INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Parede Interativa Controlada por Voz

GONÇALO LUÍS RODRIGUES DA SILVA
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Parede Interativa Controlada por Voz

Gonçalo Luís Rodrigues da Silva

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática, Área de Especialização em

Sistemas Gráficos e Multimédia

Orientador: Prof. Doutor António Vieira de Castro

Porto, outubro de 2022

Resumo

A projeção interativa sobre superfícies como paredes resulta de imagens projetadas sobre as mesmas, independentemente de se tratar de uma superfície regular ou irregular, e reage normalmente à interação dos utilizadores. Trata-se do recurso a tecnologias contemporâneas, na maioria das vezes baseada em sensores de movimento e câmaras, sendo muito utilizadas em eventos ou personalização de ambientes.

Ainda que as paredes interativas sejam altamente personalizáveis, foi possível constatar que ainda existem tipos de interação, nomeadamente a interação controlada pela voz, que não estão suficientemente explorados. O principal problema identificado é a necessidade de encontrar mecanismos que permitam proporcionar a interação destes ambientes, não apenas por gestos, mas também por comandos de voz.

Assim, foi realizado um estudo e desenvolvida uma solução protótipo que permita colmatar as lacunas identificadas, assim como provar a viabilidade de uma solução desta natureza. A solução desenvolvida permite criar e animar objetos através de comandos de voz, assim como uma série de edições e funcionalidades de gestão de projetos. Tudo isto utilizando tecnologias contemporâneas, que permitam uma distribuição e instalação simples e com uma grande compatibilidade de equipamentos.

Palavras-chave: parede interativa, projeção interativa, reconhecimento de voz

Abstract

Interactive projection on surfaces such as walls results from images projected onto them, regardless of whether it is a regular or irregular surface, and it usually reacts to the user's interaction. It is the use of contemporary technologies, most often based on motion sensors and cameras, being widely used in events or personalization of environments.

Although interactive walls are highly customizable, it was possible to verify that there are still types of interaction, namely voice-controlled interaction, which is not sufficiently explored. The main problem identified is the need to find mechanisms that allow interaction in these environments, not only through gestures, but also through voice commands.

Thus, a study was carried out and a prototype solution was developed that allows filling the identified gaps, as well as proving the feasibility of a solution of this nature. The developed solution allows the creation and animation of objects through voice commands, as well as several editions and project management features. All this using contemporary technologies, which allows for a simple distribution and installation and with a great compatibility of equipment.

Keywords: interactive wall, interactive projection, speech recognition

Agradecimentos

Em primeiro lugar, quero agradecer ao meu orientador, professor António Castro, pelo convite para pertencer ao SIIS e pelo desafio proposto, mas também por todo o apoio dado ao longo deste projeto.

Ao ISEP (Instituto Superior de Engenharia do Porto) e a todos os docentes e colegas por estes seis anos de percurso académico, que passaram a voar.

Ao SIIS (Sistemas de Interação e Inovação Social) pela disponibilização dos equipamentos e do espaço que permitiram que este projeto se concretizasse.

Por fim, mas não menos importante, a todos os colegas, amigos e familiares, em especial à minha mãe, que de alguma forma contribuíram para o sucesso deste projeto.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AHP	*Analytic Hierarchy Process*
API	*Application Programming Interface*
CRT	*Cathode Ray Tube*
CSS	*Cascading Style Sheets*
FEI	*Front End of Innovation*
HCI	*Human-Computer Interaction*
HTML	*HyperText Markup Language*
ISEP	Instituto Superior de Engenharia do Porto
LCD	*Liquid Crystal Display*
LED	*Light-Emitting Diode*
NCD	*New Concept Development*
NPM	*Node Package Manager*
PNG	*Portable Network Graphics*
SDK	*Software Development Kit*
SIIS	Sistemas de Interação e Inovação Social
UI	*User Interface*
URL	*Uniform Resource Locator*
Wasm	*WebAssembly*

1 Introdução

A projeção interativa sobre superfícies como paredes resulta de imagens projetadas sobre as mesmas, independentemente de se tratar de uma superfície regular ou irregular, e reage normalmente à interação dos utilizadores. Trata-se do recurso a tecnologias contemporâneas, na maioria das vezes baseada em sensores de movimento e câmaras, sendo muito utilizadas em eventos ou personalização de ambientes.

Um grupo de investigação recentemente criado denominado SIIS pretende colocar nas suas instalações uma sala interativa controlada por voz para explorar o potencial deste tipo de soluções no mercado, como museus ou outros espaços.

1.1 Problema

Ainda que as paredes interativas sejam altamente personalizáveis, foi possível constatar que existem tipos de interação, nomeadamente a interação controlada pela voz, que não estão suficientemente explorados. O principal problema identificado é a necessidade de encontrar mecanismos que permitam proporcionar a interação destes ambientes, não apenas por gestos, mas também por comandos de voz.

A questão da interação por voz com superfícies planas e projeções controladas pelo utilizador é ainda uma área por explorar, assim como o seu potencial que, através dos comandos de voz, enviam ordens para um software que os transforme em interações nesse piso ou parede.

1.2 Objetivos

No presente estudo, pretende-se fazer um levantamento de tecnologias existentes com potencial para projeção interativa, identificar exemplos de uso e sobretudo explorar uma área paralela que acrescente, aos mecanismos existentes, a interação controlada por comandos de voz.

Pretende-se assim realizar um estudo relacionado com a projeção para superfícies, sejam estas paredes, chãos ou tetos, e desenvolver um protótipo que permita ao utilizador o controlo dessa projeção através de interação por voz de um dado espaço físico.

Pretende-se identificar tecnologias e meios para que, através da voz, seja possível proporcionar a um utilizador o controlo do que é projetado nas superfícies de um espaço físico disponibilizado para o efeito, tornando-as, assim, dinâmicas.

O utilizador deverá poder acrescentar objetos como círculos, triângulos, quadrados ou imagens através de comandos de voz e criar animações e ações sobre esses objetos como rodar para a esquerda, para a direita, aumentar ou diminuir de tamanho ou mudar a cor, de forma que a conjugação destes elementos projetados nas paredes do espaço permitam criar um conceito artístico, mas paralelamente interativo e controlado pelo utilizador.

Deverão ainda ser estudadas alternativas de interação complementares com potencial para complementar o protótipo com interação por voz.

Para o efeito, será necessário:

1. Analisar soluções existentes;
2. Identificar tecnologias com potencial para o desenvolvimento;
3. Validar aspetos técnicos e funcionais para ajudar a resolver o problema;
4. Desenvolver e implementar a solução protótipo;
5. Testar a aplicação;
6. Avaliar alternativas de interação complementares.

1.3 Motivação

Este projeto surgiu pela necessidade por parte do SIIS de possuir uma parede interativa nas suas instalações. Ao mesmo tempo, foram detetadas duas lacunas nas projeções interativas: a

falta de soluções que permitam a interação por voz e que permitam ao utilizador personalizar a sua parede em tempo real.

As áreas de estudo onde se inserem as paredes interativas e a interação por voz, assim como tudo o que as rodeia, são também áreas pelas quais tenho muito interesse.

Por fim, o facto de prosseguir com a obtenção de um novo grau no meu percurso académico foram também fatores importantes para o desenvolvimento do presente estudo.

1.4 Estrutura

Este documento é composto por sete capítulos, nos quais se inclui o presente capítulo.

No segundo capítulo é apresentado o estado da arte das paredes interativas, sendo estudados os tipos de projeção e de interação. É também estudado o reconhecimento de voz. Por fim, são analisados e comparados os trabalhos relacionados e as tecnologias existentes.

No terceiro capítulo é apresentada a análise de valor, sendo identificada e analisada a oportunidade e gerada e selecionada a ideia. É também definido o valor da solução, estudando a proposta de valor e o modelo de negócio.

No quarto capítulo é apresentado o design da solução, começando pelo equipamento e instalação que será usado para testar a aplicação. Em seguida, é apresentado o modelo de domínio. E por fim, são apresentados e analisados os casos de uso, assim como os requisitos funcionais e não funcionais.

No quinto capítulo é apresentada a implementação da solução. Em concreto, são analisadas as tecnologias efetivamente utilizadas, assim como a estrutura do código da solução. Por fim, é analisada a implementação de cada caso de uso.

No sexto capítulo, é apresentada a avaliação da solução e em concreto como será avaliada a solução após o seu desenvolvimento. É ainda apresentado o questionário de usabilidade e analisados os resultados derivados desse questionário.

No sétimo e último capítulo, é feito um resumo de todo o trabalho realizado com um olhar para os sucessos e insucessos, assim como para o trabalho futuro.

2 Estado da arte

Neste capítulo é analisado o estado da arte do problema proposto. Tendo em consideração as características desta solução, é dado um maior destaque à componente de *software* e em especial ao reconhecimento de voz, abordando também o *hardware* utilizado mais tipicamente de forma a contextualizar como ambas as componentes se interligam. Por fim, são analisadas as tecnologias propostas para a solução, comparando as suas funcionalidades, desempenho e popularidade.

2.1 Paredes interativas

Uma parede interativa, também conhecida por projeção interativa, pode assumir uma série de definições: desde o estilo de superfície ao tipo de projeção e interação. Por norma, uma parede interativa utiliza uma superfície, regular ou irregular, que pode ou não ser uma parede real. Mas ao contrário de uma projeção normal, a projeção interativa reage e modica-se de acordo com a interação dos utilizadores (Amorim, 2018).

Uma questão interessante a analisar, e que levou em grande parte à conceção desta solução, é o facto de as várias paredes interativas existentes apenas permitirem o controlo por gestos ou toques e, especialmente, não permitirem a personalização do que está a ser apresentado.

Recentemente, foi instalada uma sala interativa no Aquário Vasco da Gama, chamada de "Janela para o Oceano". O espaço tem 20 metros quadrados e possui um dos maiores ecrãs interativos da Península Ibérica. Esta obra teve um custo de 300 mil euros e foi pensado em especial para as crianças (Meireles, 2021).

Na Figura 1 é possível ver a sala em questão, que é um excelente exemplo do que existe atualmente no mercado das paredes interativas, em particular em Portugal.

Figura 1 Sala interativa do Aquário Vasco da Gama[1]

2.2 Tipos de interação

No que diz respeito aos tipos de interação, e tal como referido, as mais comuns são a interação por toque e por gestos. Estes tipos de interação podem, no entanto, assumir uma série de formas: desde um simples toque com a mão ou um gesticular de braços, a caminhar ou saltar com os pés.

Estes tipos de interação incluem-se na área de estudo HCI (*Human-Computer Interaction*). Ainda que a interação humano-computador seja muito mais do que se vai analisar no presente estudo, é possível pelo menos resumir as interações necessárias às de entrada e dentro destas às por toque, gestos e voz (Dix, 2022).

2.2.1 Toque

A interação por toque é um tipo de interação que é muito utilizada atualmente em dispositivos com ecrãs táteis, como os telemóveis. Esta envolve por norma um toque com o dedo numa determinada superfície ao qual esta reage realizando uma ação.

[1] Imagem retirada de https://www.dn.pt/local/aquario-vasco-da-gama-abre-sala-interativa-para-atrair-mais-criancas-14411996.html

Na Figura 2 é possível ver um exemplo de uma parede com interação por toque. Neste caso, ao tocar no interruptor, a luz acende-se.

Figura 2 Exemplo de uma parede interativa por toque[2]

2.2.2 Gestos

A interação por gestos, também muito semelhante ao utilizado nos dispositivos por toque, envolve um movimento, por norma do dedo no caso de existir um toque prévio ou do braço no caso de não existir contacto com a parede.

Na Figura 3 é possível ver precisamente essa interação por gestos. O utilizador ao mover o seu braço, afasta as garrafas e revela uma publicidade.

Figura 3 Exemplo de uma parede interativa por gestos[3]

[2] Imagens retiradas de https://www.youtube.com/watch?v=BrqfPzhSYys
[3] Imagem retirada de https://www.youtube.com/watch?v=AZA6X3mPdtg

Existem uma série de equipamentos que permitem a deteção e reconhecimento de gestos. Desde uma simples câmara a equipamentos mais complexos desenvolvidos com o único propósito de reconhecer gestos.

No lado das câmaras, é possível utilizar praticamente qualquer câmara *web* ou até de telemóvel. Existem, no entanto, outros equipamentos mais avançados e com maior precisão de deteção graças à utilização de, por exemplo, sensores infravermelhos.

Do lado dos equipamentos especializados, existem também um leque bastante diversificado de soluções. A mais simples são, por exemplo, luvas que permitem a deteção e reconhecimentos dos gestos realizados por um determinado utilizador. Existem outros equipamentos, mais modernos e sem contacto, que através de sensores óticos, permitem também a deteção e reconhecimentos de gestos.

2.2.3 Posicionamento

Muito semelhante à interação por gestos, a interação por posicionamento difere da anterior principalmente por envolver movimento por parte do utilizador, ou seja, enquanto um gesto envolve normalmente um movimento de braço, a interação por posicionamento envolve também deslocação. Por norma, esta interação ocorre nas projeções realizadas no chão e que necessitam de determinar a posição para realizar uma determinada ação.

Na Figura 4 é demonstrada uma projeção interativa no chão por posicionamento. Neste caso, à medida que o utilizador caminha, vai afastando as caras e revelando uma frase.

Figura 4 Exemplo de uma projeção interativa por posicionamento[4]

2.2.4 Voz

Já a interação por voz difere significativamente das anteriores no sentido que não envolve nenhum contacto ou movimento físico. Sendo a voz o principal modo de interação da solução em estudo, esta será analisada com maior detalhe em 2.4.

2.3 Tipos de projeção

No que diz respeito aos tipos de projeção, é possível observar que, ainda que na essência os métodos usados são semelhantes há muitos anos, inovações mais recentes, especialmente no que toca à tecnologia LED (*Light-Emitting Diode*), permitem montar sistemas nunca imaginados.

Assim, é possível definir os seguintes tipos de projeção: sistemas de projeção (projetores) e ecrãs (CineMassive, 2022) (Swami, 2022).

2.3.1 Projeção frontal

Este tipo de projeção é o mais comum no que toca ao uso de projetores convencionais. Um local muito comum onde é possível ver este tipo de projeção é nos cinemas.

[4] Imagem retirada de https://www.youtube.com/watch?v=AZA6X3mPdtg

Este tipo de projeção pode utilizar múltiplos projetores em simultâneo de forma a criar uma maior área de projeção, que é depois misturada de forma a unir todas as projeções numa única imagem.

A Figura 5 mostra um exemplo de projeção frontal. Neste caso, o projetor está atrás do ator, mas ainda assim à frente da tela.

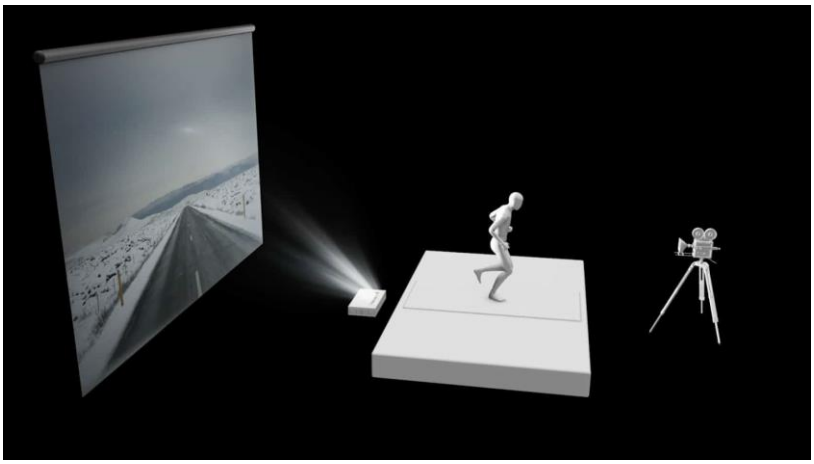


Figura 5 Exemplo de projeção frontal[5]

2.3.2 Projeção traseira

A projeção traseira (*rear projection*) é uma técnica muito semelhante à projeção frontal, no entanto, tal como próprio nome indica, é realizada por trás da tela em questão.

Tal como na projeção frontal, é possível utilizar múltiplos projetores e assim unir várias projeções numa única imagem.

Também é comum neste tipo de projeção, e com o intuito de conservar espaço, utilizar um sistema de projeção e um espelho envoltos num cubo. Essencialmente, o projetor emite a luz para o espelho que depois é refletido para a tela. Esta técnica tem também a grande vantagem de permitir melhores níveis de brilho e contraste.

Existiram também televisores que utilizavam esta tecnologia, muito populares no início da década de 2000. Estes surgiram da necessidade de criar televisores com tamanhos maiores aos que eram possíveis criar com a tecnologia CRT (*Cathode Ray Tube*), sem aumentar

[5] Imagem retirada de https://www.filmriot.com/blog/rear-front-projection/

consideravelmente a profundidade e o peso. No entanto, com a queda dos preços das tecnologias plasma e LCD (*Liquid Crystal Display*), estes acabaram por perder a maioria das vantagens que os distinguiam.

A Figura 6 mostra um exemplo de projeção traseira.

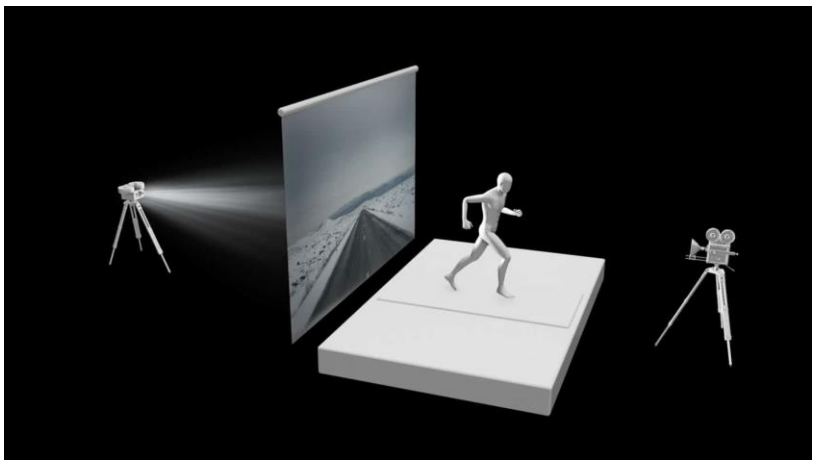


Figura 6 Exemplo de projeção traseira[6]

2.3.3 Ecrãs LCD

Este tipo de projeção utiliza os convencionais ecrãs LCD. Os ecrãs LCD tiram partido das propriedades dos cristais líquidos para, em conjunto com polarizadores, criarem uma imagem. Os ecrãs LCD por si só não emitem luz, daí ser preciso retroiluminar o painel para obter cor.

É atualmente uma das soluções com um melhor custo-benefício, mas com a grande desvantagem de não ser possível criar ambientes unidos, ou seja, existe sempre uma separação entre cada ecrã. No entanto, em muitos cenários, e dependendo do tamanho do ecrã em questão, isto pode não ser um problema.

Na Figura 7 é possível ver o estúdio de informação da TVI que faz uma utilização extensiva de ecrãs LED (à esquerda com o planeta), mas também de ecrãs LCD (à direita com o logotipo J8).

[6] Imagem retirada de https://www.filmriot.com/blog/rear-front-projection/

Figura 7 Estúdio de informação da TVI em setembro de 2020[7]

É interessante salientar que os ecrãs LED em nada têm a ver com os televisores LED que são normalmente comercializados. Esses televisores usam na mesma tecnologia LCD, com a diferença de que a luz que ilumina o painel é LED.

2.3.4 Ecrãs LED

Este tipo de projeção utiliza píxeis LED independentes e tem a grande diferença de permitir criar ecrãs gigantes sem ser possível distinguir entre os vários módulos que os compõem. É muito utilizado atualmente nos estúdios de televisão e cinema e é uma alternativa moderna muito popular, ainda que significativamente mais cara, ao tradicional ecrã verde (*green screen*) (Edwards, 2020).

Ao contrário do que por norma se pode pensar, esta tecnologia existe há já muitos anos (desde a década de 1960). Dado que são extremamente brilhantes, são também muito usados para apresentar informações ao público, tal como nos transportes públicos, aeroportos e até autoestradas. Na Figura 8 é possível ver um painel LED com os próximos destinos numa estação do metro de Londres.

[7] Imagem retirada de https://www.atelevisao.com/tvi/conheca-novo-estudio-informacao-tvi/

Figura 8 Painel LED de informação ao público

2.4 Reconhecimento de voz

O reconhecimento de voz é o passo mais importante no que toca à interação por voz. Para o seu funcionamento, é necessário que exista um dispositivo de entrada, por norma um microfone, que depois seja interpretado por um determinado dispositivo ou *software*. Dada a natureza analógica do som, é necessária uma conversão das ondas para sinais digitais. Estes podem ser então interpretados e comparados com base em reconhecimento de padrões pré-programados (Scardina, 2018).

Esta é uma das razões pela qual é tão importante para as empresas que possuem produtos com reconhecimento de voz, como as assistentes virtuais, que os dados que o utilizador fornece sejam utilizados para melhorar o serviço. Cada vez que um utilizador diz algo, e este algo é reconhecido como sendo correto, é possível utilizar essas informações para melhorar o serviço. Isto é especialmente importante nos casos dos sotaques e dialetos, que podem não ser reconhecidos à partida pelo reconhecimento de voz.

Na Figura 9 é possível ver precisamente o fluxo padrão de um reconhecimento de voz. O utilizador emite áudio analógico (a fala), que é posteriormente convertido para um sinal digital de forma que o sistema em questão possa comparar com os padrões conhecidos. Por fim, o sistema pode emitir ou responder o resultado esperado.

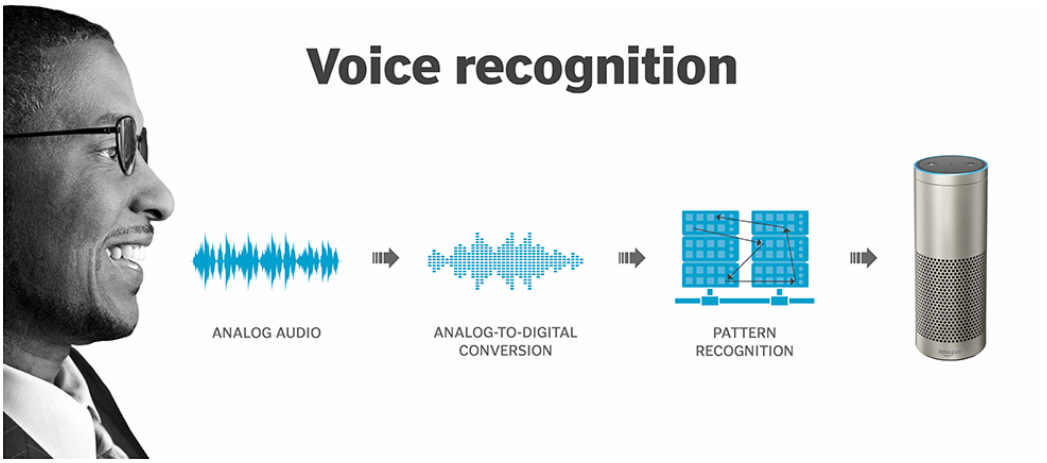


Figura 9 Processo de reconhecimento de voz[8]

2.4.1 Evolução

Nos últimos anos assistiu-se uma evolução exponencial nesta área, motivado especialmente pelo advento da inteligência artificial e do grande poder computacional que permite processar quantidades impressionantes de dados (Sonix, 2022).

Os primeiros sistemas de reconhecimento de voz surgiram na década de 1950, sendo que um dos mais conhecidos é o Audrey da Bell Laboratories. Este era apenas capaz de reconhecer dígitos de forma individual.

Na década de 1960, surgiu o Shoebox da IBM já capaz de compreender dezasseis palavras em inglês.

Na década de 1970, já existiam sistemas capazes de reconhecer mil palavras, o vocabulário equivalente ao de uma criança de três anos. É nesta altura também que surgem os primeiros sistemas capazes de interpretar várias vozes.

Nas décadas de 1980 e 1990, a evolução do reconhecimento de voz foi muito linear. Estes sistemas passaram a ser capazes de reconhecer um número muito maior de palavras, o que se deveu em grande parte à evolução muito significativa dos processadores e computadores pessoais da época.

[8] Imagem retirada de https://searchcustomerexperience.techtarget.com/definition/voice-recognition-speaker-recognition

Na primeira década do século XXI, os sistemas de reconhecimento de voz tinham já atingido taxas de precisão próximas dos 80%. Foi nesta década que surgiu o *Google Voice Search* que colocou o reconhecimento de voz nas mãos de milhões de pessoas. Isto trouxe uma evolução muito importante, uma vez que passaram a ser recolhidos dados de milhares de milhões de pesquisas que permitiram melhorar significativamente a precisão do reconhecimento de voz.

Na década de 2010, a principal conquista foram as assistentes virtuais. Graças a estas, cada vez mais utilizadores tiveram contacto com o reconhecimento de voz, o que ajudou bastante a acabar com o estigma de falar com uma máquina. Foi nesta década também que se atingiram taxas de erro mínimas, com a IBM afirmar uma taxa de 5.5% e a Google de 4.9%.

2.4.2 Tendências e aplicações

É inegável que atualmente as tecnologias de reconhecimento de voz evoluíram de uma maneira exponencial. Se há uns anos até a simples tarefa de falar para uma linha de apoio automática era um processo falível e frustrante, agora é possível ter uma conversa perfeitamente normal com uma assistente e até fazer com que esta só reconheça a própria voz do utilizador (Dolbey, 2022).

Outro exemplo muito atual de utilização de comandos de voz são as assistentes virtuais. As assistentes virtuais são agentes que podem interpretar o discurso humano e responder através de vozes sintetizadas. Atualmente, as assistentes mais populares são a Google Assistant, a Alexa da Amazon e a Siri da Apple. Estas assistentes estão presentes numa vasta gama dispositivos, onde se incluem telemóveis, relógios, televisores, colunas de som, e até interruptores e lâmpadas (estes últimos requerem, por norma, um dispositivo auxiliar para a interação por comandos de voz) (Amazon, 2022) (Google, 2022) (Apple, 2022).

2.4.3 Microfones

O microfone é um dispositivo capaz de capturar áudio analógico (a voz, por exemplo) e convertê-lo num sinal elétrico. Estes estão presentes numa série de dispositivos e são utilizados para uma série de aplicações, desde telefones e telemóveis à rádio e televisão. Existem dois tipos de microfones relevantes para o estudo desta solução e que se distinguem precisamente pelas características de captura do som: microfones dinâmicos e microfones condensadores (Nugent, 2021) (Hahn, 2019) (Levine, 2022).

2.4.3.1 Microfones dinâmicos

Os microfones dinâmicos são o tipo mais comum de microfones. Estes utilizam um diafragma ligado a uma bobina de indução. Ao vibrar o diafragma, a bobina move-se no campo magnético gerando uma corrente elétrica.

Estes microfones possuem um padrão cardioide (em forma de coração) polar, isto é, o microfone capta o som a partir da direção para o qual é apontado e cancela qualquer som vindo de outras direções.

São ideais para fontes de som ruidosas, tais como instrumentos musicais e concertos. Graças ao seu tipo de construção interna, são também muito mais resistentes que o seu homólogo.

2.4.3.2 Microfones condensadores

Os microfones condensadores são mais sofisticados que os dinâmicos, o que aumenta o custo e reduz a sua resistência. Quando comparados aos microfones dinâmicos, estes são muito mais precisos e balanceados, ainda que sejam muito mais sensíveis, o que os torna ideais para sons mais suaves, como é o caso da voz.

Este tipo de microfone utiliza um diafragma metálico posicionado na frente de uma placa também metálica. Ao mover o diafragma, a distância entre a placa varia, alterando a capacitância para produzir um sinal elétrico. O seu funcionamento é muito semelhante ao de um condensador (*capacitor*) encontrado em dispositivos eletrónicos. O tamanho do diafragma impacta a frequência do som capturado (quanto maior o diafragma, menor é a gama de frequências capturada).

Outra vantagem deste tipo de microfones é o seu leque de padrões cardioides. Para além deste tipo de padrões, é possível ainda escolher padrões omni ou bidirecionais. O padrão polar bidirecional capta o som de frente e de trás cancelando o ruído vindo das laterais. Já o padrão polar omnidirecional capta som em todas as direções não cancelando nenhum ruído.

2.5 Trabalhos relacionados

Uma das características que é interessante observar é que, apesar de existirem bastantes soluções no mercado para projeções interativas, estas são sempre anunciadas como *hardware*. Ou seja, ainda que exista uma componente de *software* inerente, o *hardware* é que é o

componente anunciado e vendido. Com isto, nem sempre é possível comparar a solução de *software* a desenvolver neste estudo com o *software* vendido junto com as paredes interativas tradicionais. Ainda assim, foi possível encontrar uma solução com especificações semelhantes: o LUMOplay.

