a análise.

1. Experiência da Pessoa Desenvolvedora

- Há quanto tempo você trabalha com desenvolvimento?
- Há quanto tempo trabalha (ou trabalhou) com iOS?
- Quantos projetos com UIKit você já teve oportunidade de desenvolver?
- Como costuma aprender sobre novas tecnologias ou frameworks?
- Quais fatores tornam esse processo de aprender novos frameworks mais fácil ou difícil para você?

2. Contextualização do Framework

- Como você avalia a documentação fornecida?
- Quais aspectos poderiam melhorar para facilitar o aprendizado?
- Há algo que poderia ser melhorado para facilitar a compreensão inicial ?

3. Execução de Tarefas (Iguais para todos os participantes)

1. Construir uma tela com duas seções.
2. Com base no desafio anterior, torne a view da primeira seção interativa, adicionando um evento que exiba uma mensagem no console **(print)**
3. A partir do desafio anterior, modificar o evento para fazer uma navegação entre telas. A nova tela deve ter três seções com duas views cada.

3.1 Desafios e Dificuldades para cada Tarefa

- Como foi sua experiência ao executar as tarefas propostas?
- Você sentiu que a API era intuitiva?
- A documentação te ajudou a completar o desafio?
- Houve algum momento em que você sentiu dificuldades em entender como utilizar um determinado recurso do framework?

4. Desafios e Dificuldades - Geral

- Quais foram os principais desafios que você enfrentou ao interagir com o framework? Foram desafios mais conceituais ou técnicos?
- Houve alguma inconsistência ou comportamento inesperado na API? Poderia dar um exemplo?
- Comparado a outras ferramentas que você já utilizou, quais foram os pontos mais difíceis ao trabalhar com esse framework?

5. Pontos de Melhoria

- Qual aspecto do framework como um todo você acredita que poderia ser melhorado?

ANEXO B – CATEGORIZAÇÃO DOS CÓDIGOS POR NÍVEL

Pessoa	Nível	In Vivo Coding	Categoria
P9	Avançado	facilitaria reproduzir designs que são enraizados da Apple	Capacidades da API
P9	Avançado	implementar outros gestos para usar, por exemplo abrir a boca	Capacidades da API
P10	Avançado	é possível integrar com projetos já existentes?	Capacidades da API
P9	Avançado	seção para mostrar como instanciar o ViewController	Informações Insuficientes
P9	Avançado	addSubview ter uma view para exemplificar melhor	Informações Insuficientes
P9	Avançado	não ter como mudar as cores da borda	Informações Insuficientes
P9	Avançado	mostrar na documentação exemplos com eventos	Informações Insuficientes
P8	Intermediário	mostrar na documentação exemplos com eventos também	Informações Insuficientes
P6	Intermediário	demorei para entender o super.init	Informações Insuficientes
P6	Intermediário	métodos em EyeTrackingView não estão documentados	Informações Insuficientes
P5	Intermediário	o EyeTrackingView não foi tão intuitivo porque não tinha na documentação	Informações Insuficientes
P5	Intermediário	não tem os parâmetros que ela recebe na documentação	Informações Insuficientes
P4	Intermediário	colocar na documentação alguns comentários	Informações Insuficientes
P3	Intermediário	não sabia de cara o que o builder iria devolver	Informações Insuficientes
P3	Intermediário	não tinha documentação do EyeTrackingView	Informações Insuficientes
P1	Iniciante	traduzir a documentação para português	Informações Insuficientes
P1	Iniciante	deixar claro que UITracking é o nome da tua ferramenta	Informações Insuficientes
P1	Iniciante	o que é preciso para funcionar, por exemplo macos, dispositivos	Informações Insuficientes
P1	Iniciante	senti falta de mais documentação do EyeTrackingView	Informações Insuficientes
P2	Iniciante	Não deu pra entender necessariamente que é pra empilhar verticalmente	Informações Insuficientes
P7	Intermediário	code example precisa de uma conclusão	Informações Insuficientes
P9	Avançado	ficaria melhor se tivesse um vídeo	Exemplos de códigos e vídeos
P9	Avançado	adicionar uma view simples, esse é o código que deveria está lá	Exemplos de códigos e vídeos
P8	Intermediário	código mais simples e que funcione	Exemplos de códigos e vídeos
P8	Intermediário	deixa um código de exemplo mais simples	Exemplos de códigos e vídeos
P4	Intermediário	seria legal colocar um exemplo de tela real	Exemplos de códigos e vídeos
P4	Intermediário	mais exemplos de como utilizar o evento	Exemplos de códigos e vídeos
P7	Intermediário	a ausencia de mais exemplos dificulta meu processo	Exemplos de códigos e vídeos
P2	Iniciante	seria legal pra mim se tivesse video	Exemplos de códigos e vídeos
P8	Intermediário	mente bugou porque tem conceito de view tree em How to Use	Informações não direcionadas e não organizadas
P8	Intermediário	colocaria o conceito de View Tree para o final	Informações não direcionadas e não organizadas
P5	Intermediário	to confusa com o título do How to use, porque apresenta um conceito de início	Informações não direcionadas e não organizadas
P8	Intermediário	confuso entender o evento	Conceitualização
P7	Intermediário	tive dificuldade em entender o que o Builder faz	Conceitualização

ANEXO C – CATEGORIZAÇÃO DOS CÓDIGOS POR NÍVEL

Pessoa	Nível	In Vivo Coding	Categoria
P7	Intermediário	entender event handle e command bus	Conceitualização
P6	Intermediário	nunca fiz nada parecido de ter uma classe que registra um evento	Conceitualização
P6	Intermediário	não tive contato com o conceito de command bus anteriormente	Conceitualização
P1	Iniciante	não entendi muito bem o commandBus não	Conceitualização
P1	Iniciante	não entendi bem o evento	Conceitualização
P8	Intermediário	dificuldade inicial foi porque eu não lembrava de UIKit	Experiência em iOS
P7	Intermediário	o gap principal seria a questão do UIKit em si	Experiência em iOS
P4	Intermediário	Senti maior dificuldade com UIKIt mesmo	Experiência em iOS
P1	Iniciante	UIKit me complicou demais	Experiência em iOS
P2	Iniciante	ainda estou me desenvolvendo em UIKit	Experiência em iOS
P8	Intermediário	adicionar na documentação como interagir	Interações com dispositivo
P8	Intermediário	não tinha uma contextualização na documentação, tipo de piscar o olho	Interações com dispositivo
P7	Intermediário	tive dificuldade em entender as interações do usuário	Interações com dispositivo
P6	Intermediário	interações no geral, como seleciona, como volta a tela,	Interações com dispositivo
P4	Intermediário	falta de direcionamento das interações possíveis	Interações com dispositivo
P10	Avançado	não ficou claro sobre as interações que o usuário faz	Interações com dispositivo
P10	Avançado	Essa parte de voltar olhando pra esquerda seria massa ter um user guide	Interações com dispositivo
P1	Iniciante	você é iniciante, nunca utilizou tal coisa. ter um comece por aqui.	Aprendizado Interativo
P1	Iniciante	seria massa ter uma documentação que vai conversando com você	Aprendizado Interativo
P2	Iniciante	se tivesse um passo a passo mais dividido seria mais fácil	Aprendizado Interativo