2.5.1 LUMOplay

O LUMOplay é um software de projeção da canadiana Lumo Interactive. É utilizado por grandes empresas, como a McDonald's, Adidas, Google, entre outras (Lumo Interactive, 2022).

Tem a particularidade de, tal como a solução do presente estudo, não necessitar de hardware específico para funcionar. O utilizador pode conceber o seu próprio sistema, sendo que também é possível adquirir um *kit* já pré-concebido.

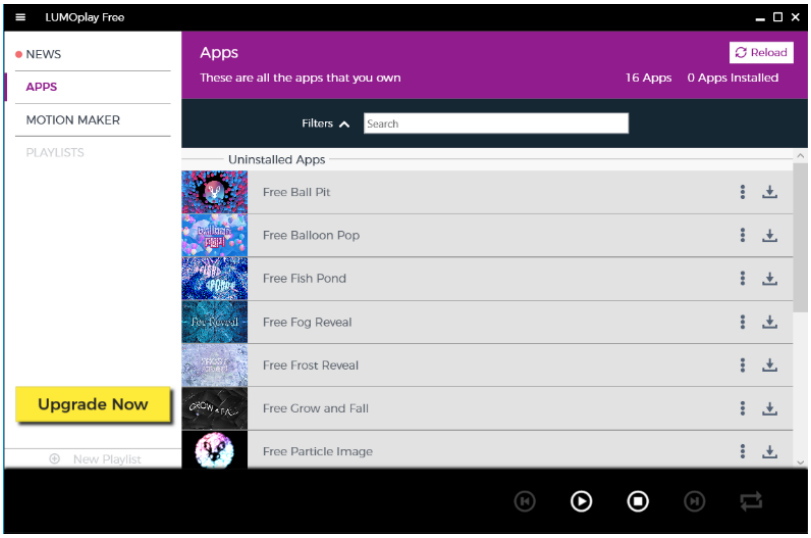


Figura 10 Interface de utilizador da versão gratuita do LUMOplay

Na Figura 10, é possível ver a interface de utilizador principal da versão gratuita mais recente do LUMOplay. A aplicação tem duas funcionalidades principais, visíveis nos separadores à esquerda: as *apps* e o *motion maker*.

As *apps* são configurações interativas pré-criadas e disponíveis para *download*. Quando se descarrega uma das *apps*, é possível abri-las e o *software* projeta essa *app* para o ecrã/projetor definido. Para além disso, os objetos no ecrã (por exemplo, bolas ou balões) reagem ao gesticular, também dependendo do dispositivo configurado para tal.

A funcionalidade *motion maker*, é um editor de cenas e permite que se edite algumas das *apps* acima descritas. Ao editar uma das *apps*, é possível fazer edições como alterar o fundo, os objetos e os sons. Para além disso, este adapta-se ao estilo da cena criada, de forma que as edições sejam específicas para a cena carregada.

2.6 Tecnologias

Tendo em conta o grande foco no *software*, escolher a tecnologia de desenvolvimento é crucial para o bom funcionamento desta solução. Assim, é necessário levantar as principais características necessárias para a solução e filtrar à partida as tecnologias que não cumprem estes requisitos mínimos.

Em primeiro lugar, a compatibilidade: a solução tem de ser multiplataforma, mas apenas para sistemas operativos de *desktop* (Windows, macOS, Linux). Não sendo expectável que uma solução desta natureza funcione a partir de um *smartphone* ou *tablet*, se a tecnologia fornecer esse suporte também é um ponto positivo.

Em segundo lugar, o desempenho: apesar de esta característica ser mais ambígua, é fundamental que a solução tenha um desempenho satisfatório. No limite, tem de suportar um número considerável de elementos animados no ecrã sem se tornar consideravelmente lenta. Também é conveniente que o próprio funcionamento, isto é, a realização de ações como a abertura e navegação na solução não seja lenta nem tenha demasiada latência.

Em terceiro lugar, o preço: não deve usar uma tecnologia com um custo demasiado elevado ou que requeira uma subscrição. Ainda que não seja expectável que usar uma tecnologia comercialmente seja totalmente livre de custos, também não deve ultrapassar os custos de produção, ou seja, o custo da tecnologia não deve inflacionar demasiado o preço final para um potencial cliente.

Assim, tendo em conta estas características básicas, é possível concluir que a melhor solução é uma *framework web* ou híbrida. Ainda que tecnologias *web* não costumem ter melhor desempenho do que tecnologias nativas, é uma boa forma de garantir a compatibilidade entre plataformas.

Escolheu-se assim, as seguintes opções:

- **Electron:** *framework* de desenvolvimento de aplicações *desktop*, utilizando linguagens *web* (JavaScript ou TypeScript) e compilando para código *web*;
- **Flutter:** *framework* de desenvolvimento de aplicações *mobile* e *desktop*, utilizando uma linguagem própria (Dart) e compilando para código nativo.

2.6.1 Electron

O Electron é uma *framework* de desenvolvimento de software mantida pelo GitHub. Permite o desenvolvimento de aplicações para sistemas operativos de computador, usando tecnologias *web*. Atualmente, é utilizada numa série de aplicações de grandes empresas, como é o caso da Microsoft (atual proprietária do GitHub) (OpenJS Foundation, 2022).

Curiosamente, é muito comum ver críticas e oposição quando uma determinada empresa cria ou transita o seu software para Electron (Podfeet, 2021). O principal motivo de crítica é que a sua utilização de disco e memória é bastante superior às outras tecnologias. Isto acontece porque o Electron cria um *wrapper* (invólucro) do Chromium (plataforma que navegadores como o Google Chrome e o Microsoft Edge utilizam) e do Node.js, essencialmente contendo instalações destas duas tecnologias em cada aplicação. Outra crítica muito comum tem a ver com o desempenho das aplicações, que por vezes pode ser substancialmente pior quando comparado a uma tecnologia nativa.

Na opinião do autor, enquanto válidas, algumas destas críticas são infundamentadas, uma vez a grande maioria do software que utiliza Electron tem um bom desempenho, tornando-se às vezes até difícil distinguir se são nativas ou não. Mas como em todos os casos, existem algumas exceções que tendem a manchar a reputação do Electron.

Uma outra grande vantagem de utilizar esta *framework*, e uma vez que utiliza tecnologias *web*, é a possibilidade de utilizar uma segunda *framework* para auxiliar no desenvolvimento. Para o presente estudo, foram analisadas e estudadas as tecnologias Angular e React.

2.6.1.1 Angular

O Angular é uma *framework web* baseada em TypeScript. É desenvolvida por uma equipa da Google com a colaboração de uma grande comunidade de pessoas e empresas (Google, 2022).

O Angular utiliza uma arquitetura bastante rígida e "opinativa", ou seja, não permite tanta liberdade e flexibilidade na organização do código. Isto pode, à partida, parecer algo negativo, mas é um dos principais motivos pelo qual é tão apreciado pelas empresas. Existindo um padrão pré-estabelecido é mais fácil gerir e manter uma solução que utilize esta tecnologia. Por outro lado, os utilizadores tendem a achar que a arquitetura e funcionalidades são mais complexas que as de uma *framework* como o React.

2.6.1.2 React

O React é uma biblioteca JavaScript utilizada para desenvolver interfaces de utilizador. É desenvolvida pela Meta (ex-Facebook) e, tal como o Angular, por uma comunidade de programadores e empresas (Meta Open Source, 2022).

O React utiliza uma arquitetura muito flexível, quase inexistente. A grande distinção entre o React e JavaScript nativo está na sua sintaxe e nos seus componentes. A sintaxe JSX (JavaScript XML) assemelha-se bastante ao HTML (*HyperText Markup Language*), porém permite tirar partido de todo o potencial do JavaScript, ou seja, criar elementos mantendo toda a lógica coesa e sem separações artificiais entre o código e o HTML. Já os componentes assemelham-se a funções JavaScript e tem como grande vantagem a criação de elementos de UI (*User Interface* ou Interface de Utilizador) independentes e reutilizáveis. Para além destas duas diferenças, o React introduz também os conceitos de estado e ciclo de vida, o que facilitam imensamente a gestão de dados e a atualização da interface em tempo real.

2.6.2 Flutter

O Flutter é um SDK (*Software Development Kit*) para aplicações *front-end* lançado em 2017 pela Google. Tal como todas as outras tecnologias apresentadas, o Flutter permite o desenvolvimento multiplataforma de aplicações a partir de uma única base de código para Android, iOS, Windows, macOS, Linux e *web* (Google, 2022).

Ao contrário das outras tecnologias, a abordagem do Flutter ao desenvolvimento é bastante diferente. Se por um lado, o Electron utiliza tecnologias *web* para o desenvolvimento e executa esse código *web* através do Chromium, o Flutter compila as aplicações para código nativo de cada sistema operativo. Para além disso, o Flutter utiliza uma linguagem própria especificamente criada para o efeito, o Dart.

2.6.3 Comparação

Nesta secção faz-se uma comparação entre todas as tecnologias apresentadas, por forma a poder melhor distingui-las, mas também para facilitar e justificar o preenchimento da análise de valor, em 3.2.4 (Cianci, 2021).

2.6.3.1 Funcionalidades

Tendo em conta as diferentes características dos dois tipos de tecnologias, aliado com o número de anos no mercado que ambas têm, existem algumas diferenças substanciais no que cada uma das tecnologias consegue atingir.

Começando pelas tecnologias *web*, mas tendo sempre em foco o Electron, estas possuem um historial de longos anos de *web*, o que permite atingir quase tudo o que se possa imaginar. O Node.js, em conjunto com o seu repositório NPM (*Node Package Manager*), disponibiliza uma grande lista de pacotes que expandem significativamente as capacidades que as tecnologias *web* não conseguem atingir de outra forma.

Quanto às tecnologias híbridas, mais uma vez focando no Flutter, já funcionam de maneira ligeiramente diferente. Ainda que o Flutter também possua o seu repositório de pacotes que permitem expandir as suas capacidades, tal como o NPM, o facto de existir há significativamente menos tempo em conjunto com o facto de necessitar de compilar esses pacotes para cada tecnologia suportada, trazem uma pequena desvantagem. É muito normal encontrar no Flutter pacotes que suportam apenas uma parte das plataformas, ao passo que no Node.js, como é executado apenas em *web*, a maioria (senão mesmo todos) dos pacotes funcionam em qualquer que seja a plataforma a executar.

2.6.3.2 Desempenho

Tendo em conta, mais uma vez, as diferentes características destas tecnologias, o desempenho atingido por cada uma também varia significativamente.

Começando pelo Electron, este é caracteristicamente conhecido por consumir consideravelmente mais recursos do que as restantes tecnologias, quer nativas, quer híbridas. Como já foi explicado em 2.6.1, uma vez que o Electron cria um pacote com o Chromium e o Node.js em cada aplicação gerada, até um simples "*Hello, World!*" ocupa por vezes mais de 100 MB em disco, para além do consumo de memória e processos (H., 2021).

Já o Flutter é bastante mais simpático na quantidade de recursos que utiliza. Mais uma vez, isto deve-se à sua natureza híbrida, já que todo o código é compilado nativamente em cada aplicação gerada. Ainda que isto não permita atingir os mesmos resultados de uma aplicação nativa, melhora significativamente o desempenho e a pegada que deixa no computador, quando comprado ao Electron.

Chegou-se, assim, às seguintes conclusões (H., 2021):

- Com o crescimento da aplicação, o Electron começa a ter dificuldades em abrir e desenhar conteúdo no ecrã rapidamente;
- O Electron é uma solução provada e testada. Tendo uma boa aplicação *web*, o Electron vai funcionar sem problemas;
- O Electron ocupa demasiado espaço em disco e utiliza demasiada memória;
- É preciso ter algum cuidado com a quantidade de módulos a importar e a forma como esses módulos são carregados e, posteriormente, distribuídos pelo Electron;
- O Flutter para sistemas operativos *desktop* atingiu a versão estável muito recentemente e, por isso, não é uma solução provada em grande escala;
- O Flutter ocupa menos espaço em disco e utiliza menos memória do que o Electron.

Apesar de o Electron ter pior desempenho que o Flutter, é importante ressalvar que existem alternativas como o WebAssembly (Wasm) que melhoram substancialmente o desempenho e a viabilidade de uma aplicação interativa. Um excelente exemplo de boa utilização do Wasm é o Figma, que ainda que sendo uma aplicação *web*, tem um desempenho ao nível de uma aplicação nativa (Wallace, 2017).

2.6.3.3 Popularidade

Apesar de estas tecnologias *web* serem relativamente recentes, também já contam com alguns anos de mercado em que é possível observar as tendências de utilização e sua popularidade e preferência por parte dos programadores.

Observando a Figura 11, que mostra as tendências de pesquisa das tecnologias em estudo no Google nos últimos cinco anos (desde o início de 2017 ao final de 2021), é possível observar que o Angular (a vermelho) e o React (a amarelo) partilham de uma forma destacada o pódio. É interessante verificar que o Angular começou a perder a liderança até que em 2018 foi ultrapassado pelo React. Por outro lado, o Flutter (a verde) tem vindo a ganhar

consistentemente popularidade, muito também por ser a tecnologia mais recente aqui em estudo. Já o Electron (a azul) tem uma baixa e estagnada popularidade segundo esta fonte.

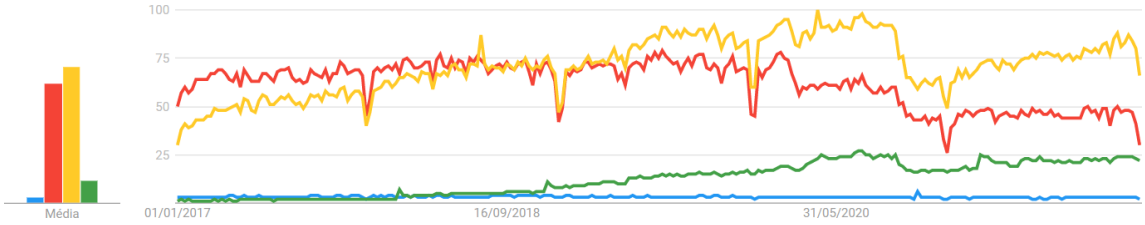


Figura 11 Popularidade das tecnologias em estudo nas pesquisas do Google[9]

Na Figura 12, que mostra o número de instalações anuais no repositório NPM, é possível verificar uma tendência semelhante à da Figura 11, ainda que de forma mais abrupta. O React (a azul) tem descolado de forma muito consistente e é de longe a tecnologia com mais instalações anuais. O Angular (a verde) mostra uma tendência de pouca evolução, ainda que também de ligeira subida. O Electron (a vermelho) tem um número de instalações bastante mais baixo ao das outras tecnologias, ainda que com uma ligeira tendência de crescimento. O Flutter não está aqui representado, uma vez que não sendo uma tecnologia *web*, tem o seu próprio repositório e, portanto, não usa o NPM.

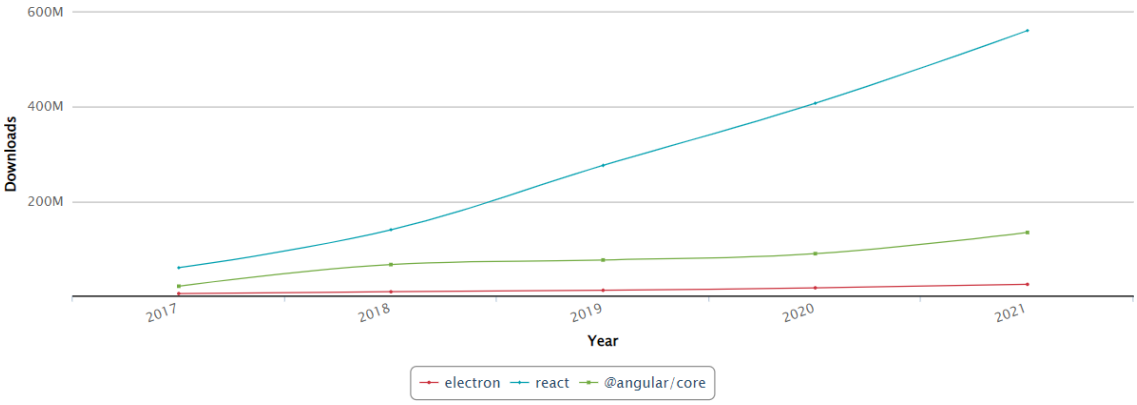


Figura 12 Popularidade das tecnologias em estudo nas instações do NPM[10]

[9] Gráfico retirado de https://trends.google.pt/trends/explore?date=2017-01-01%202021-12-31&q=%2Fg%2F11bw_559wr,%2Fg%2F11c6w0ddw9,%2Fm%2F012l1vxv,%2Fg%2F11f03_rzbg

[10] Gráfico retirado de https://npm-stat.com/charts.html?package=electron&package=%40angular%2Fcore&package=react&from=2017-01-01&to=2021-12-31

Por último, na Figura 13, que mostra a popularidade das tecnologias em estudo segundo o questionário anual de 2022 feito aos programadores utilizadores do StackOverflow, é possível verificar mais uma vez uma tendência muito semelhante às Figura 11 e Figura 12.

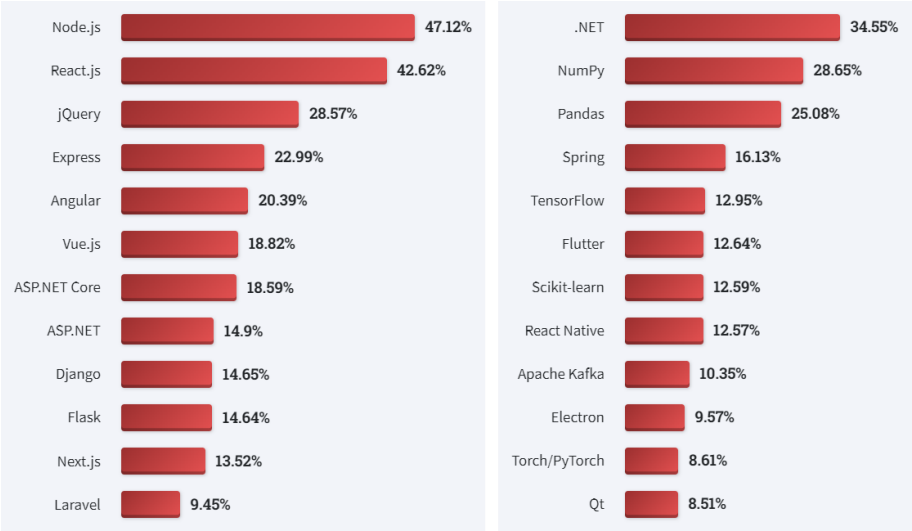


Figura 13 Popularidade das tecnologias em estudo no questionário do StackOverflow[11]

No gráfico da esquerda, o React é mais uma vez o líder da lista de tecnologias em estudo. O Angular está, ainda assim, posicionado num respeitável quinto lugar. É, no entanto, interessante reparar que o Node.js, posicionado em primeiro lugar, é a tecnologia recomendada para instalar e utilizar tanto o React, como o Angular.

No gráfico da direita, é possível ver que o Flutter se encontra em sexto lugar, uma posição que tem vindo a crescer nos últimos anos. Também presente na décima posição está o Electron, ainda que com uma percentagem não muito diferente da do Flutter. De notar, que este gráfico está a comparar um leque de tecnologias mais abrangente, daí que as posições mais acima mostram tecnologias que em quase nada se relacionam com o Flutter ou com o Electron.

[11] Gráficos retirados de https://survey.stackoverflow.co/2022/#section-most-popular-technologies-web-frameworks-and-technologies e de https://survey.stackoverflow.co/2022/#section-most-popular-technologies-other-frameworks-and-libraries

3 Análise de valor

Neste capítulo é apresentada a análise de valor para a solução proposta. O objetivo principal desta análise é o de avaliar como maximizar o valor da solução a um custo mínimo sem que esta perca qualidade.

3.1 Processo de inovação

Todo o processo de inovação pode ser dividido em três partes: o *Front End of Innovation* (FEI), o processo de desenvolvimento do novo produto e a comercialização, tal como é possível ver na Figura 14.

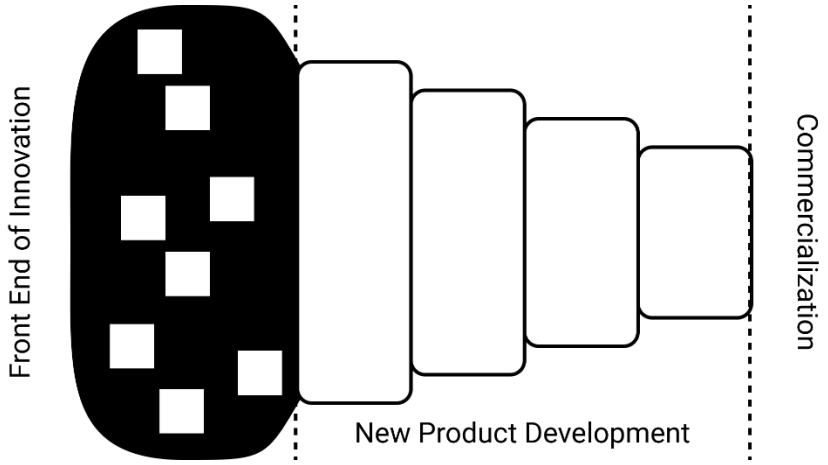


Figura 14 Front End of Innovation

Este método surgiu pelas mãos de Peter Koen em 2002, como uma evolução do *Fuzzy Front End*. Segundo o autor, os motivos que o levaram a adotar o termo *Front End of Innovation*, prendem-

se precisamente pelo facto de o termo *fuzzy* (difuso ou vago) implicar que o *front end* é misterioso, carece de responsabilidade e não pode ser avaliado criticamente (Koen, et al.) (Koen P. , 2014).

3.2 New Concept Development

O *New Concept Development* (NCD) fornece uma linguagem comum e uma visão holística do *front end*. O modelo divide o *front end* em três áreas distintas. A primeira é o motor, ou centro do modelo, que responde pela visão, estratégia e cultura que impulsiona o FEI. A segunda, ou parte interna, define os cinco elementos de atividade do *front end*. Já a terceira área consiste nos fatores ambientais externos (Koen P. , 2014).

A Figura 15, baseada no modelo de Peter Koen (Koen, et al.), demonstra o *New Concept Development*:

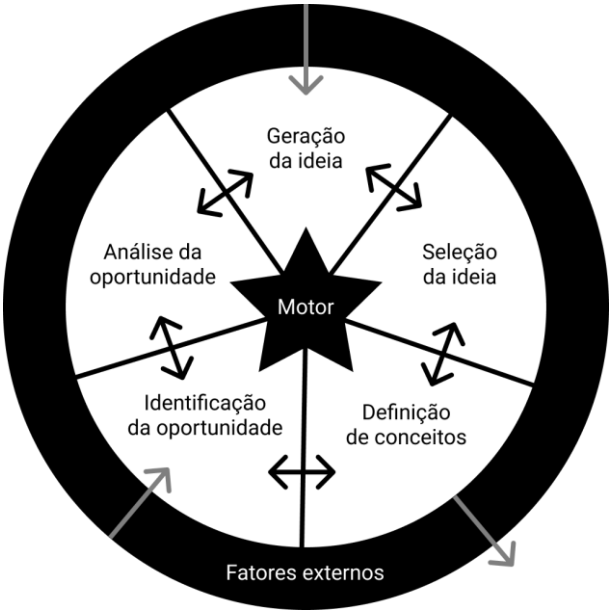


Figura 15 New Concept Development

Para o presente estudo serão identificadas e explicitadas as cinco atividades do interior do modelo: identificação da oportunidade, análise da oportunidade, geração da ideia, seleção da ideia e definição de conceitos.

3.2.1 Identificação da oportunidade

É possível identificar uma oportunidade através de uma série de origens: uma lacuna ou falha no mercado, uma necessidade por parte do mercado/cliente ou uma tendência de mercado.

No caso concreto do presente estudo, a oportunidade identificada surgiu pelas seguintes lacunas/necessidades:

- A primeira, e a mais importante, é o facto de a interação por comandos de voz não estar suficientemente explorada em paredes interativas;
- Depois, pela necessidade de existir *software* que permita criar projeções interativas que seja menos dependente do seu *hardware* de projeção;
- E por último, uma necessidade por parte do cliente de desenvolver e possuir um software de projeção com as características em estudo.

3.2.2 Análise da oportunidade

A segunda atividade consiste em analisar a oportunidade. Esta análise é realizada com o intuito de confirmar que a oportunidade identificada é, de facto, viável. A ferramenta mais comum para o efeito é o modelo SWOT.

O modelo SWOT permite analisar as oportunidades, listando os seus pontos positivos e negativos em duas dimensões distintas: interna e externa. A combinação destas dimensões gera os seguintes parâmetros:

- ***Strengths* (Forças):** são as características internas da empresa/projeto que podem ser vistas como uma vantagem em relação a outras empresas/projetos;
- ***Weaknesses* (Fraquezas):** são as características internas da empresa/projeto que podem ser vistas como uma desvantagem em relação a outras empresas/projetos;
- ***Opportunities* (Oportunidades):** são os fatores externos que a empresa/projeto pode utilizar a seu favor;
- ***Threats* (Ameaças):** são os fatores externos que podem causar problemas à empresa/projeto.

Na Figura 16 são apresentados os quatro parâmetros acima mencionados para o presente estudo:

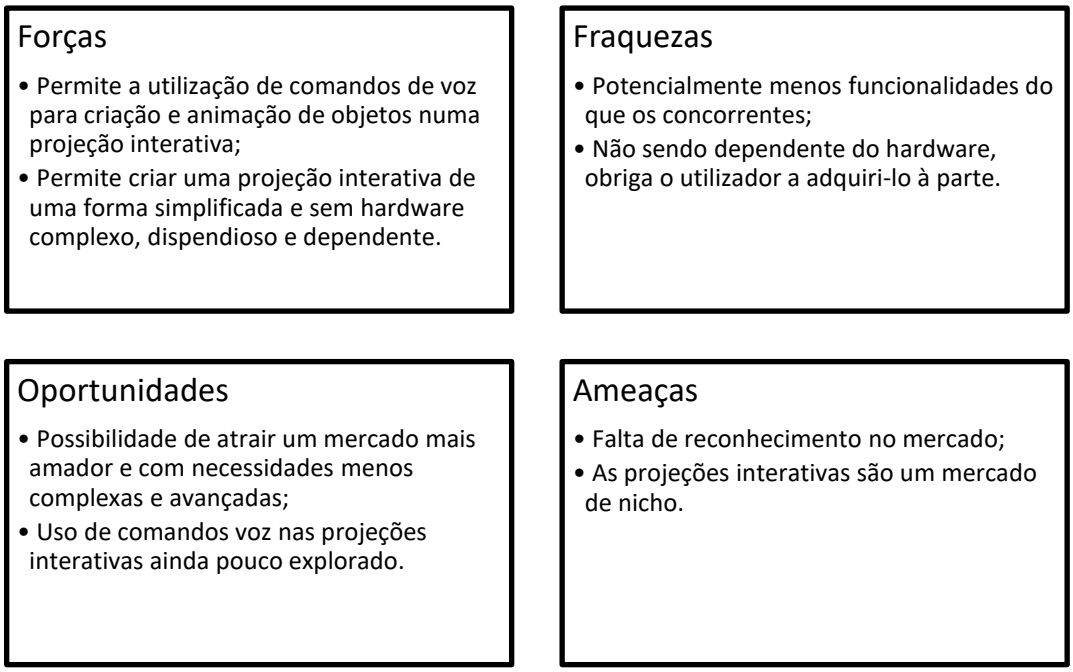


Figura 16 Modelo SWOT

3.2.2.1 Forças

As forças identificadas prendem-se precisamente com as lacunas identificadas em 3.2.1. O facto de a utilização de comandos de voz para criação e animação de objetos numa projeção interativa não estar suficientemente explorada pode ser usada como vantagem para a solução. Para além disso, a solução permite que se utilize equipamento das mais variadas naturezas, garantindo assim uma instalação descomplicada e acessível.

3.2.2.2 Fraquezas

Nas fraquezas foi identificado que a solução, pelo menos na sua primeira versão, poderá ter potencialmente menos funcionalidades que a concorrência. Em colisão com uma das forças está a obrigatoriedade de o utilizador adquirir e instalar os seus próprios equipamentos.

3.2.2.3 Oportunidades

Nas oportunidades foram identificadas a possibilidade de atrair um mercado mais amador e que, portanto, não requer nem dispõe de orçamento para soluções mais complexas e dispendiosas. Em segundo lugar, o facto de as projeções interativas controladas por voz estarem pouco exploradas.

3.2.2.4 Ameaças

Nas ameaças foram identificados os principais problemas de uma solução nova no mercado. Em primeiro lugar a falta de reconhecimento no mercado pode dificultar a sua exposição num primeiro momento. A juntar a isto, o facto de as projeções interativas já serem por si só um mercado de nicho.

3.2.3 Geração da ideia

Tendo em conta que a solução a implementar vai ser utilizada em sistemas operativos de computador, chegou-se às seguintes alternativas de implementação:

- **Utilizar tecnologia *web*:** utilização de tecnologias, como o Angular ou React, que utilize Electron para gerar a aplicação final;
- **Utilizar tecnologia híbrida:** utilização de tecnologias que não utilizem linguagens *web* e que compilem nativamente, como o Flutter;
- **Utilizar tecnologia nativa:** utilização de tecnologias nativas a cada sistema operativo suportado (Windows, macOS, Linux).

3.2.4 Seleção da ideia

Tendo por base as ideias identificadas anteriormente, é necessário fazer uma análise concreta por forma a selecionar a mais apropriada ao presente estudo. Para isso, utiliza-se o método AHP. O método AHP (Analytic Hierarchy Process) é uma técnica estruturada para organizar e analisar decisões complexas, com base na matemática e psicologia. Foi desenvolvido por Thomas Saaty, professor na Universidade de Pittsburgh, na década de 1970 (Saaty, 2008).

A primeira fase do método AHP consiste em definir o problema e estruturá-lo num diagrama hierárquico. Para isso são definidos três pontos: o objetivo, os critérios e as alternativas.

No contexto em questão, a utilização do método AHP tem por objetivo escolher o melhor tipo de tecnologia para a implementação da solução.

Tendo isso em consideração, os critérios definidos foram escolhidos tendo como maior foco o desenvolvimento e manutenção da solução, mas também a experiência final do utilizador. Assim, os critérios definidos foram os seguintes:

- **Complexidade:** este critério serve para medir a complexidade de implementação da solução, quer do ponto de vista do domínio da tecnologia, quer das dificuldades previstas durante o desenvolvimento. Considera-se que a melhor alternativa é a que tiver uma complexidade menor.
- **Desempenho:** este critério serve para avaliar o desempenho geral da solução, desde a utilização das funcionalidades previstas, até a abertura e navegação dentro da aplicação. Considera-se que a melhor alternativa é a que tiver um desempenho melhor.
- **Duração:** este critério serve para avaliar a duração necessária para se desenvolver a solução. Dada a natureza académica deste estudo, existem datas que têm de ser obrigatoriamente cumpridas. Considera-se que a melhor alternativa é a que tiver uma duração menor.
- **Manutenção:** este critério serve para avaliar os custos e dificuldade estimados de uma posterior manutenção da solução. Considera-se que a melhor alternativa é a que tiver um custo menor.

Já quanto às alternativas, estas foram identificadas e detalhadas em 3.2.3.

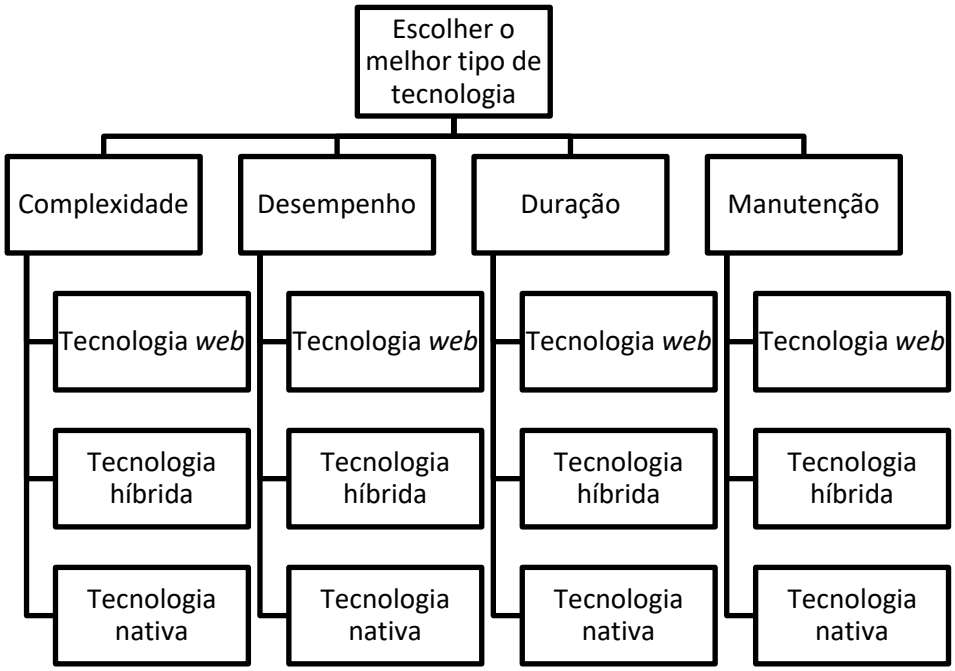


Figura 17 Árvore hierárquica AHP

3.2.4.1 Comparação entre os critérios

A segunda parte do método AHP consiste na definição de prioridades. Na Tabela 1 é apresentada a escala fundamental de Saaty para os níveis de importância de comparações (Saaty, 2008).

Tabela 1 Nível de importância de comparações

Índice de importância	Definição
1	Igual importância
3	Pouca importância
5	Importância elevada
7	Importância muito elevada
9	Importância absoluta
2, 4, 6, 8	Valores intermédios

Tabela 2 Prioridades relativas dos critérios

	Complexidade	Desempenho	Duração	Manutenção
Complexidade	1	1/3	1	3
Desempenho	3	1	3	5
Duração	1	1/3	1	3
Manutenção	1/3	1/5	1/3	1
Soma da coluna	5.33	1.87	5.33	12

Na Tabela 2 estão definidas as prioridades entre os critérios. Assim, é possível desde já concluir que:

- O critério complexidade tem:
    - pouco menos importância do que o critério desempenho;
    - a mesma importância do que o critério duração;
    - pouco mais importância do que o critério manutenção.
- O critério desempenho tem:
    - pouco mais importância do que o critério complexidade;
    - pouco mais importância do que o critério duração;
    - muito mais importância do que o critério manutenção.
- O critério duração tem:
    - a mesma importância do que o critério complexidade;
    - pouco menos importância do que o critério desempenho;
    - pouco mais importância do que o critério manutenção.
- O critério manutenção tem:
    - pouco menos importância do que o critério complexidade;
    - muito menos importância do que o critério desempenho;
    - pouco menos importância do que o critério duração.

A Tabela 3 foi construída dividindo cada elemento da Tabela 2 pela soma da respetiva coluna representada também na mesma tabela.

Tabela 3 Prioridades relativas dos critérios

	Complexidade	**Desempenho**	**Duração**	**Manutenção**	**Prioridade relativa**
Complexidade	0.19	0.18	0.19	0.25	0.20
Desempenho	0.56	0.53	0.56	0.42	**0.52**
Duração	0.19	0.18	0.19	0.25	0.20
Manutenção	0.06	0.11	0.06	0.08	0.08

Ainda na Tabela 3 foi calculada a prioridade relativa de cada critério, calculando a média de cada linha da tabela. Assim, é possível concluir desde já que o critério com maior importância é o desempenho. Em segundo, os critérios complexidade e duração, que têm a mesma importância. E por último, a manutenção.

Esta ordem pode ser justificada da seguinte forma:

- **Desempenho:** É de longe o critério mais importante numa aplicação interativa. Quando se utiliza uma aplicação interativa, como um jogo ou um editor imagem ou vídeo, o primeiro critério analisado é quase sempre o desempenho. Obviamente que os resultados vão variar consoante o equipamento utilizado, mas o tipo de tecnologia também vai influenciar bastante esta questão.
- **Complexidade:** Este é o segundo critério mais importante. Dada a natureza da implementação desta solução, quanto menor for a curva de aprendizagem e quanto menor forem as dificuldades previstas, melhor será o resultado da solução.
- **Duração:** Este critério tem igual importância ao critério complexidade, uma vez que refletem situações similares. A principal razão para a distinção destes critérios, prende-se com as tecnologias nativas. Criar duas aplicações pode não ser mais complexo, mas é substancialmente mais demorado.
- **Manutenção:** Este é o critério menos importante dos avaliados, uma vez que só se aplicará no pós-desenvolvimento. No entanto, quanto melhor implementada ficar a solução, menos custos terá na futura manutenção.

O próximo passo é avaliar a consistência das prioridades relativas definidas. Para isso, calcula-se a razão de consistência (RC) para medir se os julgamentos realizados foram consistentes em relação a amostras de juízo aleatórias. O método AHP é baseado no pressuposto de que o decisor é racional, isto é, se A é preferível a B e B é preferível a C, então A é preferível a C.

Considera-se $Ax = \lambda_{max}x$, onde x é o vetor próprio:

$$\begin{bmatrix} 1 & 1/3 & 1 & 3 \\ 3 & 1 & 3 & 5 \\ 1 & 1/3 & 1 & 3 \\ 1/3 & 1/5 & 1/3 & 1 \end{bmatrix} \times \begin{bmatrix} 0.20 \\ 0.52 \\ 0.20 \\ 0.08 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.20 \\ 0.52 \\ 0.20 \\ 0.08 \end{bmatrix}$$

$$\begin{bmatrix} 0.81 \\ 2.12 \\ 0.81 \\ 0.32 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.20 \\ 0.52 \\ 0.20 \\ 0.08 \end{bmatrix}$$

$$\lambda_{max} = \frac{\frac{0.81}{0.20} + \frac{2.12}{0.52} + \frac{0.81}{0.20} + \frac{0.32}{0.08}}{4} = \frac{4.05 + 4.08 + 4.05 + 4}{4} \cong 4.05$$

Uma vez calculado λ_{max}, é necessário calcular o índice de consistência (IC) para, em seguida, calcular a razão de consistência (RC).

$$IC = \frac{\lambda_{max} - n}{n - 1} = \frac{4.05 - 4}{4 - 1} \cong 0.02$$

O índice aleatório (IR) refere-se a um grande número de comparações efetuadas par a par. Este índice foi calculado para matrizes quadradas de ordem n pelo Laboratório de Oak Rigde, nos Estados Unidos. A Tabela 4 define os valores de IR em função do número de critérios.

Tabela 4 Tabela de IR para matrizes quadradas de ordem n

1	2	3	**4**	5	6	7	8	9	10
0.00	0.00	0.58	**0.90**	1.12	1.24	1.32	1.41	1.45	1.49

$$RC = \frac{IC}{IR} = \frac{0.02}{0.90} \cong 0.02$$

Como 0.02 < 0.1, os valores das prioridades relativas estão consistentes.

O próximo passo é construir a matriz de comparação paritária para cada um dos critérios. Para isso, são calculadas as prioridades relativas das alternativas para cada critério. O processo é igual ao realizado para as prioridades relativas dos critérios, razão pela qual os cálculos intermédios serão omitidos.

Assim, e seguindo a ordem alfabética já definida anteriormente, começa-se pelo critério complexidade, tal como mostra a Tabela 5.

Tabela 5 Prioridades relativas das alternativas para o critério complexidade

Complexidade	**Tecnologia *web***	**Tecnologia híbrida**	**Tecnologia nativa**	**Prioridade relativa**
Tecnologia *web*	1	1/3	7	0.33
Tecnologia híbrida	3	1	5	**0.58**
Tecnologia nativa	1/7	1/5	1	0.08

No que toca ao critério complexidade, verificou-se que:

- **Tecnologia *web*:** Toda a natureza *web* e multiplataforma deste tipo de tecnologia favorece e reduz a complexidade. É expectável que, no entanto, para atingir o objetivo desejado de desempenho, exista um aumento de complexidade do desenvolvimento.
- **Tecnologia híbrida:** Ao contrário das tecnologias *web*, as tecnologias híbridas compilam para código nativo, não sendo expectável tantas complexidades durante o desenvolvimento quando comparado com os outros dois tipos.
- **Tecnologia nativa:** a tecnologia nativa perde muitos pontos neste critério, uma vez que requer a aprendizagem e domínio de mais do que uma tecnologia (dependendo do número de sistemas operativos que se pretende suportar).

O próximo critério a ser analisado é critério desempenho, tal como mostra a Tabela 6.

Tabela 6 Prioridades relativas das alternativas para o critério desempenho

Desempenho	**Tecnologia *web***	**Tecnologia híbrida**	**Tecnologia nativa**	**Prioridade relativa**
Tecnologia *web*	1	1/3	1/5	0.11
Tecnologia híbrida	3	1	1/3	0.26
Tecnologia nativa	5	3	1	**0.63**

No que toca ao critério desempenho, verificou-se que:

- **Tecnologia *web*:** É já sabido que, dada a sua natureza, as tecnologias *web* oferecem sempre um pior desempenho do que os outros tipos. Será necessária alguma cautela adicional para mitigar esta questão, sem aumentar consideravelmente a complexidade.
- **Tecnologia híbrida:** Por outro lado, e mais uma vez dada a sua natureza, as tecnologias híbridas tem um desempenho bastante superior às tecnologias *web*. Isto deve-se ao facto de estas compilarem o seu código híbrido para código nativo, atingindo um desempenho semelhante ao de uma aplicação nativa.

- **Tecnologia nativa:** Este é o critério em que a tecnologia nativa se destaca em relação a qualquer outro tipo de tecnologia. Dado que o código é escrito e compilado de forma nativa, é atingido o desempenho máximo para a plataforma em questão.

O próximo critério a ser analisado é critério duração, tal como mostra a Tabela 7.

Tabela 7 Prioridades relativas das alternativas para o critério duração

Duração	**Tecnologia *web***	**Tecnologia híbrida**	**Tecnologia nativa**	**Prioridade relativa**
Tecnologia *web*	1	1	7	**0.47**
Tecnologia híbrida	1	1	7	**0.47**
Tecnologia nativa	1/7	1/7	1	0.07

No que toca ao critério duração, verificou-se que:

- **Tecnologia *web*:** A duração de desenvolvimento é muito semelhante à da tecnologia híbrida, uma vez que é necessário apenas criar uma base de código e as alterações requeridas para cada plataforma em específico são muito pequenas.
- **Tecnologia híbrida:** Tal como nas tecnologias *web*, as tecnologias híbridas compilam apenas com uma base de código, reduzindo assim a duração do seu desenvolvimento.
- **Tecnologia nativa:** A duração de desenvolvimento sofre muito dependendo do número de plataformas a suportar. Dado que um dos requisitos da solução é ser multiplataforma, pelo menos duas plataformas terão de ser suportadas, duplicando (no mínimo) a duração do desenvolvimento.

O último critério a ser analisado é critério manutenção, tal como mostra a Tabela 8.

Tabela 8 Prioridades relativas das alternativas para o critério manutenção

Manutenção	**Tecnologia *web***	**Tecnologia híbrida**	**Tecnologia nativa**	**Prioridade relativa**
Tecnologia *web*	1	1/3	7	0.33

Tecnologia híbrida	3	1	5	**0.58**
Tecnologia nativa	1/7	1/5	1	0.08

Já no que toca ao critério manutenção, verificou-se que:

- **Tecnologia *web*:** Existindo apenas uma base de código, a manutenção da solução fica simplificada. No entanto, a utilização excessiva de dependências para adicionar funcionalidades tende a gerar mais incompatibilidades, o que requer manutenções mais constantes;
- **Tecnologia híbrida:** Tal como nas tecnologias *web*, a existência de uma única base de código simplifica a manutenção. No entanto, a novidade deste tipo de tecnologias, especialmente nos sistemas operativos de computador, pode levar à necessidade de manutenções regulares para manter o código atualizado;
- **Tecnologia nativa:** As tecnologias nativas são mais estáveis no número de funcionalidades a mudar, mas, dado que existem várias bases de código, requer pelo menos o dobro do trabalho para manter.

Assim, foi construída a Tabela 9 que mostra a matriz de comparação paritária para cada critério, considerando cada uma das alternativas escolhidas. Estes valores foram obtidos a partir das prioridades relativas de cada uma das quatro tabelas anteriores.

Tabela 9 Matriz de comparação paritária

	Complexidade	**Desempenho**	**Duração**	**Manutenção**
Tecnologia *web*	0.33	0.11	0.47	0.33
Tecnologia híbrida	0.58	0.26	0.47	0.58
Tecnologia nativa	0.08	0.63	0.07	0.08
Prioridades relativas	0.20	0.52	0.20	0.08

Por fim, e realizando a multiplicação da matriz paritária pelas prioridades relativas também representadas na Tabela 9, obtém-se as prioridades compostas das alternativas, representadas na Tabela 10.

Tabela 10 Prioridades compostas das alternativas

Prioridades compostas das alternativas
0.24
0.39
0.36

Pode-se concluir, portanto, que a tecnologia que oferece mais vantagens são as tecnologias híbridas. Dado que ganha muito pontos no desempenho, as tecnologias nativas aparecem em segundo lugar. No entanto, dada a natureza temporal limitada deste estudo, a verdadeira alternativa às tecnologias híbridas são as tecnologias *web*.

3.3 Valor da solução

Após identificar a solução a ser implementada, é importante compreender o valor que esta cria para os seus utilizadores e clientes finais. Para isso, será analisada a proposta de valor e o modelo de negócio.

3.3.1 Proposta de valor

A proposta de valor tem como objetivo demonstrar a correspondência entre as necessidades dos consumidores e os benefícios que resultam da utilização de um produto ou serviço (Osterwalder & Pigneur, 2010). Para isso será utilizado o modelo CANVAS, que analisa as dores que os utilizadores têm, mas ao mesmo tempo os geradores e aliviadores dessas dores. Assim, do lado do utilizador são analisados os seus problemas e do lado da proposta de valor é explicado como criar valor para o utilizador.

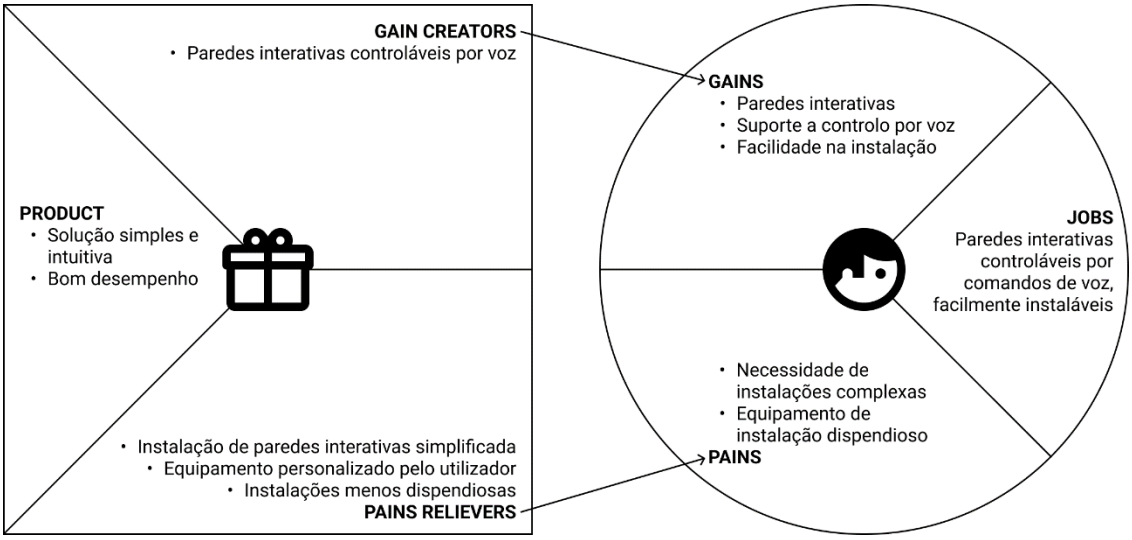


Figura 18 Modelo CANVAS na perspetiva da proposta de valor

Assim, e de acordo com a Figura 18, é possível retirar as seguintes conclusões:

- Os utilizadores procuram paredes interativas que possibilitem a interação por voz e que sejam facilmente instaláveis;
- Os utilizadores esperam que as paredes interativas não necessitem de instalações complexas, nem de equipamento dispendioso;
- A solução oferece uma instalação simplificada, em que o utilizador pode personalizar o equipamento e, portanto, ajustá-lo ao seu orçamento;
- Tem ainda o benefício adicionado de ser simples e intuitiva e oferecer um bom desempenho na sua utilização.

3.3.2 Modelo de negócio

O modelo de negócio representa a forma como a empresa está estruturada para gerar e capturar valor. Desta forma, quando se cria um negócio é fundamental pensar nele numa perspetiva de modelo de negócio e que meios se vão utilizar para gerar valor para os consumidores e potenciais acionistas, assim como capturar valor para a empresa.

Para isso, é usado novamente o modelo CANVAS, porém desta vez não na perspetiva de proposta de valor, mas sim de modelo de negócio. Esta é uma maneira sistematizada e prática de conceber um modelo de negócio.

<table>
<tr><td rowspan="2">Parceiros chave
ISEP
SIIS</td><td>Atividades-chave
Desenvolvimento de software;
Análise das necessidades do cliente.</td><td rowspan="2">Proposta de valor
Facilidade de implementação de uma parede interativa;
Controlo por comandos de voz;
Desempenho da solução.</td><td>Relação com clientes
Suporte na instalação e manutenção do software</td><td rowspan="2">Segmentos de consumidor
Pequenas empresas com instalações multimédia, como museus</td></tr>
<tr><td>Recursos-chave
Equipa de desenvolvimento.</td><td>Canais
Aplicação multiplataforma;
Redes sociais</td></tr>
<tr><td colspan="3">Estrutura de custos
Eventuais custos da tecnologia para comercialização;
Marketing e publicidade.</td><td colspan="2">Fontes de retorno
Eventual comercialização</td></tr>
</table>

Figura 19 Modelo CANVAS na perspetiva do modelo de negócio

A Figura 19 mostra o modelo CANVAS na perspetiva do modelo de negócio. Este está dividido em duas secções (delineadas pela linha a negrito): do lado direito está o *front stage*, que corresponde à parte do negócio que lida diretamente com o cliente, e do lado esquerdo está o *back stage*, que corresponde aos métodos necessários para concretizar o *front stage*.

3.3.3 Quality Function Deployment

O QFD (Quality Function Deployment) é um processo e um conjunto de ferramentas que são usadas de forma eficiente para definir os requisitos do cliente e para os converter em especificações de engenheira. O QFD foi desenvolvido originalmente no Japão no final da década de 1960 por Yoji Akao, enquanto trabalhava num estaleiro da Mitsubishi. Foi, entretanto, adotado por outras empresas, como a Toyota, e no início da década de 1980 levado para os Estados Unidos, também pelas empresas automotivas (Quality-One, 2022).

3.3.3.1 Necessidades do cliente

O primeiro passo consiste em definir as necessidades do cliente. Estas necessidades foram levantadas com base nos requisitos do cliente para a solução:

- **Controlo por comandos de voz:** o utilizador deve poder controlar a solução através de comandos de voz;
- **Gestão de objetos:** o utilizador deve poder adicionar objetos, como quadrados, círculos e triângulos, assim como eliminá-los;
- **Animação de objetos:** o utilizador deve poder animar os objetos adicionados, movendo, rodando ou alterando o seu tamanho;
- **Personalização de objetos:** o utilizador deve poder personalizar os objetos, mudando a sua cor, tamanho ou posição;
- **Gestão de projetos:** o utilizador deve poder criar projeto para que possam ser utilizados mais tarde.

A cada uma destas necessidades são atribuídos valores de importância absolutos, de 1 a 5, onde 1 representa a menor importância para o cliente e 5 a maior importância. Em seguida, os valores são normalizados. Estes resultados são apresentados na Tabela 11:

Tabela 11 Necessidades do cliente e respetivas importâncias absolutas e normalizadas

Necessidades	Importância absoluta	Importância normalizada
Controlo por comandos de voz	5	24%
Gestão de objetos	5	24%
Animação de objetos	4	19%
Personalização de objetos	4	19%
Gestão de projetos	3	14%

3.3.3.2 Características de engenharia

O segundo passo é semelhante ao anterior e consiste em definir as características de engenharia que respondem às necessidades levantadas pelo cliente. Assim, foram levantadas as seguintes características:

- **Tecnologia adequada:** esta característica avalia se a tecnologia é adequada aos requisitos previstos para a solução;
- **Interface de utilizador:** esta característica avalia se a interface de utilizador é amigável e cumpre a sua função;
- **Bom desempenho:** esta característica avalia se o desempenho da solução é satisfatório, mesmo em situações de maior pressão.

Por fim nas Figura 20 e Figura 21, são as apresentadas duas escalas que permitem relacionar as necessidades do cliente e as características de engenharia e correlacionar as diferentes características de engenharia, respetivamente.

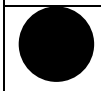
Matriz de relação		
	Forte	9
	Média	3
	Fraca	1
	Sem relação	0

Figura 20 Matriz de relação

Matriz de correlação	
++	Muito forte
+	Forte
−	Fraca
−−	Muito fraca
	Sem relação

Figura 21 Matriz de correlação

De acordo com a Figura 22, é possível retirar as seguintes conclusões:

- A interface de utilizador é a característica de engenharia que apresenta uma maior importância;
- A tecnologia adequada é a característica de engenharia que apresenta uma menor importância;
- A tecnologia adequada e o bom desempenho têm uma correlação muito forte;
- A tecnologia adequada e a interface de utilizador têm uma correlação forte.

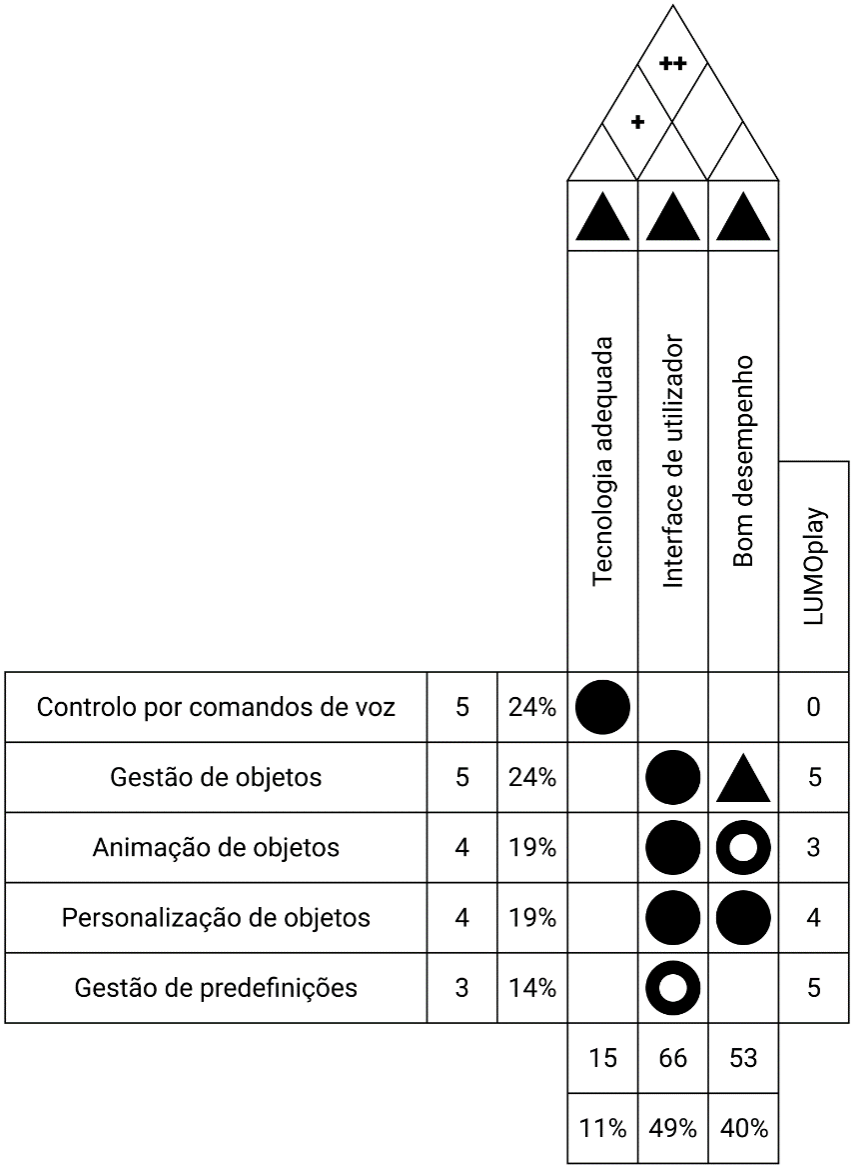


Figura 22 Quality Function Deployment

4 Design

Neste capítulo, é analisado em detalhe o design da solução. Em primeiro lugar, o conceito da solução, seguido da arquitetura através do modelo de domínio. Em seguida os requisitos funcionais que levam à análise dos casos de uso. Por fim, os requisitos não funcionais.

4.1 Conceito

Visto que o principal objetivo desta solução é permitir a interação dos utilizadores com o que os rodeia, foi concebida uma instalação onde será possível testar e executar a solução.

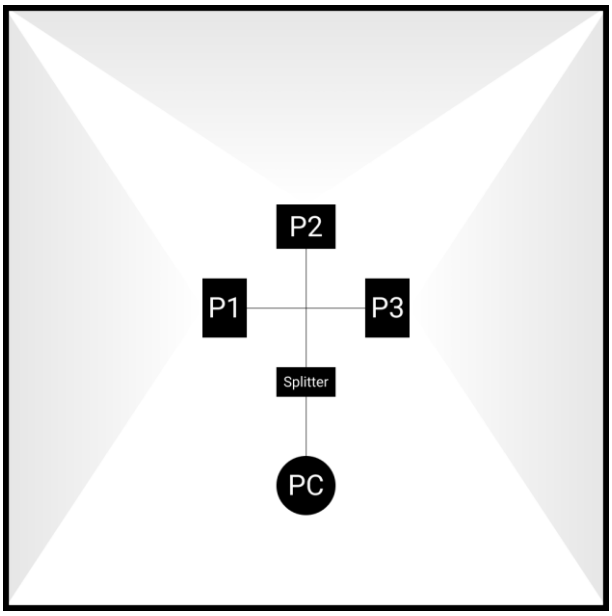


Figura 23 Planta da instalação para testar a solução

Na Figura 23 é possível ver a planta da instalação que será utilizada para testar a solução. Em primeiro lugar, o computador (representado por PC) onde está instalada a solução. Este liga-se então à porta de entrada do distribuidor (*splitter*), de forma a distribuir o sinal pelos projetores (representados por P1, P2 e P3). Por fim, cada projetor liga-se a uma das quatro portas de saída do distribuidor projetando assim o sinal para cada uma das paredes. De notar que esta instalação é apenas uma das várias possibilidades de distribuição e apresentação dos projetores.

4.1.1 Equipamentos

Quanto ao equipamento a utilizar, o SIIS disponibilizou quatro projetores do modelo Qumi Q3 Plus. Estes projetores serão ligados por um distribuidor, que irá distribuir o sinal do computador, tal como mostra a Figura 24.

Figura 24 Projetores e distribuidor utilizados para testar a solução

4.2 Arquitetura

Nesta secção, será analisada a arquitetura da solução através do modelo de domínio onde se apresentam os principais conceitos de negócio. São ainda analisadas alternativas de arquitetura para os vários tipos de tecnologias estudados em 3.2.

4.2.1 Modelo de domínio

O modelo de domínio permite observar como os vários elementos de negócio interagem internamente.

Na Figura 25 é possível observar o modelo de domínio da solução em estudo. O principal objeto é o *canvas*, onde todos os objetos estão inseridos. Estes objetos podem ser de várias naturezas: retângulos, círculos, polígonos, imagens ou textos. Cada um destes objetos pode conter várias

edições, como filtros. Os objetos podem também conter várias animações, como mover, rodar, redimensionar, entre outras. Para guardar o progresso desenvolvido no *canvas*, é possível criar projetos que permitem reutilizar o trabalho anteriormente desenvolvido.

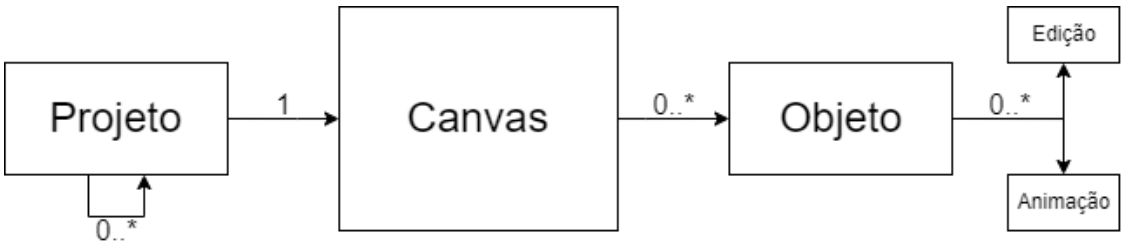


Figura 25 Modelo de domínio da solução

4.2.2 Alternativas de arquitetura

Nesta secção, são apresentadas as alternativas de arquitetura da solução, neste caso em concreto as diferenças que os tipos de tecnologia causam. É uma análise de alto nível, sendo por isso ocultadas muitas das especificidades das tecnologias e plataformas em concreto (Microsoft, 2022).

4.2.2.1 Tecnologia *web*

A primeira alternativa estudada é a utilização de uma tecnologia *web*. Apesar de também ser considerada híbrida, no sentido de que executa código comum que não é nativo à plataforma, esta difere significativamente da tecnologia híbrida estudada em 4.2.2.2.

Na Figura 26 é apresentada a arquitetura de uma aplicação *web*. À direita está representado o código da solução, chamado aqui de código *web*. Ao compilar a solução é gerada uma aplicação nativa que é executada na plataforma em questão e que tem acesso à maioria dos serviços nativos, como notificações, localização, câmara, microfone, entre outros. No entanto, dentro desta aplicação nativa está a ser executada uma aplicação *web*, ou seja, o código e as vistas continuam a ser totalmente *web* e não são adaptadas à plataforma.

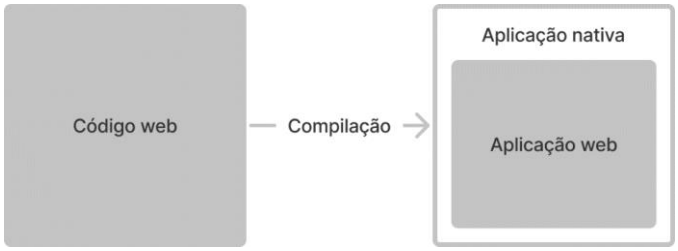


Figura 26 Compilação de uma aplicação *web*

4.2.2.2 Tecnologia híbrida

Em segundo lugar, estudou-se a utilização de uma tecnologia híbrida. Mais uma vez, apesar de ter semelhanças à tecnologia *web*, uma vez que também compila código multiplataforma, esta difere significativamente na maneira como o faz.

Na Figura 27 é apresentada a arquitetura de uma aplicação híbrida típica. À direita está apresentado o código híbrido da solução. Ao compilar a solução, o código anterior é então compilado em código nativo muito semelhante ao de uma aplicação nativa. Por fim, quando esta aplicação é executada, uma vez que o código anterior é nativo, a aplicação final também é totalmente nativa e executa apenas o mínimo de código não nativo à plataforma.

Figura 27 Compilação de uma aplicação híbrida

4.2.2.3 Tecnologia nativa

Por fim, a última alternativa estudada é o desenvolvimento de uma ou múltiplas aplicações nativas. Aqui o processo é muito mais direto, uma vez que o código é escrito nativamente e, portanto, executa diretamente na plataforma para a qual foi desenvolvido. De mencionar que, ao contrário das tecnologias anteriores, o código só vai compilar na plataforma para a qual foi escrito, sendo necessária uma rescrita completa da solução caso se pretenda utilizá-la numa plataforma diferente da original.

Na Figura 28 é mostrado precisamente o processo de compilação de uma aplicação nativa. O código é compilado e executado diretamente na plataforma em questão, sem nenhum processo adicional.

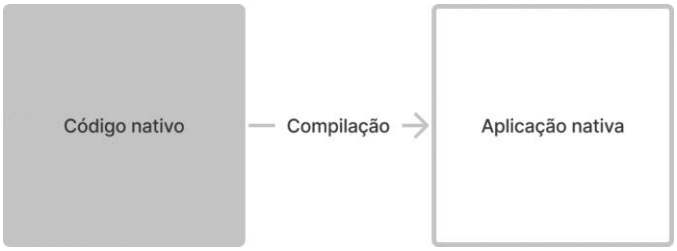


Figura 28 Compilação de uma aplicação nativa

4.3 Requisitos funcionais

Na presente secção são apresentados os requisitos funcionais levantados pelo SIIS. Por definição, um requisito funcional define uma funcionalidade de um determinado sistema. Isto é, representa o que a aplicação é, de facto, capaz de realizar.

Assim, os requisitos funcionais levantados são:

- Permitir a interação da solução através de comandos de voz;
- Permitir acrescentar objetos (círculos, triângulos, quadrados ou imagens);
- Permitir criar animações sobre os objetos adicionados.

4.4 Casos de uso

Nesta secção são analisados e detalhados os casos de uso escolhidos para implementação. Estes casos de uso têm por base os requisitos funcionais.

Para definir as funcionalidades da solução, foram analisados os principais casos de uso. Dada a prevalência do *canvas* e dos seus objetos, a maioria dos casos de uso envolvem ou estão ligados a estas áreas da solução.

Na Figura 29 são apresentados os casos de uso para a solução estudada. Ao todo, os oito casos de uso estão divididos em dois tipos: os relacionados diretamente com os objetos (à esquerda)

e os relacionados diretamente com os projetos (à direita). Todos os casos de uso são analisados em detalhe nesta secção.

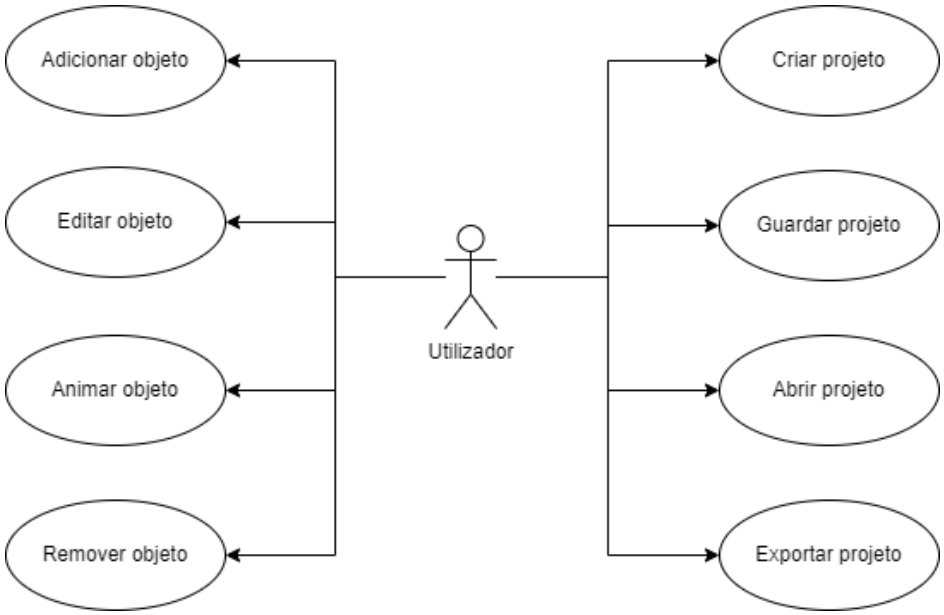


Figura 29 Diagrama de casos de uso

4.4.1 Adicionar objeto

O primeiro caso de uso é umas das primeiras ações que o utilizador realiza quando abre a solução: a criação de um objeto no *canvas*.

Na Tabela 12 é apresentado o caso de uso “Adicionar objeto” em detalhe. O ator deste caso de uso é o utilizador da solução. Não existem pré-condições para a realização deste caso de uso.

Tabela 12 Caso de uso "Adicionar objeto"

Ator	Utilizador
Descrição	O utilizador necessita de adicionar um objeto para começar o seu projeto
Pré-condições	-
Cenário principal	1. O utilizador escolhe o tipo de objeto pretendido e adiciona-o ao *canvas* 2. O *canvas* apresenta o objeto pretendido
Cenário alternativo	O utilizador adiciona o objeto através de comandos de voz

Pós-condições	O *canvas* deverá ter adicionado o objeto pretendido

A Figura 30 mostra o diagrama de sequência para o caso de uso em análise. O utilizador escolhe o tipo de objeto pretendido e adiciona-o. Após a adição, o *canvas* mostra esse objeto.

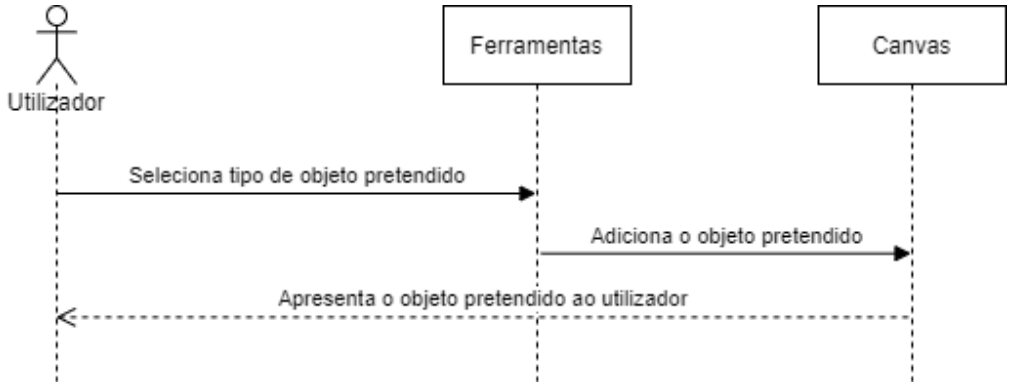


Figura 30 Diagrama de sequência do caso de uso "Adicionar objeto"

4.4.2 Editar objeto

O segundo caso de uso é a edição de um objeto. Neste caso, o utilizador poderá mover, rodar, aumentar ou diminuir, ou duplicar um objeto.

Na Tabela 13 é apresentado o caso de uso “Editar objeto” em detalhe. O ator deste caso de uso é o utilizador da solução. A única pré-condição existente é a necessidade de existir pelo menos um objeto no *canvas*.

Tabela 13 Caso de uso "Editar objeto"

Ator	Utilizador
Descrição	O utilizador pretende editar um objeto previamente adicionado
Pré-condições	É necessário existir pelo menos um objeto no *canvas*
Cenário principal	1. O utilizador seleciona o objeto a editar 2. São apresentadas as opções de edição 3. Realiza uma ou múltiplas edições ao objeto 4. As edições realizadas são apresentadas ao utilizador

Cenário alternativo	O utilizador edita o objeto através de comandos de voz
Pós-condições	O *canvas* deverá refletir as edições realizadas

A Figura 31 mostra o diagrama de sequência para o caso de uso em análise. Existem duas alternativas para selecionar o objeto a editar: selecionando o objeto diretamente no *canvas* ou utilizando a lista de objetos adicionados nas ferramentas. Após a seleção do objeto pretendido, são então apresentadas as opções de edição: no *canvas* é possível editar a posição, tamanho e rotação do objeto e nas ferramentas é possível duplicar, quadruplicar e adicionar filtros (este último apenas para as imagens). As edições são automaticamente refletidas no objeto.

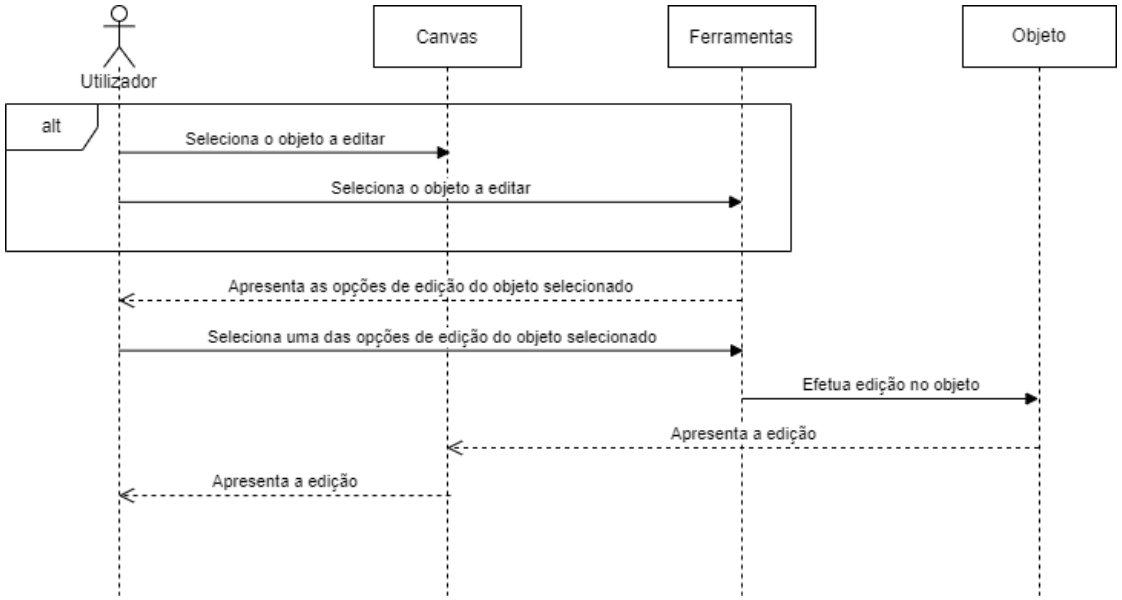


Figura 31 Diagrama de sequência do caso de uso "Editar objeto"

4.4.3 Animar objeto

O terceiro caso de uso é o de animação de um objeto. Neste caso, o utilizador escolhe uma animação, como mover, rodar e aumentar ou diminuir. A grande diferença para o caso de uso anterior está no facto de que neste caso o objeto é animado automaticamente, ou seja, é uma animação num ciclo infinito e não uma edição manual.

Na Tabela 14 é apresentado o caso de uso “Animar projeto” em detalhe. O ator deste caso de uso é o utilizador da solução. A única pré-condição existente é a necessidade de existir pelo menos um objeto no *canvas*.

Tabela 14 Caso de uso "Animar objeto"

Ator	Utilizador
Descrição	O utilizador pretende animar um objeto previamente adicionado
Pré-condições	É necessário existir pelo menos um objeto no *canvas*
Cenário principal	1. O utilizador seleciona o objeto a animar 2. São apresentadas as opções de animação 3. O utilizador seleciona uma ou várias animações 4. As animações são iniciadas e apresentadas ao utilizador
Cenário alternativo	O utilizador anima o objeto através de comandos de voz
Pós-condições	O *canvas* deverá refletir as animações iniciadas

A Figura 32 mostra o diagrama de sequência do caso de uso “Animar objeto”. Uma vez mais, existem duas alternativas para selecionar o objeto a animar: selecionando o objeto diretamente no *canvas* ou utilizando a lista de objetos adicionados nas ferramentas. Após a seleção do objeto pretendido, são então apresentadas as opções de animação: nas ferramentas é possível mover para a esquerda ou para a direita, rodar no sentido horário ou anti-horário e aumentar ou diminuir o objeto. As animações são automaticamente iniciadas e refletidas no objeto.

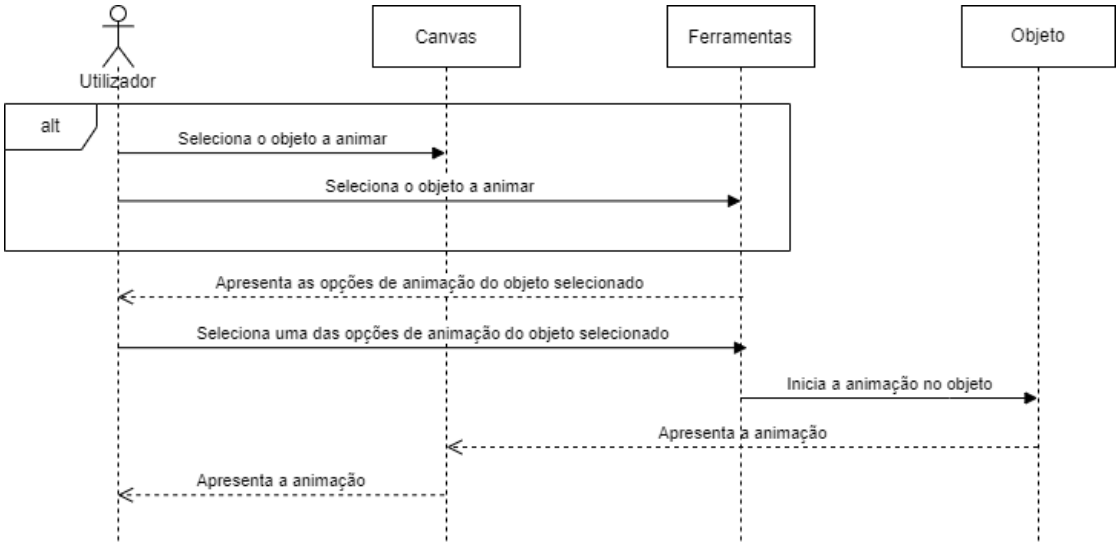


Figura 32 Diagrama de sequência do caso de uso "Animar objeto"

4.4.4 Remover objeto

O quarto caso de uso é a remoção de um objeto. Neste caso, o utilizador vai selecionar o objeto pretendido e removê-lo do *canvas*.

Na Tabela 15, é apresentado o caso de uso “Remover objeto” em detalhe. O ator deste caso de uso é o utilizador da solução. A única pré-condição existente é a necessidade de existir pelo menos um objeto no *canvas*.

Tabela 15 Caso de uso "Remover objeto"

Ator	Utilizador
Descrição	O utilizador pretende remover um objeto previamente adicionado
Pré-condições	É necessário existir pelo menos um objeto no *canvas*
Cenário principal	1. O utilizador seleciona o objeto a remover 2. É apresentada a opção de remoção 3. Seleciona a opção de remoção 4. O objeto é removido e a ação é refletida no *canvas*
Cenário alternativo	O utilizador remove o objeto através de comandos de voz
Pós-condições	O *canvas* deverá ter removido o objeto pretendido

A Figura 33 mostra o diagrama de sequência do caso de uso “Remover objeto”. Tal como nos casos de uso de edição e animação, existem duas alternativas para selecionar o objeto a remover: selecionando o objeto diretamente no *canvas* ou utilizando a lista de objetos adicionados nas ferramentas. Após a seleção do objeto pretendido, são então apresentadas as opções de edição e animação nas ferramentas onde se encontra a opção de remoção do objeto. A remoção do objeto é automaticamente refletida no *canvas*.

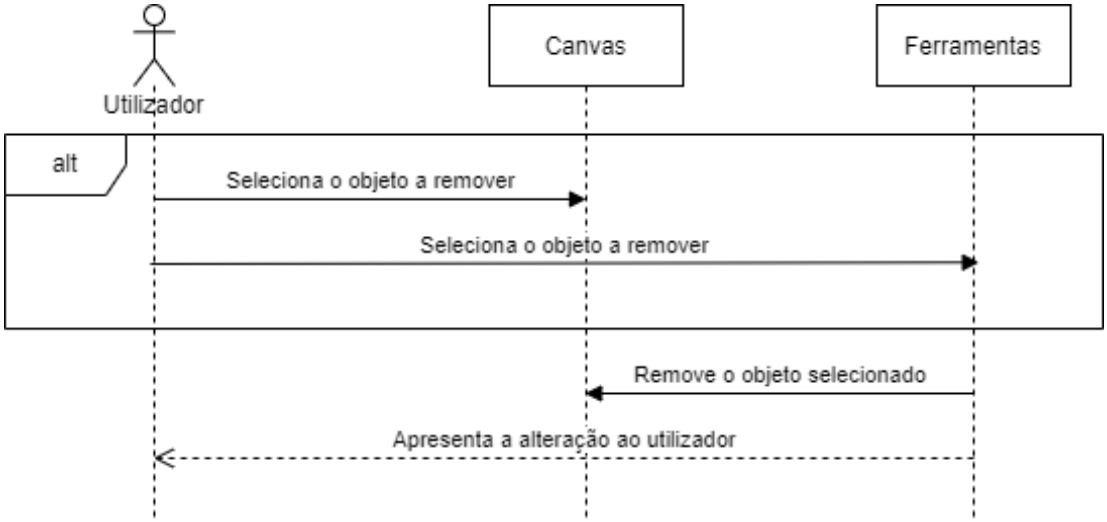


Figura 33 Diagrama de sequência do caso de uso "Remover objeto"

4.4.5 Criar projeto

O quinto caso de uso é o da criação de um projeto. Ao abrir a solução pela primeira vez um projeto é criado automaticamente, mas caso o utilizador pretenda recomeçar o seu trabalho, este caso de uso cobre essa situação.

Na Tabela 16 é apresentado o caso de uso “Criar projeto” em detalhe. O ator deste caso de uso é o utilizador da solução. Não existem pré-condições para a realização deste caso de uso.

Tabela 16 Caso de uso "Criar projeto"

Ator	Utilizador
Descrição	O utilizador pretende criar um projeto
Pré-condições	-
Cenário principal	1. O utilizador abre o menu 2. Clica na opção de criar um projeto 3. O *canvas* é reiniciado e apresentado ao utilizador
Cenário alternativo	O utilizador abre a solução pela primeira vez
Pós-condições	O novo projeto deverá ser criado

A Figura 34 mostra o diagrama de sequência do caso de uso “Criar projeto”. Existem duas maneiras de criar um projeto: abrindo a solução pela primeira vez (ou recarregando) ou através do menu. Ao usar a alternativa do menu, o utilizador clica na opção de novo projeto e o *canvas* atual é eliminado.

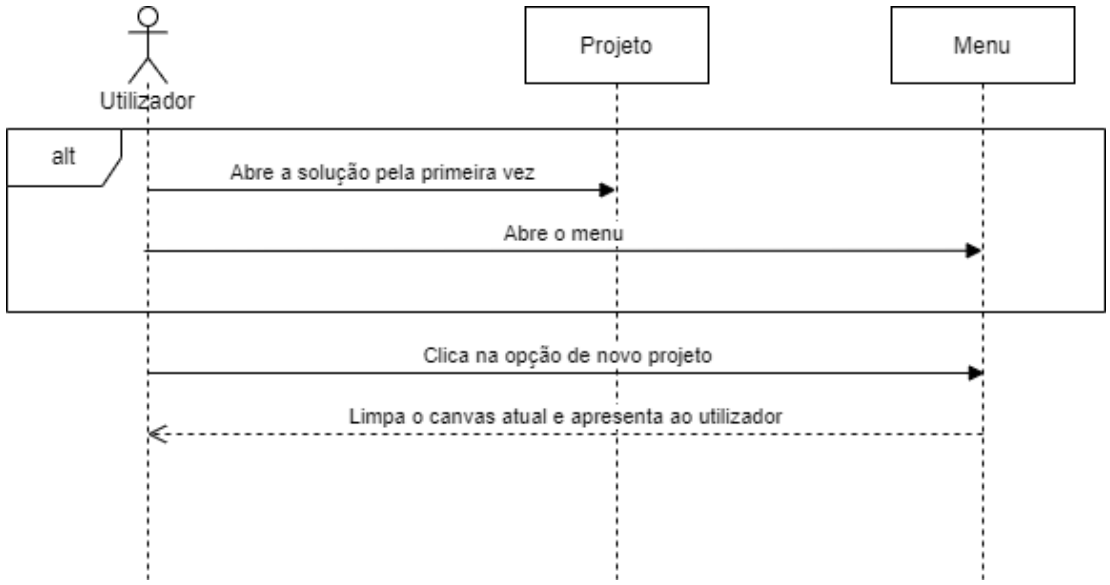


Figura 34 Diagrama de sequência do caso de uso “Criar projeto”

4.4.6 Guardar projeto

O sexto caso de uso é o de guardar um projeto. Neste caso, o utilizador poderá guardar o trabalho realizado no *canvas* para que possa reutilizar ou continuar mais tarde.

Na Tabela 17 é apresentado o caso de uso “Guardar projeto” em detalhe. O ator deste caso de uso é o utilizador da solução. A única pré-condição é a necessidade de se criar um projeto previamente.

Tabela 17 Caso de uso "Guardar projeto"

Ator	Utilizador
Descrição	O utilizador pretende guardar o projeto atual
Pré-condições	É necessário criar um projeto previamente
Cenário principal	1. O utilizador clica no botão de menu

	2. Clica na opção de guardar um projeto 3. Escolhe a localização do projeto a guardar 4. O sistema guarda o projeto e regressa ao *canvas*
Cenário alternativo	-
Pós-condições	O sistema deverá guardar o projeto atual

A Figura 35 mostra o diagrama de sequência do caso de uso “Guardar projeto”. Para guardar um projeto, o utilizador precisa de aceder ao menu e clicar na opção de guardar projeto. Uma janela de sistema é aberta para que o utilizador possa escolher a localização onde pretende guardar o projeto. Após confirmar a localização, o projeto é guardado e o utilizador regressa ao *canvas*.

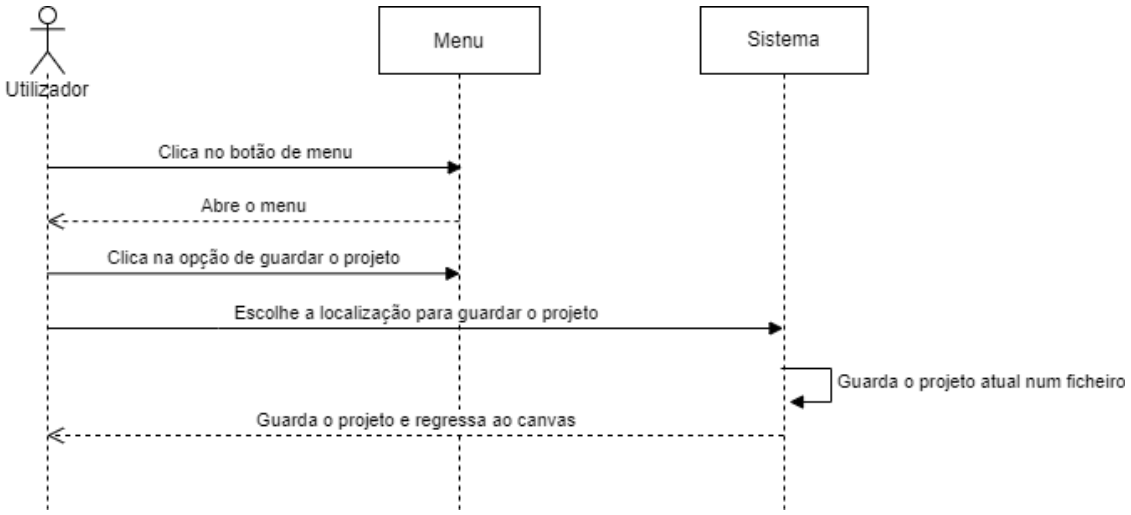


Figura 35 Diagrama de sequência do caso de uso "Guardar projeto"

4.4.7 Abrir projeto

O sétimo caso de uso é o de abrir um projeto. Neste caso, o utilizador tem de possuir um projeto anteriormente guardado para abrir e poder reutilizar ou continuar o seu trabalho.

Na Tabela 18 é apresentado o caso de uso “Abrir projeto” em detalhe. O ator deste caso de uso é o utilizador da solução. A única pré-condição é a necessidade de existir um projeto guardado previamente.

Tabela 18 Caso de uso "Abrir projeto"

Ator	Utilizador
Descrição	O utilizador pretende abrir um projeto previamente guardado
Pré-condições	É necessário existir um projeto previamente guardado
Cenário principal	1. O utilizador clica no botão de menu 2. O utilizador clica na opção de abrir um projeto 3. O utilizador escolhe a localização do projeto guardado 4. O projeto é aberto e carregado no *canvas*
Cenário alternativo	-
Pós-condições	O sistema deverá abrir o projeto guardado

A Figura 36 mostra o diagrama de sequência do caso de uso “Abrir projeto”. Para abrir um projeto, o utilizador precisa de aceder ao menu e clicar na opção de abrir projeto. Uma janela de sistema é aberta para que o utilizador possa escolher a localização do projeto guardado. Após confirmar a localização e selecionar o ficheiro, o projeto é aberto e o utilizador regressa ao *canvas*.

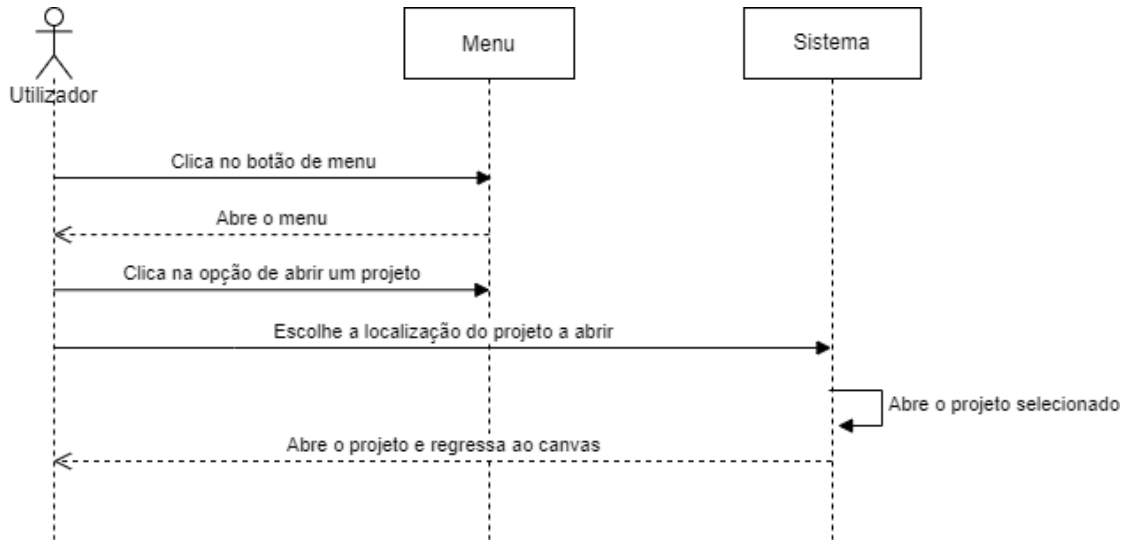


Figura 36 Diagrama de sequência do caso de uso "Abrir projeto"

4.4.8 Exportar projeto

O último caso de uso é o de exportar um projeto. A diferença deste caso de uso para o de guardar, é que neste caso o utilizador não está a guardar um ficheiro com as informações dos objetos presentes no *canvas*, mas sim uma imagem do seu trabalho.

Na Tabela 19 é apresentado o caso de uso "Exportar projeto" em detalhe. O ator deste caso de uso é o utilizador da solução. Não existem pré-condições para a realização deste caso de uso.

Tabela 19 Caso de uso "Exportar projeto"

Ator	Utilizador
Descrição	O utilizador pretende exportar o projeto atual
Pré-condições	-
Cenário principal	1. O utilizador clica no botão de menu 2. O utilizador clica na opção de exportar o projeto 3. O utilizador escolhe a localização do projeto a exportar 4. O sistema exporta o projeto e regressa ao *canvas*
Cenário alternativo	-
Pós-condições	O sistema deverá exportar o *canvas* numa imagem

A Figura 37 mostra o diagrama de sequência do caso de uso "Exportar projeto". Para exportar um projeto, o utilizador precisa de aceder ao menu e clicar na opção de exportar projeto. Uma janela de sistema é aberta para que o utilizador possa escolher a localização onde pretende exportar o projeto. Após confirmar a localização, o projeto é exportado numa imagem e o utilizador regressa ao *canvas*.

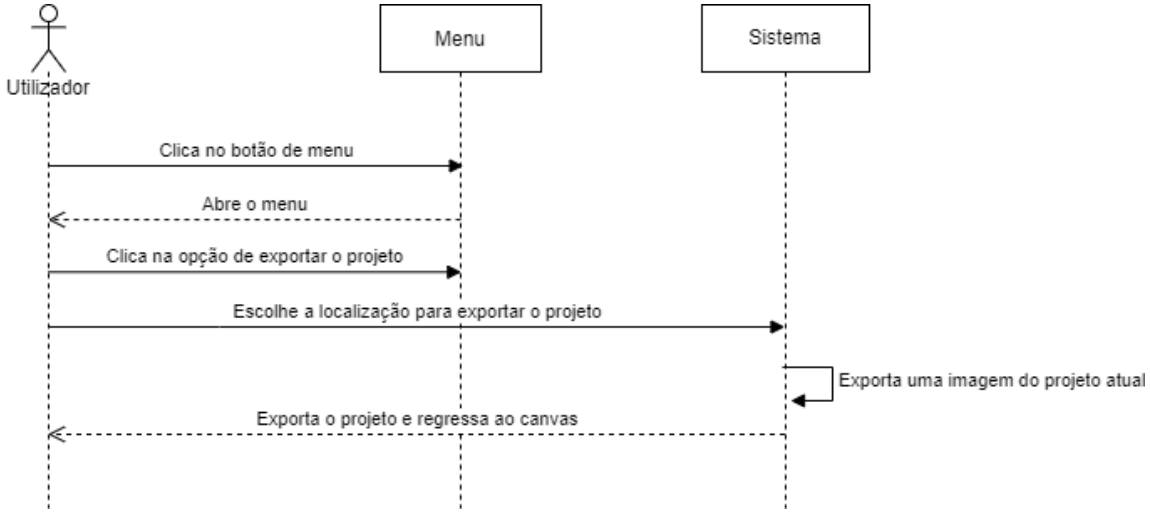


Figura 37 Diagrama de sequência do caso de uso "Exportar projeto"

4.5 Requisitos não funcionais

Por oposição aos requisitos funcionais, os requisitos não funcionais não representam uma funcionalidade em si. Para representar estes requisitos, mas também os requisitos funcionais não capturados pelos casos de uso, é utilizado o modelo FURPS+, que divide os vários requisitos em atributos de qualidade. Entre estes estão funcionalidade (F), para os requisitos funcionais não capturados pelos casos de uso, e usabilidade (U), fiabilidade (R), desempenho (P), suportabilidade (S) e outras restrições (+), para os requisitos não funcionais.

Assim, a seguinte lista apresenta todos os requisitos não funcionais levantados:

Funcionalidade

- Não foram levantados mais requisitos funcionais para além dos apresentados em 4.3.

Usabilidade

- A solução deve suportar o controlo do *canvas* através do uso de rato e teclado, assim como através de comandos de voz.

Fiabilidade

- Não foram levantados requisitos de fiabilidade.

Desempenho

- A solução deve ter um desempenho satisfatório, sem atrasos ou lentidões notórias, até pelo menos um número razoável de objetos no *canvas* (100 objetos).

Suportabilidade

- A solução deve suportar múltiplas plataformas para uma simples instalação, mas também uma variedade de projetores/monitores e microfones.

Outros

- A solução deve ser escrita numa linguagem multiplataforma para permitir um desenvolvimento mais simples e rápido em várias plataformas.

5 Implementação

Neste capítulo é apresentada e explicada a implementação da solução. São analisadas as tecnologias utilizadas e as suas dependências, assim como toda a arquitetura e estrutura do código. Por fim, são analisados os casos de uso na perspetiva da implementação.

5.1 Tecnologias

Nesta secção, são apresentadas as tecnologias utilizadas na implementação da solução em estudo. A primeira tecnologia, foi utilizada apenas num período inicial e não faz parte da solução final. Já a segunda, foi a tecnologia utilizada de forma definitiva e que correspondeu às necessidades da solução.

5.1.1 Flutter

A primeira tentativa de implementação da solução em estudo foi realizada utilizando Flutter. A ideia seria utilizar tecnologias multiplataforma que não recorressem ao uso de tecnologias *web*.

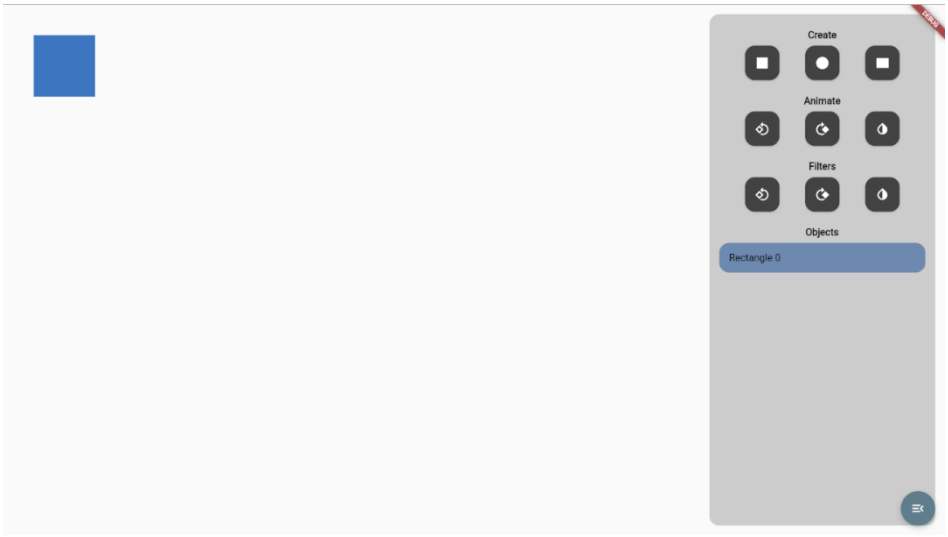


Figura 38 Solução não definitiva implementada em Flutter

Na Figura 38, é apresentada a solução implementada em Flutter. Esta é uma solução não definitiva e que foi abandonada relativamente cedo no seu desenvolvimento. As únicas funcionalidades implementadas foram a interface e a criação de objetos. Existiu uma tentativa de implementação da interação por rato, no entanto, após algumas semanas de implementação, concluiu-se ser uma tarefa demasiado complexa e dispendiosa que não iria alcançar os resultados esperados.

Assim, os motivos para o cancelamento desta implementação foram os seguintes:

- A falta de dependências que facilitassem a utilização do *canvas* disponível no Flutter. O Flutter, tal como muitas outras tecnologias, possui uma *framework* de *canvas* para desenho e animação de objetos. A *framework* em si não é limitativa, porém requer a implementação de todas as ações e interações do utilizador. Na fase de abandono desta implementação, já era possível adicionar e mover objetos, no entanto, a interação com o rato causou muitos problemas e não era aceitável para utilização.
- A falta de dependências que permitissem a utilização de comandos de voz nas várias plataformas. A maioria das dependências apenas permitiam a utilização de comandos de voz em *web* e *mobile* (Android e iOS), o que para uma solução que será utilizada maioritariamente em sistemas operativos *desktop* (Windows, macOS e Linux) era um grande entrave. A única solução para contornar esta limitação é a utilização de APIs (*Application Programming Interface*) online de transcrição, que têm a desvantagem de obrigar a utilização permanente de internet, para além do custo associado a estas.

5.1.2 React

Após o abandono da primeira tentativa de implementação, foi escolhida uma tecnologia *web* para a escrita desta solução. A tecnologia em questão é o React (já previamente estudada em 2.6.1.2).

Para a criação do projeto, foi utilizada a dependência *Create React App*. Esta dependência tem como intuito facilitar a criação de um projeto em React, utilizando tecnologias modernas que facilitam o desenvolvimento e lançamento da aplicação (Meta Open Source, 2022). Dada a simplicidade e leveza do React, não foram realizadas muitas configurações para além do que é fornecido por padrão.

O React utiliza o Node.js como base, o que permite a instalação de várias dependências, como o próprio React e outras que facilitam a criação de um canvas. À parte disso, o React é extremamente simples de configurar e utilizar, uma vez que utiliza praticamente JavaScript nativo.

Para o caso desta solução em específico, existem uma série de componentes criados para tirar proveito das vantagens da reutilização de componentes. Entre eles estão:

- *App*: Componente principal e que armazena configurações gerais como o tema e algumas variáveis de estado globais. Pode ser utilizado no futuro para, por exemplo, adicionar um ecrã de boas-vindas que pode ou não preceder o ecrã de *canvas*.
- *Canvas*: Componente central da solução, uma vez que armazena a maioria da lógica de UI do Fabric.js. Existe uma classe utilitária que partilha código não relacionado com a UI e que também é utilizada por este componente.
- *ObjectTools*: Este componente armazena as ferramentas, quer de rato, quer de voz. É onde é possível adicionar os objetos ao *canvas* e ver os objetos já adicionados.
- *ObjectSelection*: Este componente armazena as ferramentas de seleção, uma vez mais quer para o rato, quer para a voz. São as opções de edição e animação disponíveis assim que se seleciona um ou mais objetos.
- *ObjectList*: Lista de objetos que é apresentada nas *ObjectTools*.
- *VoiceControl*: Componente que permite a ativação e escuta dos comandos de voz. É apresentado em conjunto com as *ObjectTools* e *ObjectSelection* apenas quando o modo de voz está ativado, permitindo assim a gestão centralizada dos comandos de voz para estes componentes.

- Menu: É onde estão as opções de gestão de projetos e de personalização das ferramentas.

5.2 Estrutura

Nesta secção é explicada a estrutura da solução implementada, analisando também algumas das escolhas e dificuldades de cada uma das funcionalidades.

5.2.1 Canvas

O *canvas* é a principal funcionalidade da solução. Este serve para representar todos os objetos adicionados e as suas propriedades e animações. É aqui que o utilizador pode ver o resultado das suas edições.

Estando esta solução a utilizar uma tecnologia *web*, existem duas opções de *canvas* que podem ser utilizados: o *canvas* nativo do HTML5 ou o *canvas* do OpenGL. No caso concreto, foi utilizado o *canvas* do HTML5, uma vez que é de mais alto nível e, portanto, mais adequado ao tipo de representações necessárias para esta solução.

Na Figura 39 é possível ver o canvas da solução no estado inicial, ou seja, vazio.

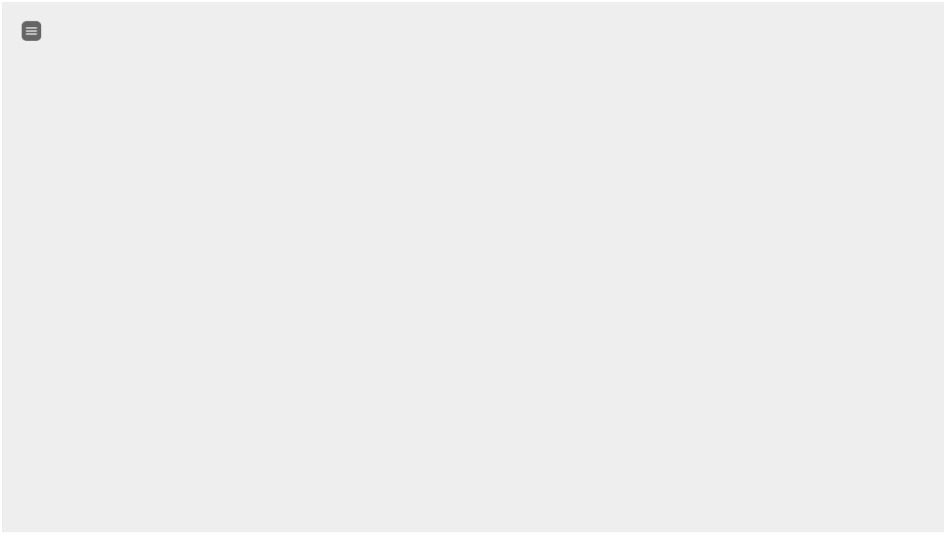


Figura 39 *Canvas* vazio

Já na Figura 40 é apresentado o *canvas* com os vários tipos de objetos disponíveis adicionados.

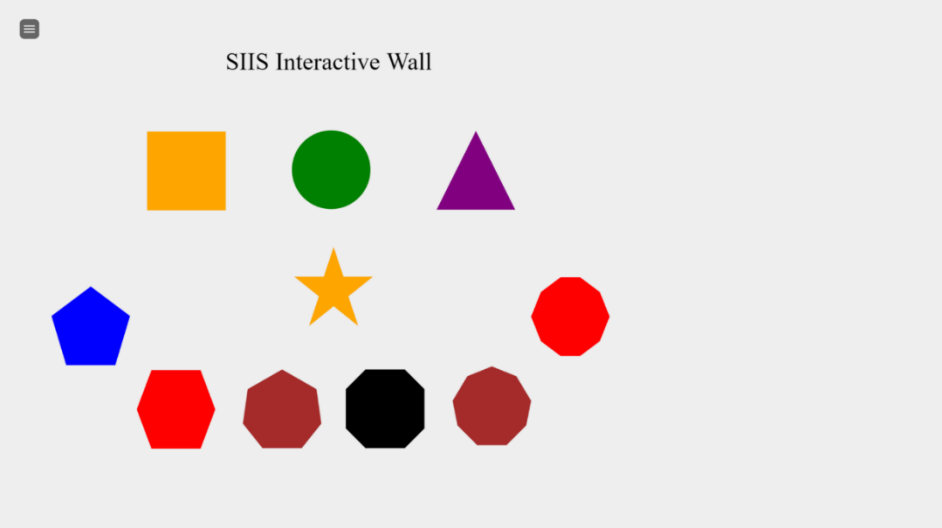


Figura 40 *Canvas* com vários tipos de objetos

5.2.1.1 Fabric.js

O Fabric.js é uma dependência que tem como objetivo criar uma abstração do *canvas* do HTML5, assim como uma série de outras adições que ajudam na utilização do *canvas* (Fabric.js, 2022).

A título de exemplo, e para dar ao leitor uma ideia do principal propósito desta dependência, nos Código 1 e Código 2 é mostrada uma comparação da criação de um *canvas* e da adição de um quadrado utilizando o *canvas* nativo e o Fabric.js.

```
var canvasEl = document.getElementById('c');

var ctx = canvasEl.getContext('2d');
ctx.fillStyle = 'red';
ctx.fillRect(100, 100, 20, 20);
```

Código 1 Criação de um canvas e adição de um quadrado no canvas HTML

```
var canvas = new fabric.Canvas('c');

var rect = new fabric.Rect({
  left: 100,
  top: 100,
  fill: 'red',
  width: 20,
  height: 20
});

canvas.add(rect);
```

Código 2 Criação de um canvas e adição de um quadrado no Fabric.js

Neste caso, as diferenças podem não ser imediatamente óbvias, mas assim que se introduzem transformações e animações, estas começam a ser relevantes.

Para além destas questões de código, esta dependência oferece também grandes vantagens na interação do utilizador com a solução. A interação dos objetos e a sua transformação com um simples clique é apenas um exemplo.

5.2.1.2 Three.js

O Three.js é uma biblioteca 3D que utiliza WebGL para desenhar conteúdo 3D numa página *web*. Uma vez que o WebGL é uma biblioteca de muito baixo nível, o que requer a implementação de praticamente todo e qualquer objeto a desenhar, esta biblioteca permite a criação de cenas, luzes, sombras, materiais, texturas, entre outros, de forma mais simples e direta (Three.js, 2022).

5.2.1.3 PixiJS

O PixiJS é uma biblioteca puramente 2D, mas utiliza, ao contrário da grande parte das outras bibliotecas 2D, WebGL para desenhar o conteúdo apresentado. Torna-se uma solução muito interessante, porque permite a criação de objetos 2D como todas as outras bibliotecas, mas apresenta um desempenho muito superior às bibliotecas que apenas utilizam o canvas HTML para desenhar o conteúdo (PixiJS, 2022).

5.2.1.4 Porquê o Fabric.js?

Ultimamente, a decisão entre estes *canvas* deveu-se a uma combinação entre as funcionalidades disponíveis e o desempenho.

Para avaliar o desempenho de cada uma destas bibliotecas, foi utilizado um *website* que permite executar cenários de *stress* em cada um destes *canvas*. Para cada uma das bibliotecas, foi executado um teste num computador com baixo desempenho, o que permite avaliar o pior cenário, com mil objetos em cena.

No caso do Fabric.js, o desempenho foi satisfatório, com uns estáveis 30 fotogramas por segundo. O Three.js foi o que obteve um pior desempenho, ainda que com uma média de 30 fotogramas por segundo também. Já o caso do PixiJS foi o que obteve o melhor desempenho de todos, com uns estáveis 60 fotogramas por segundo.

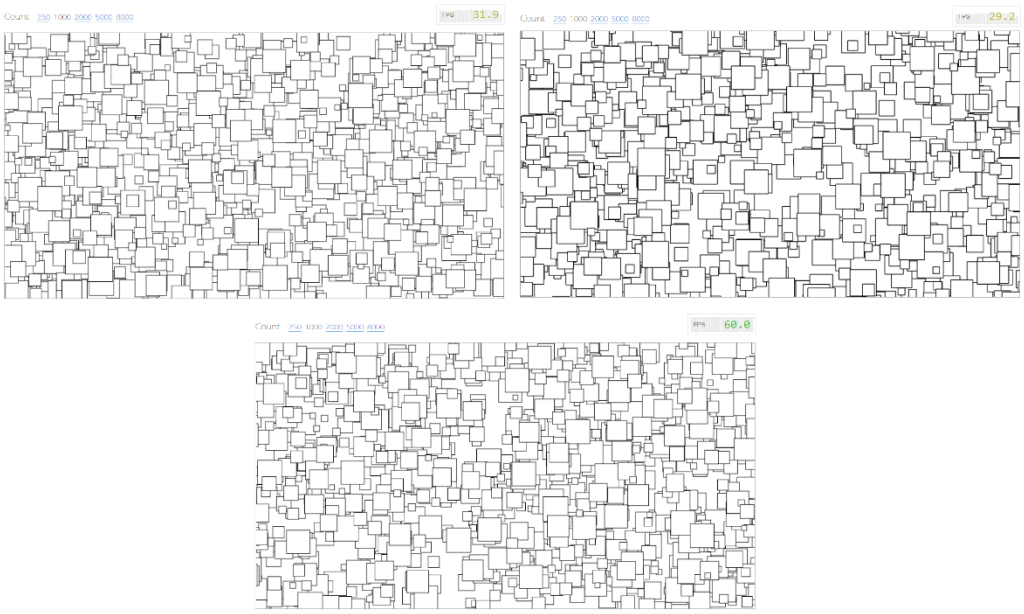


Figura 41 Comparativo de desempenho das bibliotecas de *canvas*[12]

5.2.2 Ferramentas

As ferramentas é a funcionalidade que permite a interação do utilizador com o *canvas*. Para além das funcionalidades nativas do Fabric.js, é possível adicionar, editar, animar e remover objetos a partir desta secção. As ferramentas estão disponíveis para utilização com rato ou comandos de voz.

Na implementação, existe uma separação lógica entre as ferramentas que permitem a criação de objetos (*ObjectTools*) e as ferramentas que permitem a edição e animação dos objetos (*ObjectSelection*). Tal como o nome pretende indicar, as primeiras são as apresentadas quando a solução é aberta e as segundas são apresentadas quando um objeto é selecionado.

Nas Figura 42 e Figura 43 é possível observar as ferramentas de objeto para utilização com rato e com comandos de voz, respetivamente.

[12] Comparativo obtido de https://benchmarks.slaylines.io/

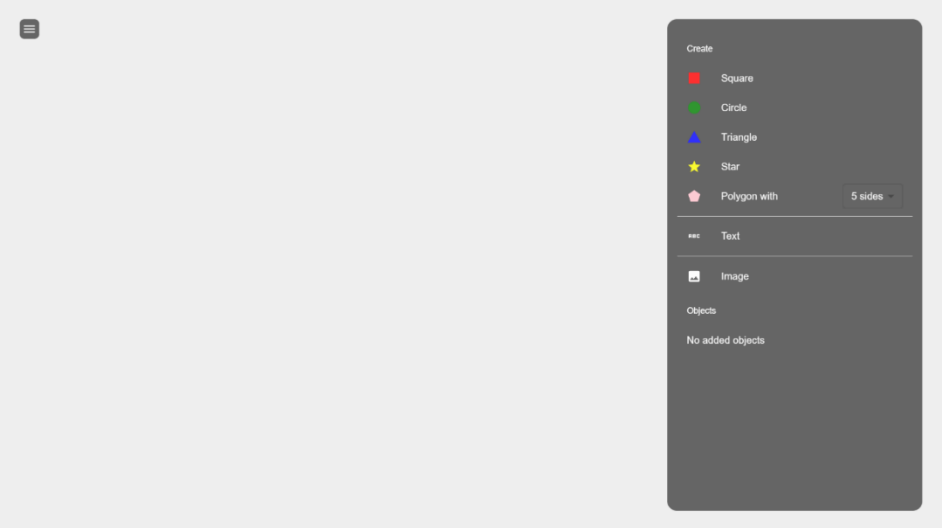


Figura 42 Ferramentas de objeto para rato

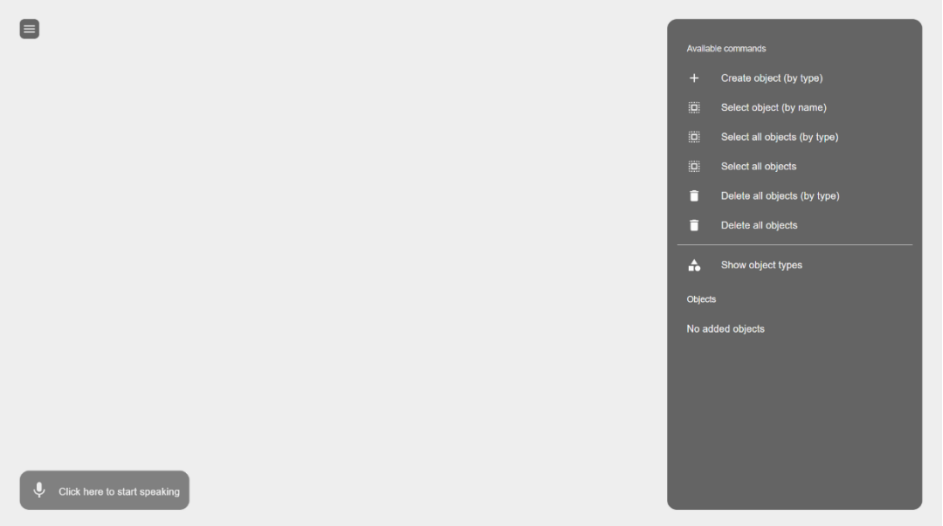


Figura 43 Ferramentas de objeto para voz

E nas Figura 44 e Figura 45 é possível observar as ferramentas de seleção também para utilização com rato e com comandos de voz, respetivamente.

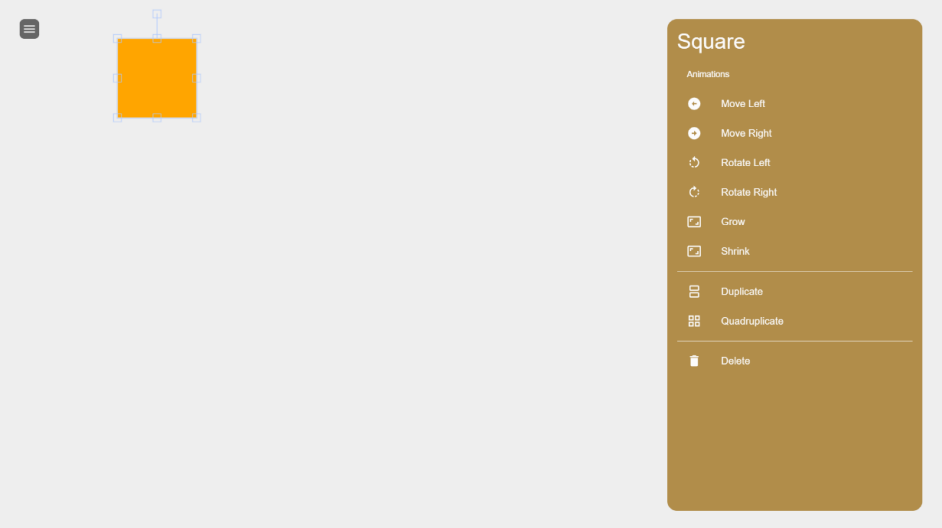


Figura 44 Ferramentas de seleção para rato

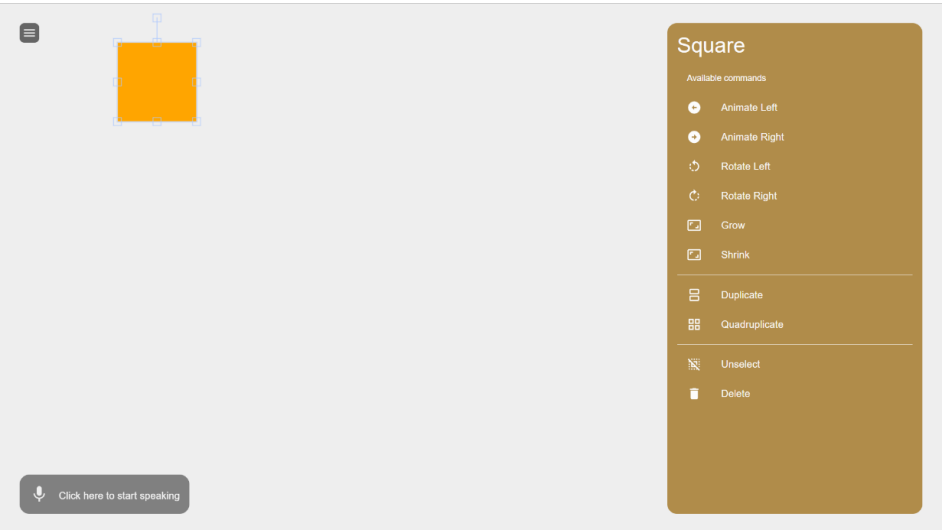


Figura 45 Ferramentas de seleção para voz

Ainda relacionado com as ferramentas estão outros dois componentes independentes: a lista de objetos (*ObjectList*) e o botão de ativação dos comandos de voz (*VoiceControl*).

A lista de objetos apresenta todos os objetos adicionados ao *canvas*, por ordem de adição. Cada objeto presente na lista é representado pelo seu nome e a pela sua cor, para facilitar a sua distinção.

O botão de ativação dos comandos de voz agrega toda a lógica de ativação/desativação do microfone, assim como o processamento e distribuição dos comandos de voz para o componente correto.

5.2.3 Menu

O menu apresenta as opções mais gerais da solução. Isto é, criar, guardar ou abrir um projeto, exportar uma imagem do *canvas*, abrir ou fechar as ferramentas e alternar entre os seus dois modos, e a janela acerca da solução são as opções disponíveis neste menu.

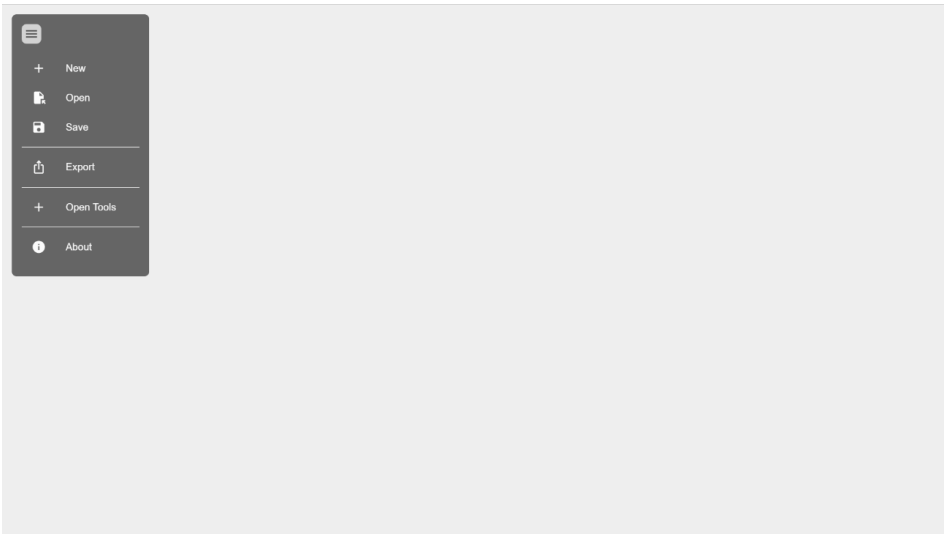


Figura 46 Menu da solução

5.2.4 Comandos de voz

Os comandos de voz adicionados foram selecionados com base nas funcionalidades mais relevantes para o utilizador quando estes estão a interagir com a solução. Existem duas secções da solução nas quais são permitidos o uso de comandos de voz: nas ferramentas e na seleção de objeto.

Assim, foram criados comandos de voz para as seguintes ações:

- Criar um objeto do tipo pretendido;
- Selecionar o objeto com o nome pretendido;
- Selecionar todos os objetos de um determinado tipo;
- Selecionar todos os objetos no *canvas*;
- Eliminar todos os objetos de um determinado tipo;
- Eliminar todos os objetos do *canvas*;
- Apresentar uma lista dos tipos de objeto disponíveis;
- Iniciar a animação de mover para a esquerda;

- Iniciar a animação de mover para a direita;
- Iniciar a animação de rotação para a esquerda;
- Iniciar a animação de rotação para a direita;
- Iniciar a animação de crescimento do objeto;
- Iniciar a animação de redução do objeto;
- Duplicar o objeto selecionado;
- Quadruplicar o objeto selecionado em matriz;
- Aplicar o filtro sépia (apenas para imagens);
- Aplicar o filtro vermelho (apenas para imagens);
- Aplicar o filtro verde (apenas para imagens)
- Aplicar o filtro azul (apenas para imagens)
- Desselecionar o objeto atual;
- Eliminar o objeto atual.

Apesar de os comandos de voz apenas funcionarem nestas secções da solução, a implementação de mais comandos é uma tarefa facilmente alcançável, podendo estes serem estendidos, por exemplo, ao menu.

Na análise apresentada em 2.6.1 é realçada a utilização extensiva da tecnologia Electron na solução. Apesar de o Electron não impactar de alguma forma a funcionalidade geral da solução, este cria um problema nos comandos de voz. Quando se utiliza tecnologias de desenvolvimento *web*, estas utilizam um navegador, por norma o Google Chrome ou baseado em Chromium devido à sua relação íntima com o motor utilizado tanto pelo Node.js como pelo Electron, e por isso possuem capacidade nativa de ditado/transcrição. O problema é que, recentemente, a Google removeu o suporte a esta capacidade para as aplicações geradas em Electron. Desta forma, a única maneira de utilizar comandos de voz é recorrendo a uma ferramenta de terceiros, como uma API online. Apesar de existirem uma série de empresas a fornecer este tipo de soluções, estas possuem duas desvantagens: a necessidade de pagamento para uma utilização razoável e a permanente necessidade de internet (Amazon, 2022) (Google, 2022) (Microsoft, 2022).

Assim, foi utilizada uma dependência que utiliza as capacidades de transcrição nativas do navegador ao mesmo tempo que permite a utilização de uma API alternativa, no caso a API de voz do Azure da Microsoft. Esta solução é a mais adequada, uma vez que permite a utilização

da solução gerada para uma aplicação Electron, enquanto permite a sua utilização direta num navegador. Para além disso, é uma API francamente melhor e mais personalizável o que permite otimizar os comandos às necessidades da solução.

5.3 Casos de uso

Com base no que foi apresentado e analisado na arquitetura, nesta secção é detalhada, para cada caso de uso, a sua implementação e os vários desafios e dificuldades inerentes.

5.3.1 Adicionar objeto

Para a implementação do caso de uso "Adicionar objeto", foram utilizadas as capacidades do Fabric.js. Assim sendo, existem uma série de funções predefinidas para adicionar os objetos mais comuns (retângulos, círculos, triângulos, etc.).

No caso específico do Código 3, para adicionar um quadrado é criado um objeto do tipo *rect* que recebe uma série de parâmetros que o permitem personalizar: neste caso, o nome, a posição, o tamanho e a cor. Adicionalmente é realizada uma extensão à função *toObject* do objeto, apenas para permitir que o seu nome seja também guardado no projeto (por padrão, isto não acontece). Por fim, adiciona-se ao *canvas* permitindo, assim, a sua visualização.

```
const addSquare = () => {
  const rect = new fabric.Rect({
    name: nameObjectWith("Square"),
    originX: "center",
    originY: "center",
    left: 256,
    top: 128,
    fill: getRandomColor(),
    width: 128,
    height: 128,
  });

  rect.toObject = (function (toObject) {
    return function () {
      return fabric.util.object.extend(toObject.call(this), {
        name: this.name,
      });
    };
  })(rect.toObject);

  addObject(rect);
};
```

Código 3 Função de adicionar um quadrado

No entanto, para os objetos mais complexos é possível desenhá-los manualmente definindo os seus pontos. No caso específico da solução, esta opção foi utilizada para as estrelas e para os polígonos com vários lados. Tal como é mostrado nos Código 4 e Código 5, a criação deste objeto é muito semelhante à do quadrado, com a diferença de o seu tipo ser *polygon* e, no parâmetro adicional que recebe, é-lhe passada uma lista de pontos que permite saber exatamente como desenhar o polígono em questão.

```
const addStar = () => {
  const star = new fabric.Polygon(starPoints(), {
    name: nameObjectWith("Star"),
    originX: "center",
    originY: "center",
    fill: getRandomColor(),
    left: 256,
    top: 128,
  });

  star.toObject = (function (toObject) {
    return function () {
      return fabric.util.object.extend(toObject.call(this), {
        name: this.name,
      });
    };
  })(star.toObject);

  addObject(star);
};
```

Código 4 Função de adicionar uma estrela

```
export const starPoints = () => {
  return [
    { x: 64, y: 0 },
    { x: 80, y: 48 },
    { x: 128, y: 48 },
    { x: 88, y: 80 },
    { x: 104, y: 128 },
    { x: 64, y: 96 },
    { x: 24, y: 128 },
    { x: 40, y: 80 },
    { x: 0, y: 48 },
    { x: 48, y: 48 },
  ];
};
```

Código 5 Função de desenhar os pontos de uma estrela

Ao adicionar um objeto, este aparece no *canvas* tal como mostra a Figura 47. De notar que o nome e a cor do objeto são gerados automaticamente. O nome é gerado tendo em conta o tipo de objeto (*Square*, por exemplo) seguido de um número para os distinguir (o primeiro objeto

não tem número). A cor é gerada aleatoriamente, mediante um conjunto dez cores predefinidas, por forma a simplificar a sua alteração por voz.

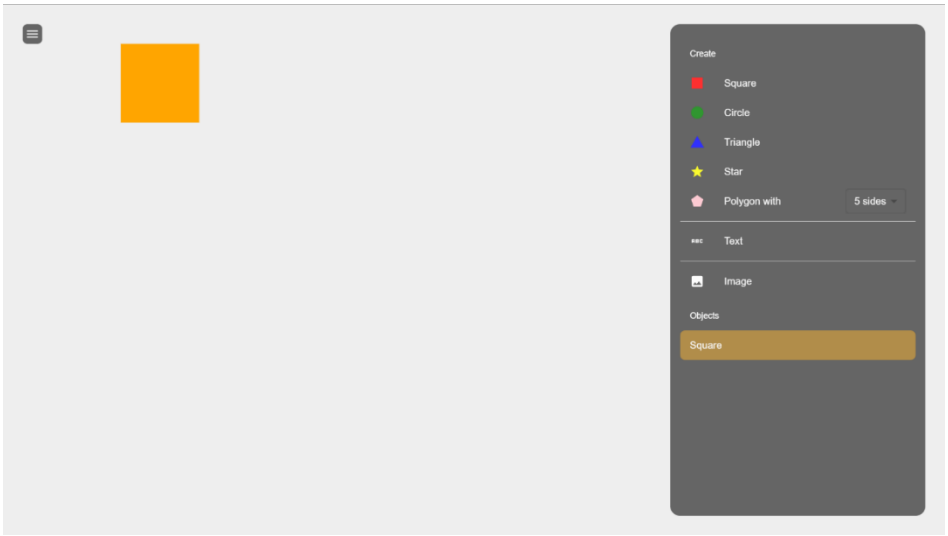


Figura 47 Objeto adicionado ao *canvas*

5.3.2 Editar objeto

Para a implementação do caso de uso "Editar objeto", foram criadas várias funções para as várias edições disponíveis.

Neste caso em particular, existem muitas edições que são fornecidas "*out of the box*" pelo próprio Fabric.js. Entre outras, estão as edições de selecionar um objeto e de mover e alterar o tamanho do objeto selecionado.

No entanto, existem edições que foram manualmente implementadas. No caso, duplicar e quadruplicar, assim como aplicar diversos filtros (apenas disponível em imagens por limitação do próprio *canvas*).

O Código 6 mostra a função de duplicar um objeto. Para isso, é utilizada a função *clone* do Fabric.js que gera uma cópia do objeto selecionado. A esta cópia é alterada a sua posição (100 píxeis para direita para não se sobrepor) e o seu nome (utilizando a mesma metodologia de um novo objeto).

```
const duplicate = (object) => {
  object.clone((clone) => {
    clone.left += object.width + 100;
    clone.name = nameObjectWith(object.name.split(" ")[0]);
```

```
    addObject(clone);
  });
};
```

Código 6 Função de duplicar um objeto

Já na Figura 48 é possível ver o objeto adicionado na Figura 47, mas com o seu tamanho, largura, posição e rotação alterados. Estas alterações são alcançáveis através da caixa de seleção que aparece em volta do objeto quando este é selecionado. De notar que estas edições estão disponíveis em comandos de voz (para o modo de voz) e para múltiplos objetos.

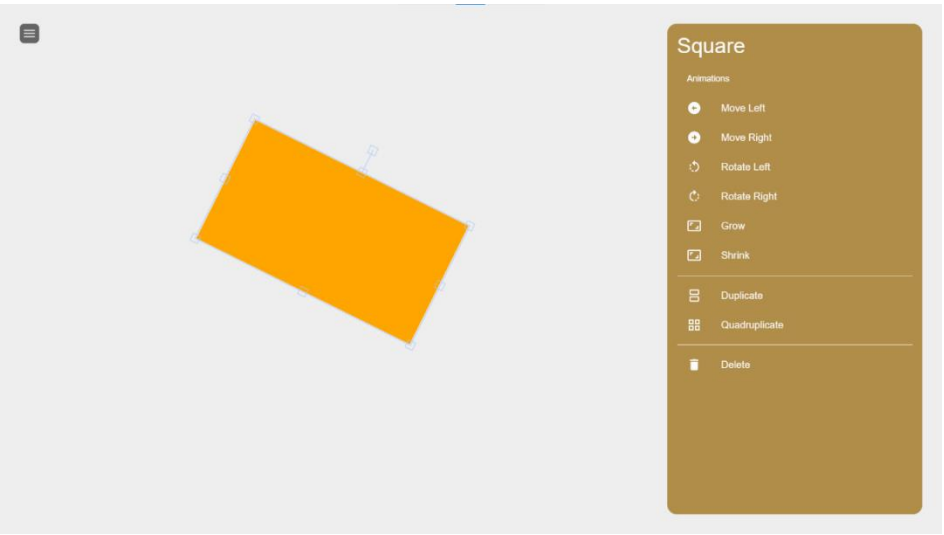


Figura 48 Objeto editado manualmente

5.3.3 Animar objeto

Para a implementação do caso de uso "Animar objeto", foram mais uma vez utilizadas as capacidades nativas do Fabric.js. Em concreto, esta dependência inclui uma função *animate* que pode ser personalizada de acordo com a animação pretendida.

No caso do Código 7, o parâmetro *"left"* é passado com o valor de "-=512" o que diz ao Fabric.js que deve mover o objeto 512 píxeis para a esquerda.

```
const animateLeft = (object) => {
  object.animate("left", "-=512", {
    duration: 2000,
    onChange: canvas.renderAll.bind(canvas),
    onComplete: () => {
      animateRight(object);
    },
    easing: fabric.util.ease.easeInOutQuad,
```

```
    });
};
```

Código 7 Função de animação para a esquerda do objeto

Existem, no entanto, outras opções como *"angle"* para alterar a rotação do objeto e "*radius*", "*scale*" ou "*height/width*", dependendo do seu tipo, para alterar o tamanho do objeto.

Para além do tipo de animação e do seu valor, a função *animate* recebe como terceiro parâmetro um objeto que permite indicar uma série de detalhes sobre a animação. No exemplo, estão a duração da animação, o tipo de suavização e as funções executadas a cada fotograma (*onChange*) e no fim da animação (*onComplete*). A primeira chama a função *renderAll* obrigatória para que a animação do objeto seja visualizada, caso contrário apenas o valor é atualizado. A segunda, é utilizada para inverter a animação, ou seja, ao terminar a animação para a esquerda, começa a animação para a direita, voltando, por isso, à sua posição original. Esta estratégia é utilizada em todas as animações e permite criar uma animação infinita.

Na Figura 49 é possível ver o quadrado já apresentado anteriormente, a mover-se para a esquerda, a rodar para a esquerda e a crescer. Esta figura foi capturada num fotograma aleatório, porém permite visualizar abstratamente a animação pretendida. De notar que estas animações estão disponíveis em comandos de voz (para o modo de voz) e para múltiplos objetos.

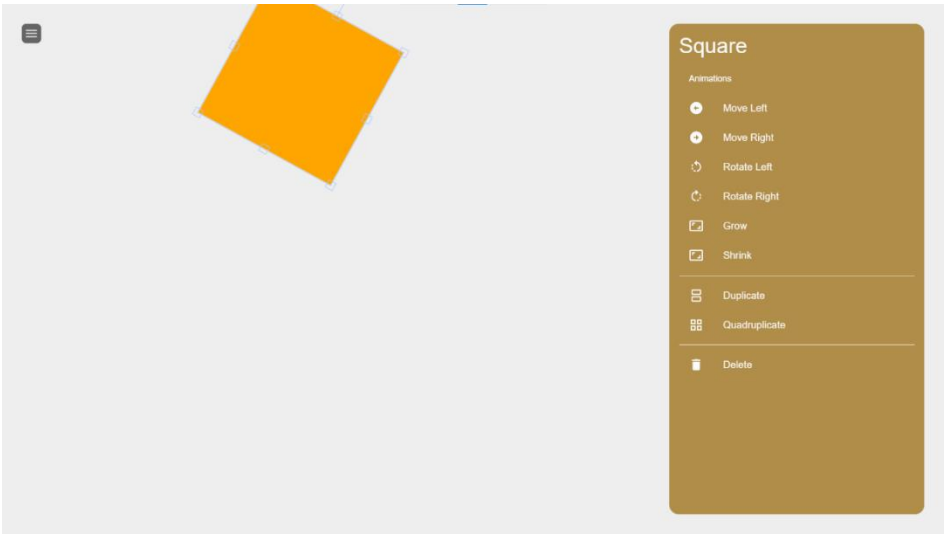


Figura 49 Objeto animado automaticamente

5.3.4 Remover objeto

Para a implementação do caso de uso “Remover objeto”, foi utilizada mais uma funcionalidade do Fabric.js. Em primeiro lugar, é desselecionado o objeto atual para garantir que caixa de seleção desaparece com o objeto. Em segundo lugar, é eliminado o objeto e forçado o canvas a recarregar.

```
const deleteObject = (object) => {
  unselectObject();
  canvas.remove(object);
  canvas.requestRenderAll();
};
```

Código 8 Função de remover um objeto

Na Figura 50 é possível ver a opção de remoção do objeto selecionado. Assim que se clica na opção, o objeto e a caixa de seleção desaparecem. De notar que esta ação está disponível em comando de voz (para o modo de voz) e para múltiplos objetos.

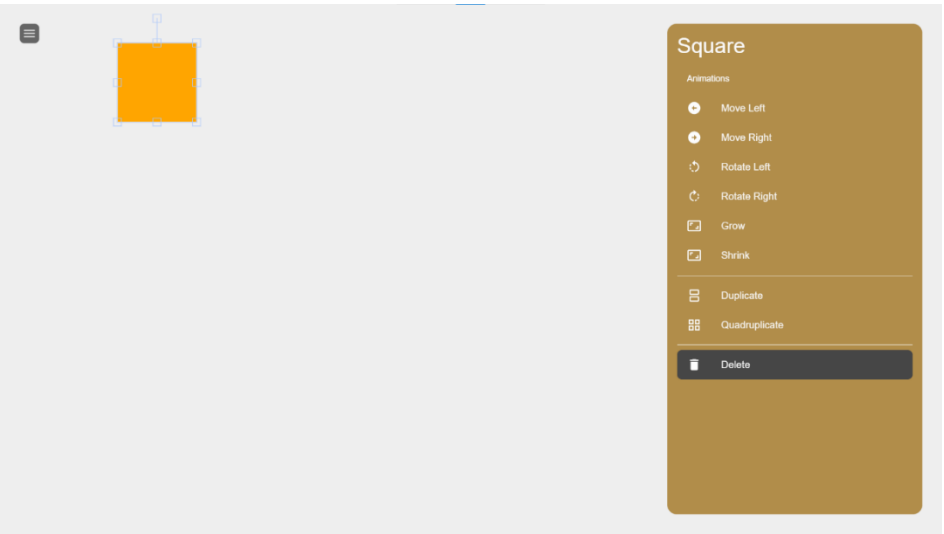


Figura 50 Opção de remover objeto

5.3.5 Criar projeto

Para a implementação do caso de uso “Criar projeto”, existe uma funcionalidade muito simples do Fabric.js que limpa todo o *canvas*. Esta ação é suficiente, uma vez que toda a gestão dos objetos é realizada pela própria dependência.

```
const clearCanvas = () => canvas.clear();
```

Código 9 Função de limpar o *canvas*

Quando se inicia a solução pela primeira vez (ou recarrega), o *canvas* é também reinicializado.

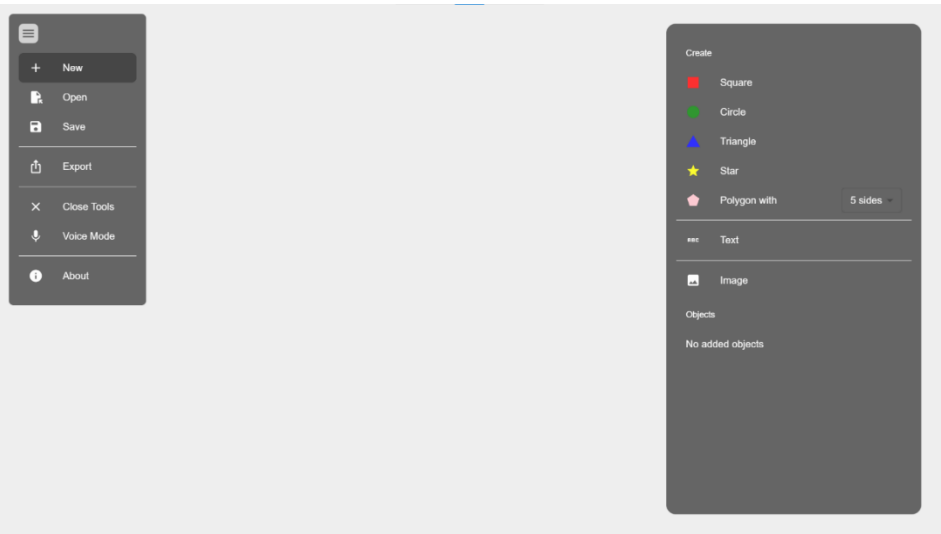


Figura 51 Opção de criar um projeto

5.3.6 Guardar projeto

Para a implementação do caso de uso "Guardar projeto", são necessários alguns passos.

Em primeiro lugar é necessário apresentar uma janela para o utilizador confirmar a localização do projeto a guardar. Para isso é chamada a função "*showSaveFilePicker*" que recebe como parâmetro o tipo de ficheiros aceites, neste caso ficheiros JSON (*JavaScript Object Notation*).

Em seguida, é criado um objeto (*writable*) que permite escrever o ficheiro JSON para a localização escolhida pelo utilizador. Este ficheiro JSON contém toda a informação presente no *canvas*. Isto é possível porque o objeto *canvas* suporta a conversão direta para o formato JSON. O formato JSON é, na verdade, exatamente igual à notação de um objeto JavaScript, daí o seu nome.

```
const saveCanvas = async () => {
  const fileHandle = await window.showSaveFilePicker({
    types: [
      {
        description: "",
        accept: { "application/json": [".json"] },
      },
    ],
  });
```

```
  const fileStream = await fileHandle.createWritable();
  await fileStream.write(
    JSON.stringify(canvas) + "\n" + JSON.stringify(fabric.runningAnimations)
  );
  await fileStream.close();
};
```

Código 10 Função de guardar um projeto

Na Figura 52 é possível ver a opção de guardar um projeto presente no menu.

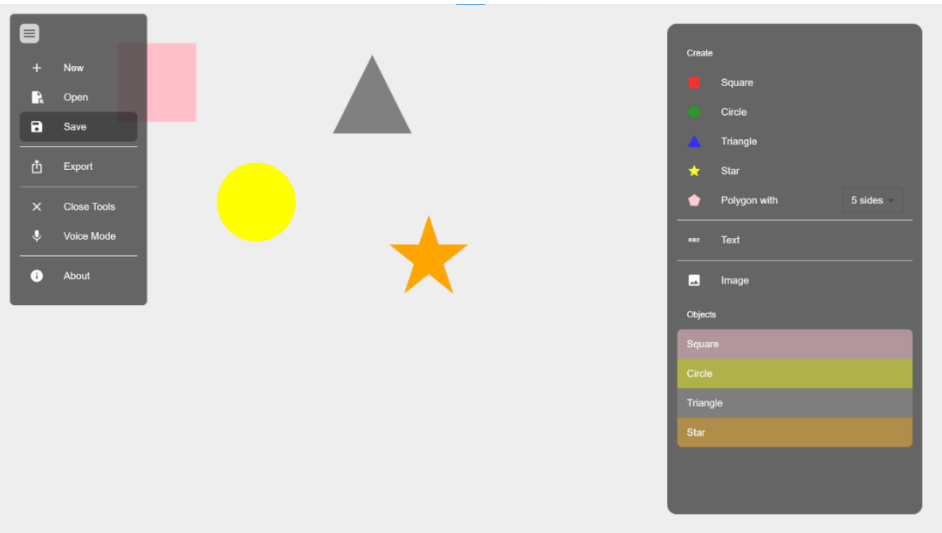


Figura 52 Opção de guardar um projeto

Já na Figura 53 é possível ver a janela de sistema que é aberta para que o utilizador possa escolher a localização onde guardar o projeto.

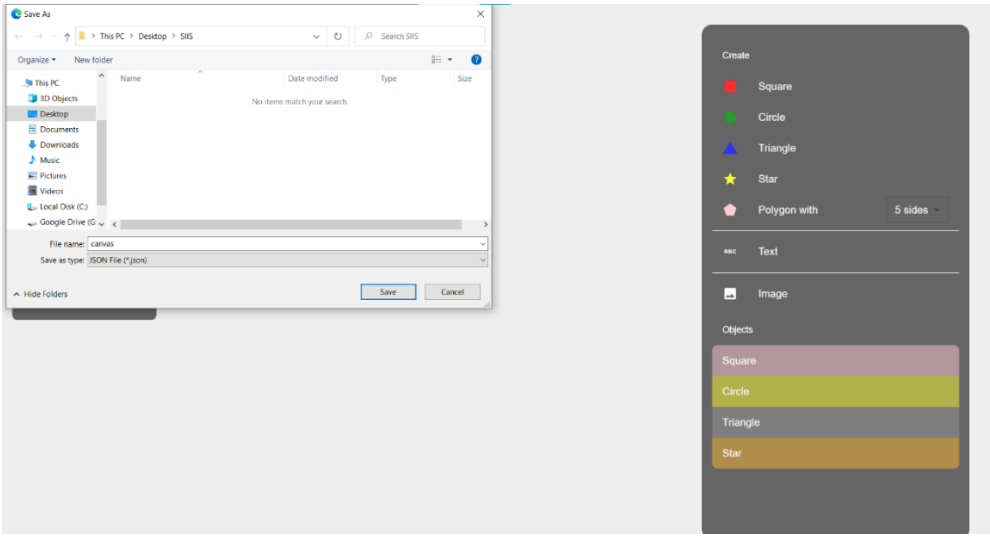


Figura 53 Janela de seleção de localização do objeto a guardar

5.3.7 Abrir projeto

Para a implementação do caso de uso “Abrir projeto”, são mais uma vez necessários alguns passos. A fluxo de ações é muito semelhante ao do caso de uso anterior, mas a forma de o atingir é diferente.

Em primeiro lugar é criado um objeto *FileReader* que vai receber o ficheiro do projeto a abrir. Este ficheiro é obtido de forma diferente, uma vez que, ao contrário do caso anterior, é necessário conhecer a localização do ficheiro previamente.

Em seguida, e mal o ficheiro esteja carregado em memória, é realizado o inverso do caso de uso anterior e o ficheiro JSON é novamente convertido num objeto *canvas*. Por fim são recarregados todos os objetos para possibilitar a sua visualização imediata.

```
const openCanvas = (saveFile) => {
  const fileReader = new FileReader();
  fileReader.readAsText(saveFile);

  fileReader.onload = () => {
    canvas.loadFromJSON(fileReader.result.split("\n")[0]);

    JSON.parse(fileReader.result.split("\n")[1]).forEach((animation) => {
      fabric.runningAnimations.push(animation);
    });

    canvas.requestRenderAll();
  };
};
```

Código 11 Função de abrir um projeto

Na Figura 54 é possível ver a opção de abrir um projeto disponível no menu.

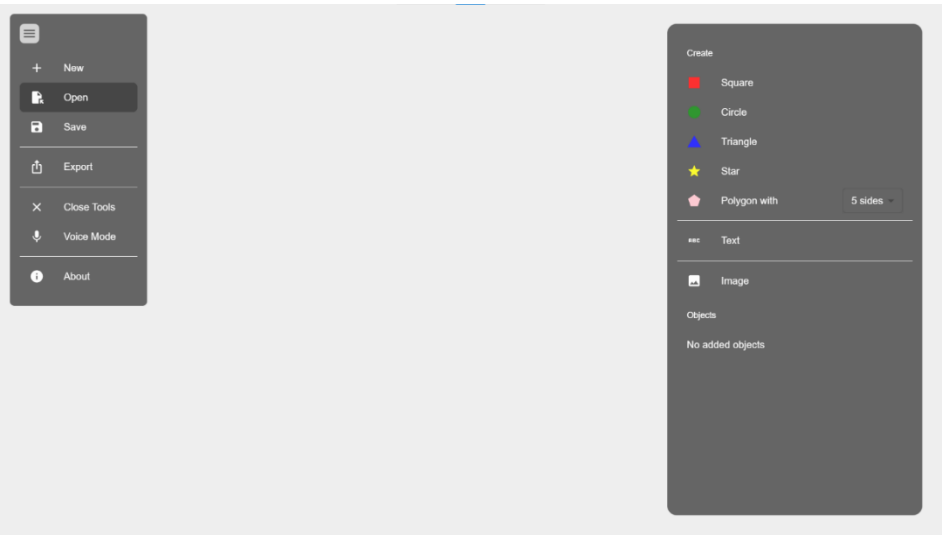


Figura 54 Opção de abrir um projeto

Já na Figura 55 é possível ver a janela de sistema que é aberta para que o utilizador possa escolher a localização do projeto a abrir.

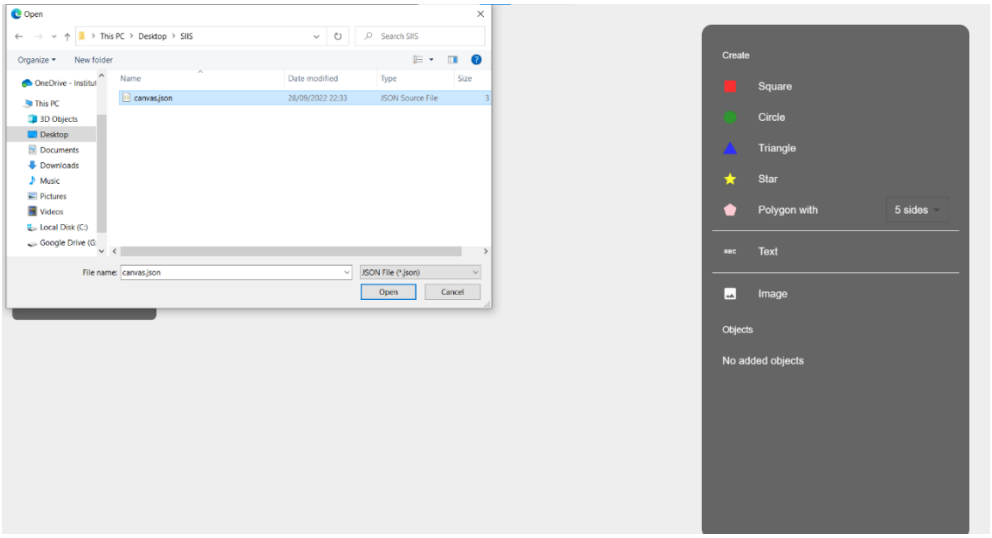


Figura 55 Janela de seleção da localização do objeto a abrir

Após os passos anteriores, a Figura 56 mostra o projeto aberto e carregado novamente no *canvas*. De notar que existe uma limitação com as animações, uma vez que estas são também exportadas para o ficheiro, mas não são carregadas de volta por limitação do Fabric.js.

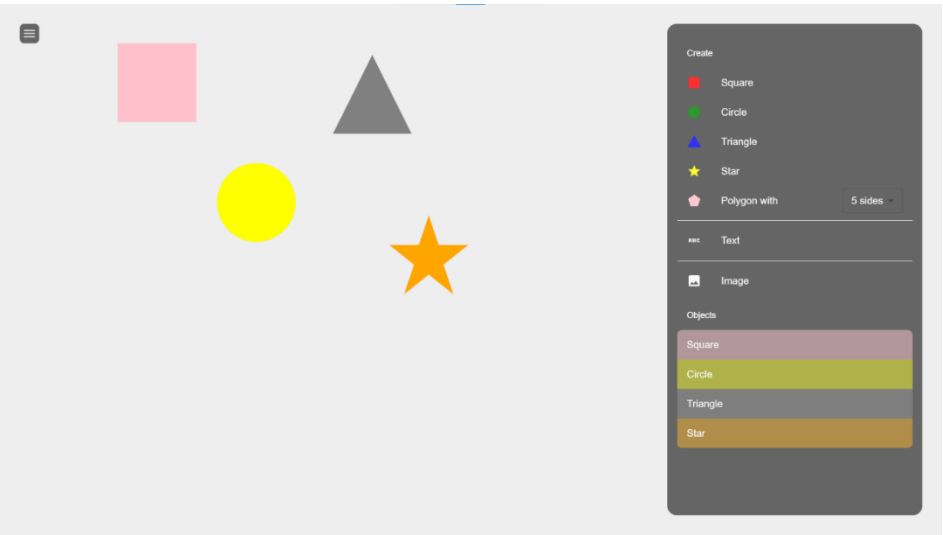


Figura 56 Projeto aberto atráves de um ficheiro

5.3.8 Exportar projeto

Para a implementação do caso de uso “Exportar projeto”, foi criada uma função que gera uma imagem do *canvas* atual.

Em primeiro lugar, o *canvas* é convertido num *Data URL* que contém uma representação em imagem do canvas em formato PNG (*Portable Network Graphics*) (parâmetro predefinido) e com um multiplicador de duas vezes permitindo que a imagem seja de maior dimensão.

Em seguida, é criado um objeto (*blob*) que utiliza o URL (*Uniform Resource Locator*) anteriormente criado num objeto em bruto contendo todas as informações necessárias para escrever uma imagem.

Por fim, é realizada precisamente a mesma ação que no caso de uso em 5.3.6, com a única diferença de que a janela de guardar só aceita ficheiros do formato PNG.

```
const exportCanvas = async () => {
  const dataURL = canvas.toDataURL({ multiplier: 2 });
  const blob = await fetch(dataURL).then((r) => r.blob());

  const fileHandle = await window.showSaveFilePicker({
    types: [
      {
        description: "",
        accept: { "image/png": [".png"] },
      },
    ],
  });
```

```
  const fileStream = await fileHandle.createWritable();
  await fileStream.write(blob);
  await fileStream.close();
};
```

Código 12 Função de exportar um projeto

Na Figura 57 é possível ver a opção de exportar um projeto disponível no menu.

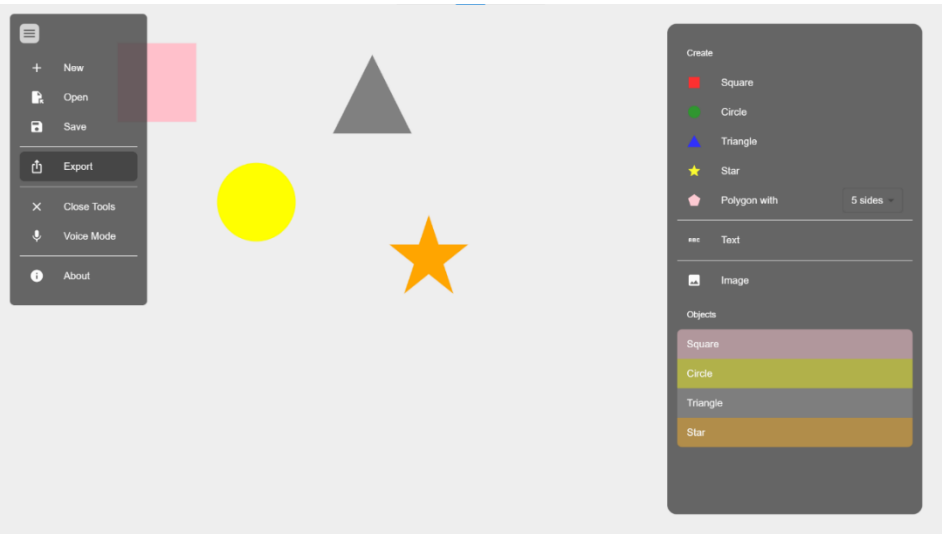


Figura 57 Opção de exportar um projeto

Já na Figura 58 é possível ver a janela de sistema que é aberta para que o utilizador possa escolher a localização onde exportar a imagem do *canvas*.

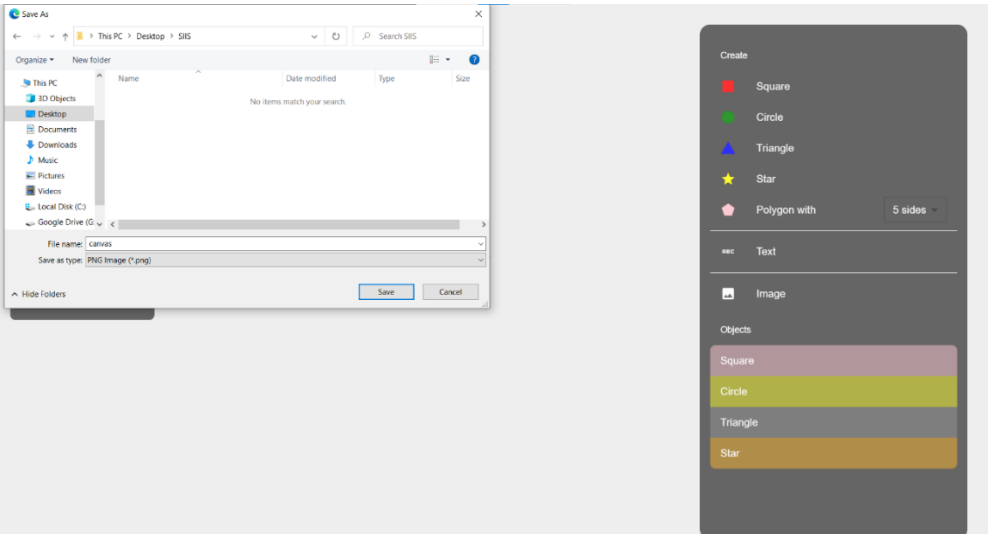


Figura 58 Janela de seleção da localização do objeto a exportar

E na Figura 59 é mostrada a imagem exportada, com todos os objetos presentes no *canvas* e com a dimensão do *canvas* no momento da exportação.

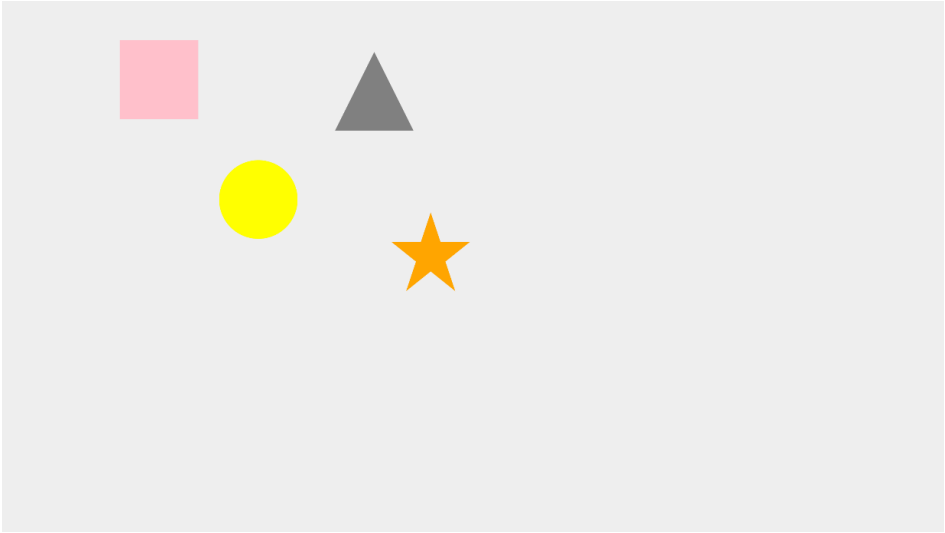


Figura 59 Imagem do projeto exportado

6 Avaliação

Neste capítulo é analisado como é avaliada a solução e quais metodologias são utilizadas para quantificar a sua qualidade.

6.1 Metodologia

Para testar e avaliar a solução em estudo, foram realizadas várias apresentações durante e no final do desenvolvimento da solução protótipo.

6.1.1 Apresentações

Foram realizadas duas apresentações em fases distintas do desenvolvimento. A primeira foi realizada a 17 de junho de 2022 para o grupo de investigadores do SIIS, nas suas instalações. Dado o estado ainda prematuro da solução, a apresentação foi controlada, não tendo havido lugar à participação direta dos assistentes, nem a questionários de usabilidade.

Nas Figura 60 e Figura 61, é possível observar a apresentação em curso, onde é possível ver os vários objetos no canvas, assim como o monitor do computador que controla a solução.

Figura 60 Apresentação da solução (projeção)

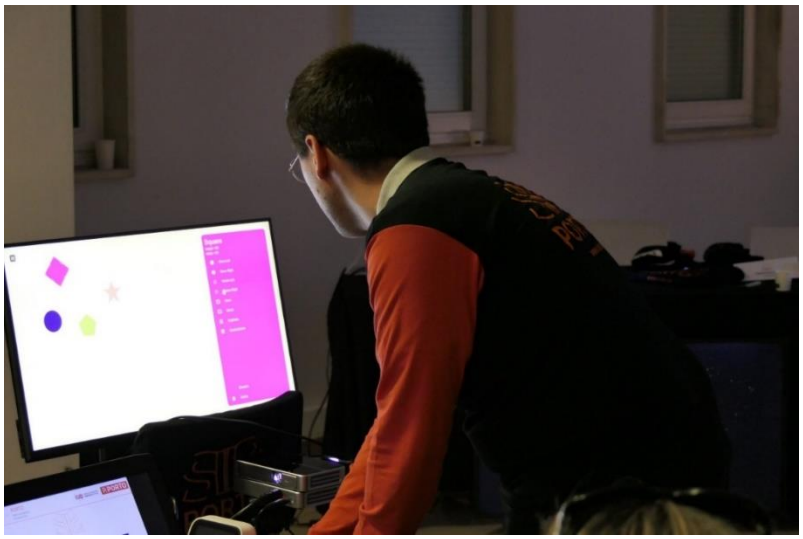


Figura 61 Apresentação da solução (monitor)

A segunda foi realizada a 3 de outubro de 2022, para um grupo de alunos do ISEP. Nesta apresentação, a apresentação foi significativamente mais extensa e detalhada e houve lugar à participação dos assistentes. Dadas as condições do ambiente de apresentação, a participação focou-se principalmente no teste dos comandos de voz da solução.

6.1.2 Questionário de usabilidade

Para medir a usabilidade da solução, foi realizado um questionário a um grupo de utilizadores. Dada as necessidades de instalação da solução, o questionário foi realizado num ambiente

controlado e com uma amostra relativamente pequena. Este questionário é composto por quinze questões (três de avaliação do público-alvo e doze de avaliação da solução) e tem por objetivo avaliar questões gerais sobre a usabilidade.

O questionário foi realizado nos dias 3 e 4 de outubro de 2022, com as seguintes questões:

1. Qual é o seu género?
2. Qual é a sua idade?
3. Quais são as suas habilitações literárias?
4. Assistiu a uma demonstração da parede interativa controlada por voz?
5. Considera que uma solução desta natureza é interessante e inovadora?
6. Já teve alguma experiência em interação por voz?
7. Considera interessantes os sistemas de interação por voz?
8. A interface é simples e intuitiva para novos utilizadores?
9. Os comandos de voz são complicados de entender?
10. A solução compreendeu corretamente os comandos?
11. As animações dos objetos geram efeitos interessantes?
12. O número de edições e animações é suficiente?
13. O número de comandos de voz é suficiente?
14. Como classifica o potencial da solução apresentada?
15. Tem alguma sugestão de funcionalidade ou melhoria que gostaria de ver?

6.2 Resultados

Após as apresentações e o questionário estarem realizados, nesta secção procede-se à análise dos resultados por forma a avaliar e retirar as devidas conclusões sobre a solução, quer do ponto de vista funcional, quer do conceito em si.

A primeira pergunta realizada, com o único intuito de caracterizar o público-alvo, foi o género dos participantes. Os resultados apresentados no Gráfico 1 mostram uma maioria de participantes de género masculino, com uma minoria de género feminino. Um dos participantes não quis partilhar o seu género.

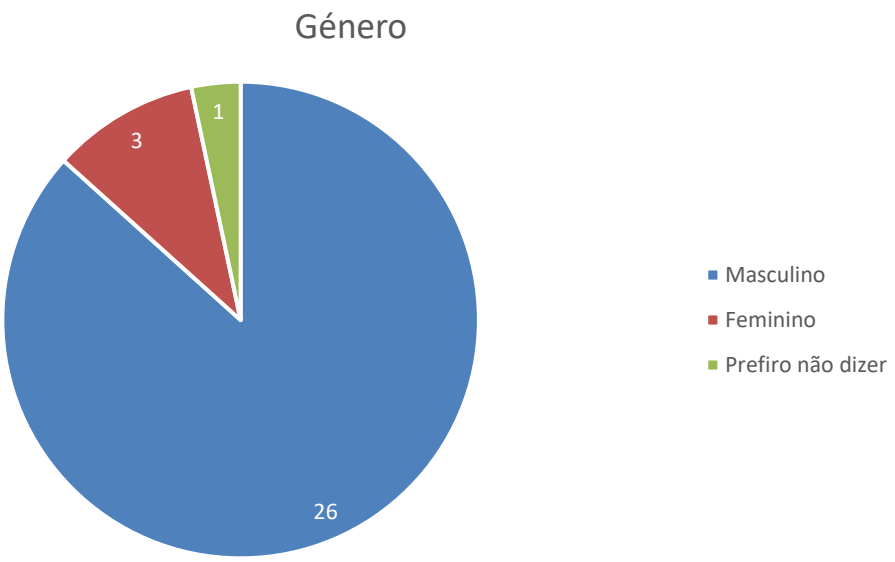


Gráfico 1 Distribuição do público-alvo por género

Já no Gráfico 2, onde é apresentada a faixa etária dos participantes, é possível observar uma tendência de participantes com idades entre os 18 e os 24 anos. Houve ainda participações de pessoas com idades entre os 25 e os 44 anos.

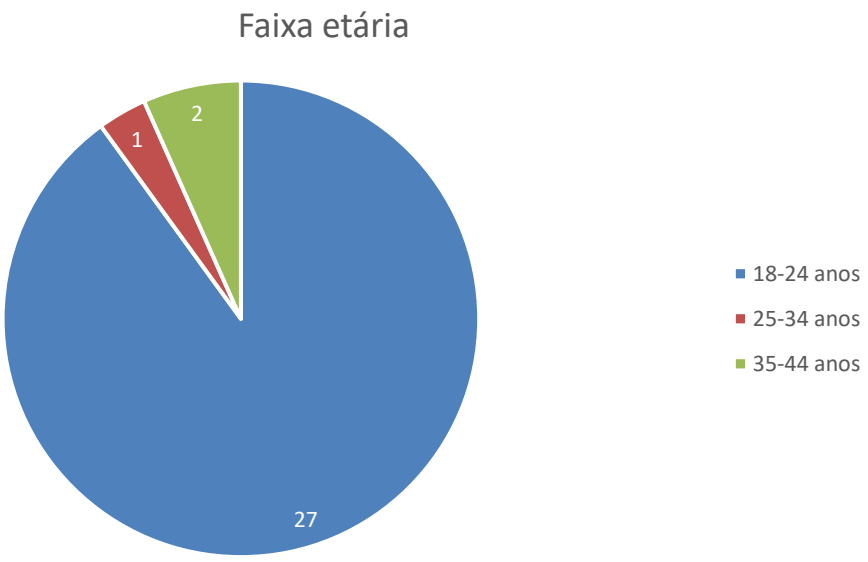


Gráfico 2 Distribuição do público-alvo por faixa etária

Já no Gráfico 3, onde são apresentadas as habilitações literárias dos participantes, é possível observar que a maioria dos participantes possui uma licenciatura. Em menor número, estão participantes com o ensino secundário e com mestrado.

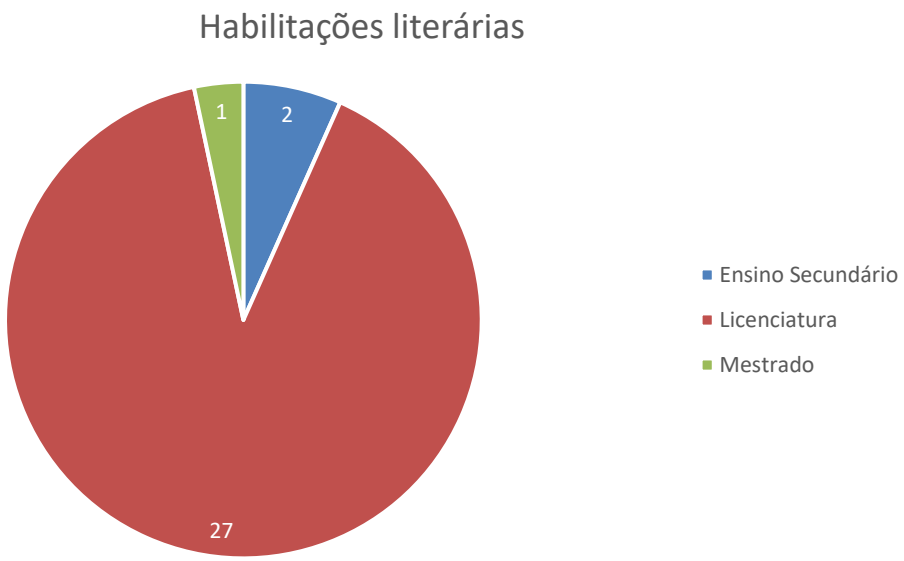


Gráfico 3 Distribuição do público-alvo por habilitações literárias

No Gráfico 4, é apresentado o número de participantes que assistiram à demonstração da parede interativa controlada por voz. Isto é um fator importante, uma vez que condiciona as perguntas relativas ao funcionamento da solução. Ainda assim, a maioria dos participantes (83%) tiveram a oportunidade de assistir à demonstração, contra 17% que não assistiram.

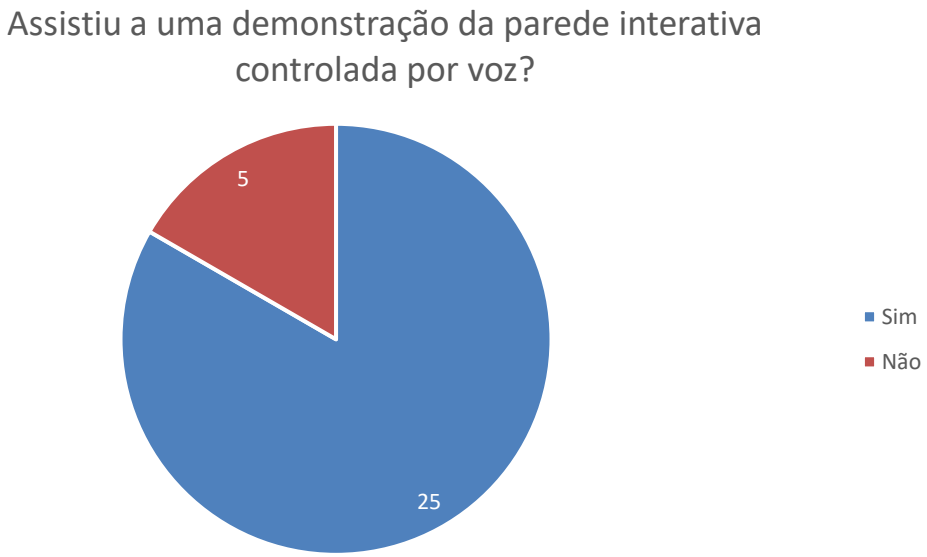


Gráfico 4 Distribuição dos participantes quanto à pergunta "Assistiu a uma demonstração da parede interativa controlada por voz?"

No Gráfico 5, onde se pergunta se os participantes acham que uma solução da natureza da do presente estudo é interessante e inovadora, 93% dos participantes responderam sim, contra apenas 7% que responderam que não.

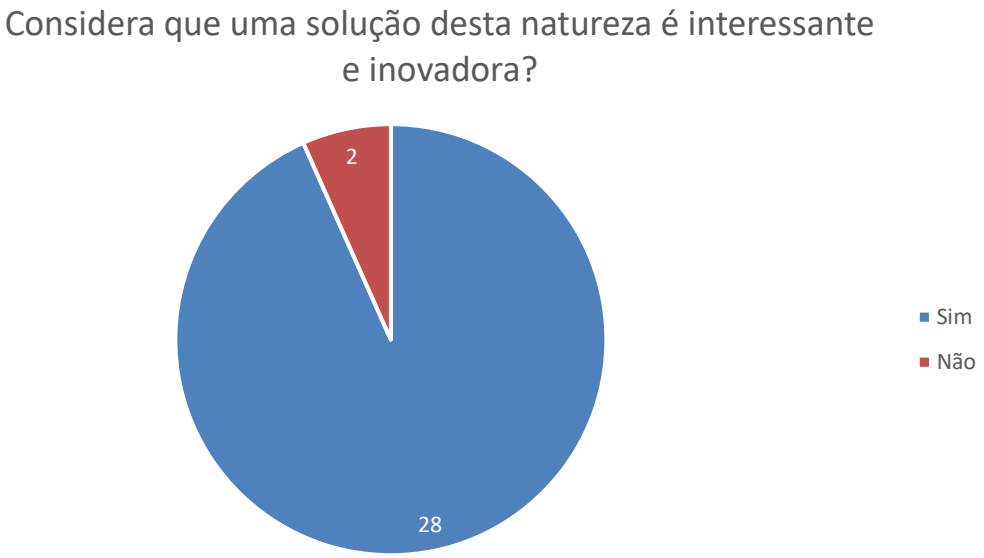


Gráfico 5 Distribuição dos participantes quanto à pergunta "Considera que uma solução desta natureza é interessante e inovadora?"

No Gráfico 6, onde se pergunta se os participantes tiveram alguma experiência em interação por voz, 83% dos participantes responderam sim, contra 17% que responderam que não.

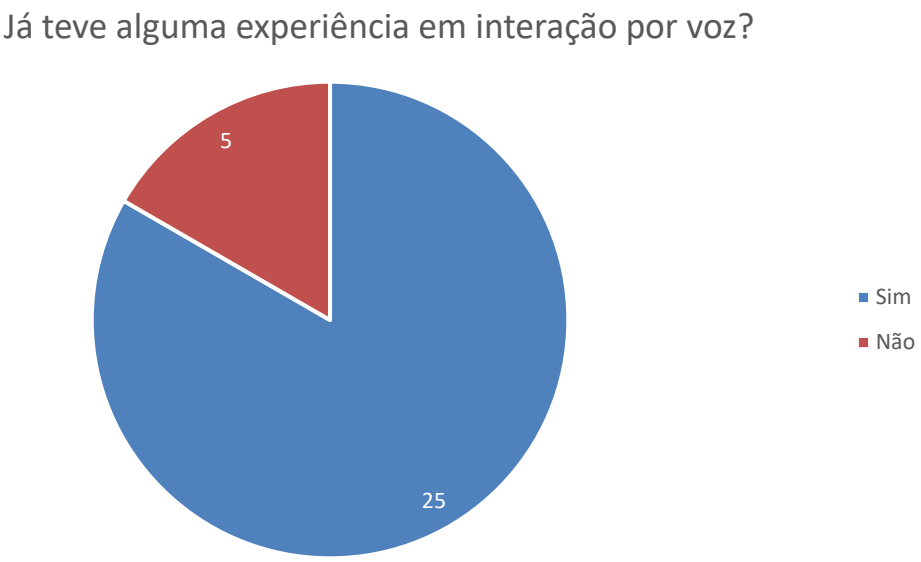


Gráfico 6 Distribuição dos participantes quanto à pergunta "Já teve alguma experiência em interação por voz?"

No Gráfico 7, onde se pergunta se os participantes consideram interessantes os sistemas de interação por voz, 93% dos participantes responderam sim, contra apenas 7% que responderam que não.

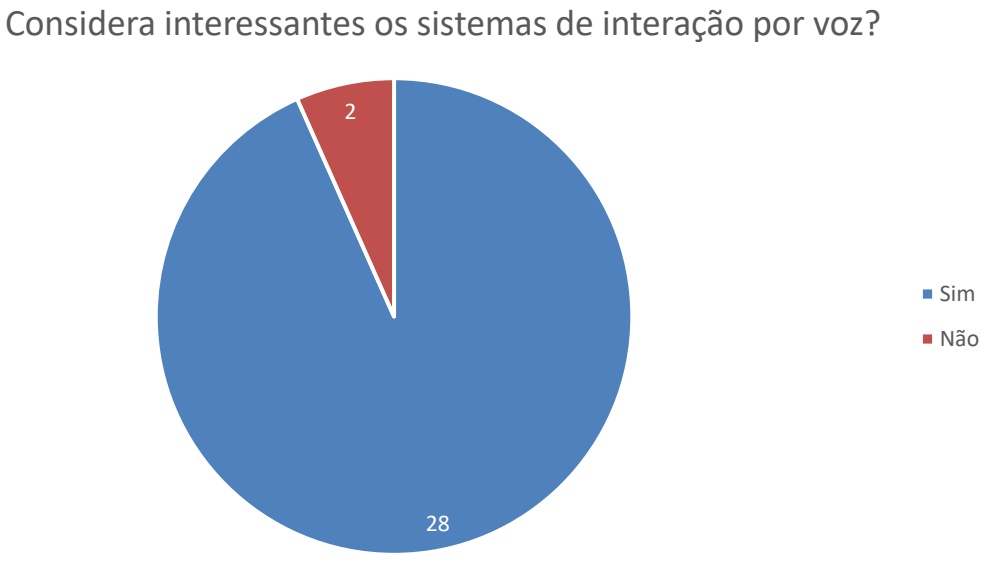


Gráfico 7 Distribuição dos participantes quanto à pergunta "Considera interessantes os sistemas de interação por voz?"

A partir deste ponto, apenas os participantes que assistiram à demonstração da parede interativa controlada por voz responderam às seguintes perguntas acerca da funcionalidade da solução.

No Gráfico 8, onde se pergunta se a interface é simples e intuitiva para novos utilizadores, a maior parte dos participantes (48%) diz concordar. Em percentagens iguais (24%) estão os participantes que dizem concordar totalmente ou nem concordar, nem discordar. Apenas um participante afirmou discordar da pergunta.

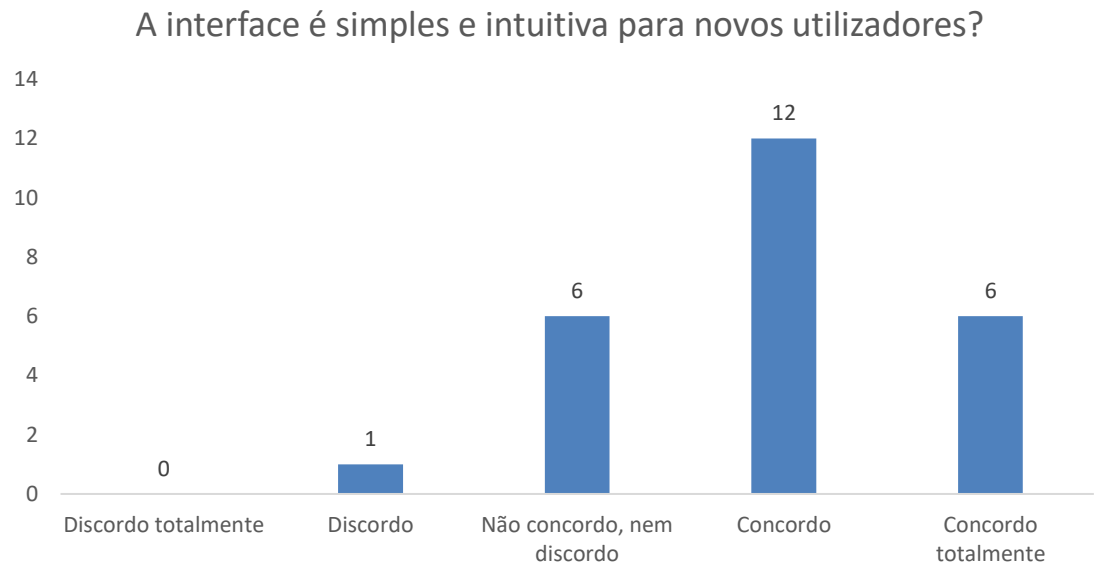


Gráfico 8 Distribuição dos participantes quanto à pergunta "A interface é simples e intuitiva para novos utilizadores?"

No Gráfico 9, onde se pergunta se os comandos de voz da solução são difíceis de entender, a maioria dos participantes (72%) respondeu que discorda ou discorda totalmente. Em percentagens menores, estão os participantes que responderam às outras possibilidades.

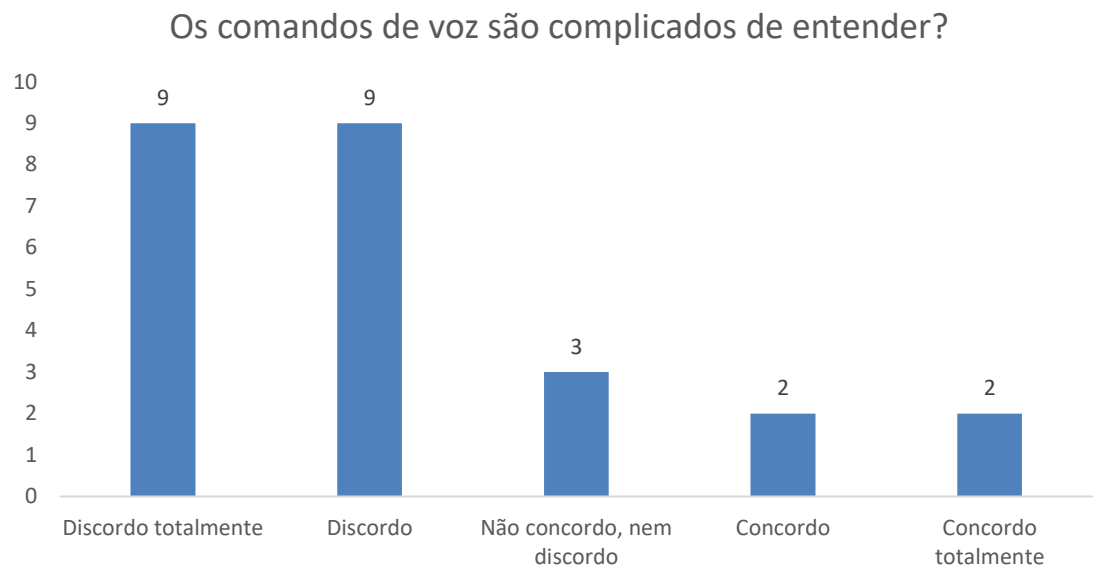


Gráfico 9 Distribuição dos participantes quanto à pergunta “Os comandos de voz são complicados de entender?”

No Gráfico 10, onde se pergunta se a solução compreendeu corretamente os comandos, os resultados já são mais variados. A maioria (88%) encontra-se nas posições intermédias da avaliação. Um participante disse discordar totalmente e dois concordar totalmente.

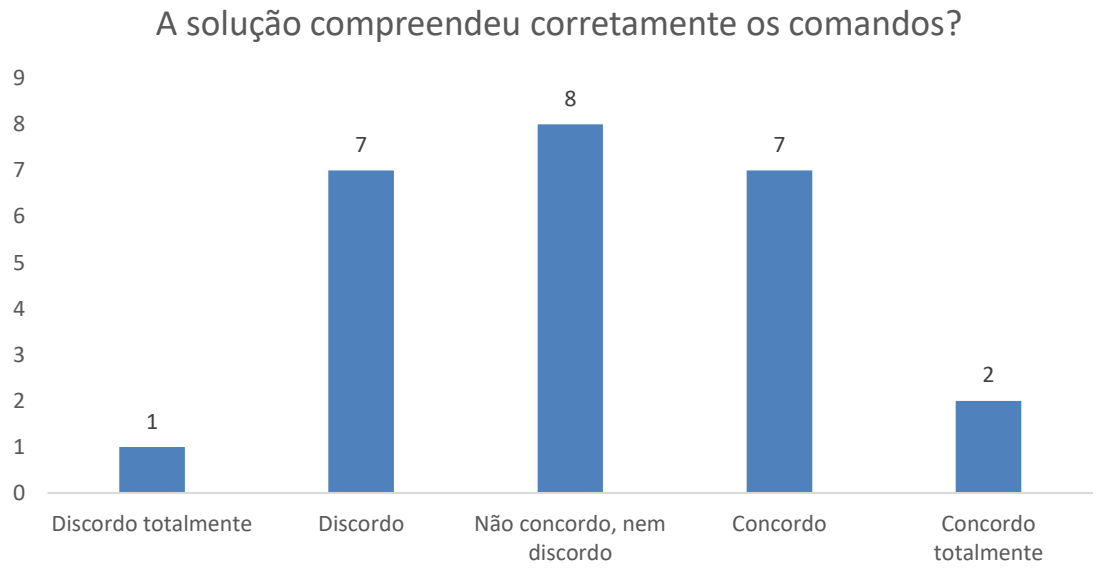


Gráfico 10 Distribuição dos participantes quanto à pergunta "A solução compreendeu corretamente os comandos?"

No Gráfico 11, onde se pergunta se as animações dos objetos geram efeitos interessantes, a maioria (56%) diz concordar ou tem uma posição neutra. Ainda com uma percentagem relevante (20%) estão os participantes que concordam totalmente. Apenas quatro participantes afirmam discordar da pergunta.

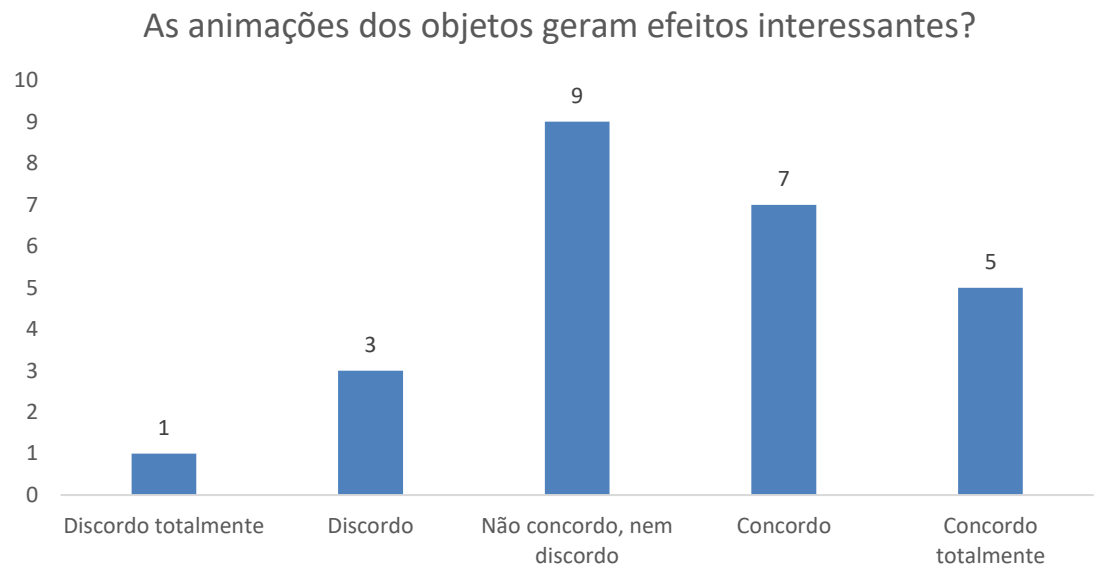


Gráfico 11 Distribuição dos participantes quanto à pergunta "As animações dos objetos geram efeitos interessantes?"

No Gráfico 12, onde se pergunta se o número de edições e animações é suficiente, a maior parte dos participantes (40%) diz não concordar, nem discordar. Existe uma percentagem significativa (28%) de participantes que afirma discordar da pergunta. Também significativa é a percentagem (20%) dos participantes que afirma concordar. Apenas três participantes se encontram nas posições extremas da votação.

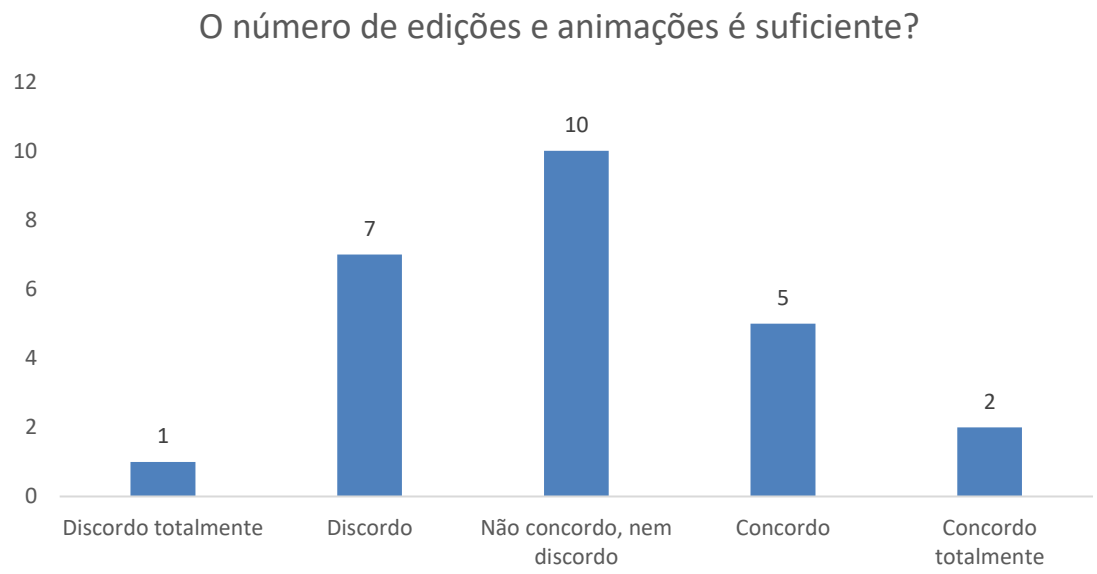


Gráfico 12 Distribuição dos participantes quanto à pergunta "O número de edições e animações é suficiente?"

No Gráfico 13, onde se pergunta se o número de comandos de voz é suficiente, existem posições mais díspares. Ainda que a maior parte (36%) diz não concordar, nem discordar, existe uma percentagem significativa (32%) que afirma discordar que o número de comandos de voz é suficiente. Ainda significativa é a percentagem (20%) dos participantes que afirma concordar. Apenas três participantes se encontram nas posições extremas da votação.

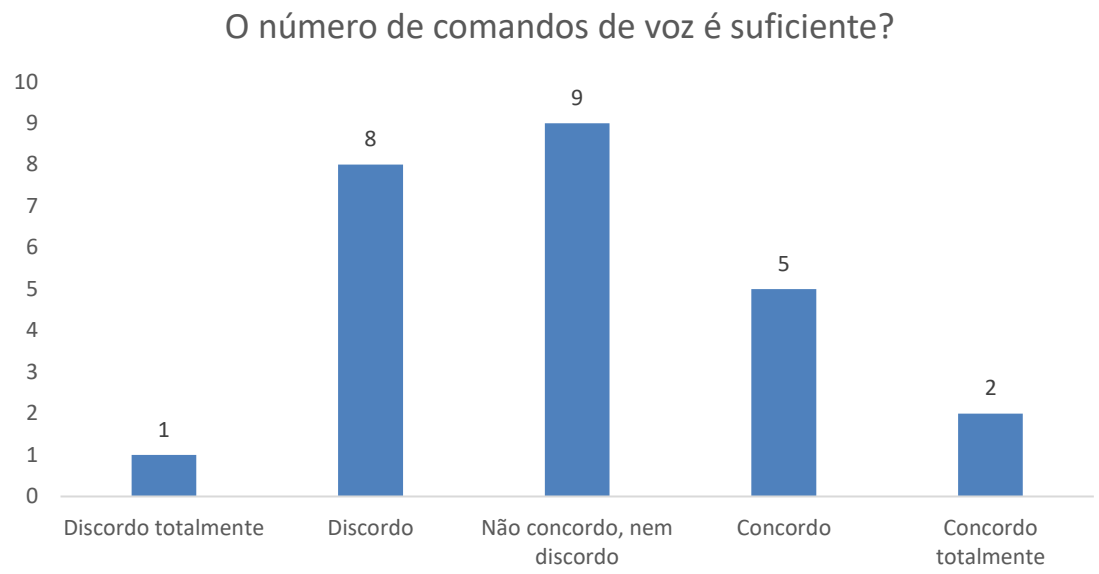


Gráfico 13 Distribuição dos participantes quanto à pergunta "O número de comandos de voz é suficiente?"

No Gráfico 14, onde se pede para classificar o potencial da solução apresentada, a maioria dos participantes (72%) afirma que a solução tem potencial ou muito potencial. Um número ainda significativo (20%), diz que a solução tem muito potencial ou um potencial razoável. Apenas dois participantes afirmam que a solução tem pouco potencial.

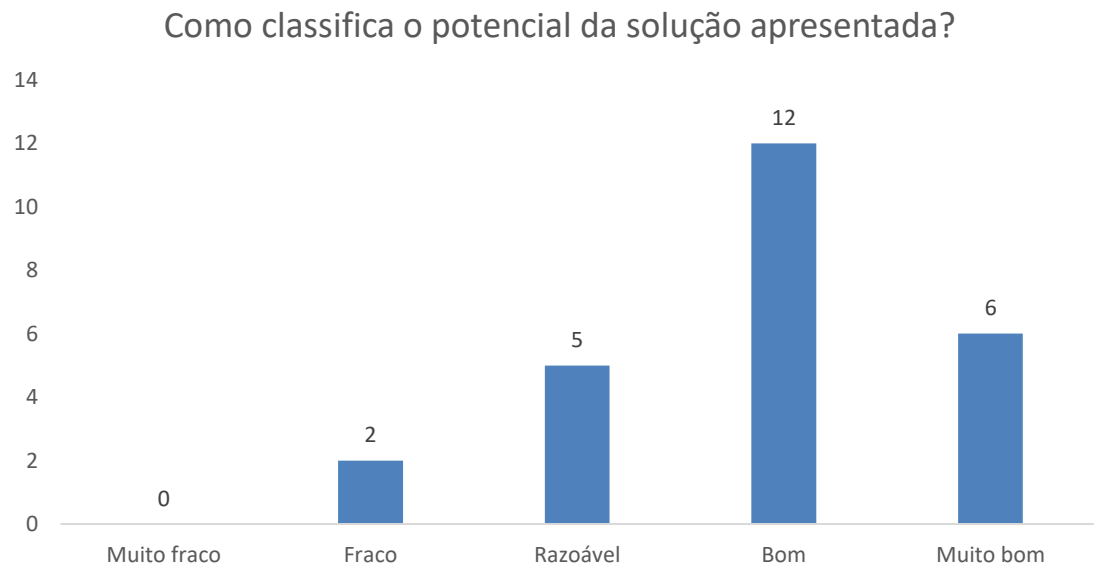


Gráfico 14 Distribuição dos participantes quanto à pergunta "Como classifica o potencial da solução apresentada?"

Para as restantes questões realizadas, os resultados permitiram não só capturar alguns defeitos não detetados durante o desenvolvimento, mas principalmente recolher informação e ideias valiosas para o futuro da solução.

No que toca aos defeitos encontrados na utilização da solução, o principal ponto apontado foi o reconhecimento dos comandos de voz. Esta questão deveu-se em grande parte à qualidade do microfone utilizado. Utilizando equipamento de melhor qualidade e mais adaptado ao ambiente, este problema é praticamente mitigado.

Quanto às sugestões para o futuro, foram indicadas uma série de melhorias e/ou novas funcionalidades. Entre elas estão:

- Aplicação de máscaras de cores sobre os objetos;
- Utilização de rastreamento ocular (*eye tracking*) para selecionar e mover objetos;
- Adição de novas animações para os objetos;
- Adição de uma etiqueta com o nome dos objetos no *canvas*;
- Assistência para pessoas invisuais;
- Possibilidade de detetar um comando de voz válido dentro de uma frase maior;
- Possibilidade de indicar múltiplos comandos de uma só vez;
- Existência de uma grelha (*grid*) que permita ao utilizador escolher a posição dos objetos no *canvas*;
- Possibilidade de alterar o ponto de ancoragem de um objeto;
- Possibilidade de mover objetos com comandos de voz utilizando a grelha ou valores fixos.

7 Conclusão

Com o desenvolvimento desta solução, foi possível levantar tecnologias que permitisse o desenvolvimento de uma parede interativa controlada por comandos de voz. Ao mesmo tempo, foi possível avaliar a viabilidade deste tipo de soluções, não só do ponto de vista técnico e tecnológico, mas também do ponto de vista de mercado e de potenciais públicos-alvo.

Quando se planeou esta solução, era já sabido, à partida, da quantidade de possibilidades e alternativas para alcançar os mesmos objetivos. Ainda assim, com pequenas exceções, foi possível atingir os objetivos pretendidos e provar a viabilidade de uma solução desta natureza.

O reconhecimento de voz atingiu já uma maturidade tal que é possível praticamente utilizá-la em qualquer dispositivo sem qualquer esforço ou complexidade. Nos dias de hoje, dada a grande capacidade dos dispositivos eletrónicos, incluindo os móveis, basta uma simples ligação à internet para realizar uma qualquer ação utilizando apenas comandos de voz.

Uma vez que a solução desenvolvida foi realizada totalmente de raiz, algumas funcionalidades não atingiram a maturidade que se pretendia dadas as limitações temporais impostas. Não há dúvidas que, no entanto, a fundação está montada e novas funcionalidades, quer da componente visual, isto é, o *canvas*, quer da componente do reconhecimento de voz podem ser implementadas sem problemas. A isto, ainda se pode incluir outras formas de interação, como o reconhecimento por gestos.

7.1 Objetivos concretizados

De uma forma geral, tanto os objetivos, como os casos de uso foram cumpridos com êxito. No entanto, devido a limitações temporais e/ou técnicas, e dependendo da avaliação efetuada, nem todas as tarefas propostas foram cumpridas na totalidade ou, pelo menos, no nível que originalmente foram planeadas.

7.1.1 Objetivos

O primeiro objetivo, analisar soluções existentes, foi cumprido na sua totalidade. Ainda que existam poucas soluções no mercado com as características ideais para a comparação com a solução em estudo, foi possível realizar uma análise comparativa.

O segundo objetivo, identificar tecnologias com potencial para o desenvolvimento, foi também cumprido na sua totalidade. Foram analisados dois tipos de tecnologias, *web* e híbrida, analisando as suas características por forma a facilitar a decisão.

O terceiro objetivo, validar aspetos técnicos e funcionais para ajudar a resolver o problema, foi também cumprido na sua totalidade. Neste passo, foi feita uma junção da análise anterior com tentativas de implementação, por forma a avaliar a viabilidade das tecnologias.

O quarto objetivo, desenvolver e implementar a solução protótipo, foi também cumprido na sua totalidade. Escolhida a tecnologia e tendo por base os requisitos e casos de uso, foi implementada a solução protótipo apresentada no presente estudo.

O quinto objetivo, testar a aplicação, foi também cumprido na sua totalidade. Após a implementação da solução, foram realizados testes funcionais e de usabilidade. Os testes funcionais foram realizados ao longo do desenvolvimento, por forma a garantir o bom funcionamento das funcionalidades implementadas. Já os testes de usabilidade foram realizados no fim do desenvolvimento, com um grupo de pessoas de forma a avaliar a qualidade da solução.

E o sexto objetivo, avaliar alternativas de interação complementares, foi o objetivo com uma taxa de conclusão mais baixa. Ainda que a interação por gestos tenha sido analisada de um ponto de vista teórico e técnico, nada foi realizado no sentido de uma futura implementação.

7.1.2 Casos de uso

O primeiro caso de uso, adicionar objeto, foi implementado na sua totalidade. Existe, no entanto, a possibilidade de adicionar novos tipo de objetos no futuro.

O segundo caso de uso, editar objetivo, foi cumprido a 80%. Ainda que a maioria das edições tenham sido implementadas, existem algumas, como é o caso da troca de cor de um objeto, que não ficaram concluídos nesta versão.

O terceiro caso de uso, animar objeto, foi concluído a 90%. Ainda que todas as animações propostas tenham sido implementadas, existe a possibilidade de implementar outras no futuro. Existem também alguns problemas com determinadas animações que não foram corrigidos em tempo útil.

O quarto caso de uso, remover objeto, foi cumprido na sua totalidade. Todos os cenários testados provaram que a funcionalidade está completa.

O quinto caso de uso, criar projeto, foi cumprido na sua totalidade. Dada a sua simples implementação, foi possível garantir o bom funcionamento deste caso de uso em todos os cenários testados.

O sexto caso de uso, guardar projeto, foi cumprido na sua totalidade. Todos as informações presentes no *canvas*, incluindo as animações, são guardas com êxito.

O sétimo caso de uso, abrir projeto, foi cumprido a 75%. Ainda que o projeto seja importado corretamente e na totalidade, as animações carregadas em memória não são reiniciadas no seu estado anterior.

E o oitavo caso de caso, exportar projeto, foi também cumprido na sua totalidade.

7.2 Trabalho futuro

Existem vários períodos temporais para o qual se pode avaliar o futuro desta solução: a curto, a médio e a longo prazo.

A curto prazo, é expectável melhorias às funcionalidades existentes, assim como correções detetadas nos testes de usabilidade. Entre outros possíveis problemas, foram detetados os seguintes:

- Correção das animações ao importar um projeto previamente gravado;
- A rotação dos objetos não é contínua (ainda que seja um comportamento esperado, seria interessante corrigir);
- A animação de vários objetos no mesmo grupo, causa alguns problemas visuais quando selecionados.

A médio prazo, é expectável a implementação de novas funcionalidades, quer as planeadas, mas não implementadas, quer sugeridas nos questionários de usabilidade. Em concreto, destacam-se as seguintes:

- Adição de mais animações;
- Alteração da velocidade e suavização das animações;
- Adição de mais edições, como alteração de cores e nomes, entre outras;
- Adição de etiquetas nos objetos presentes no canvas;
- Adição de mais comandos de voz;
- Melhorias no reconhecimento de comandos de voz.

E por fim, a longo prazo, a implementação de outras formas de interação, como os já mencionados gestos, mas também outros como *eye tracking*.

7.3 Apreciação final

Como todos os projetos, nem tudo o que foi planeado correu de acordo com o esperado. Ainda assim, foi possível provar a viabilidade de uma solução desta natureza e quais os desafios na sua implementação. Foi um projeto muito enriquecedor e que, sem dúvida, contribuiu para o estudo e evolução, não só do conceito de parede interativa, como das possibilidades de expansão ainda possíveis para este tipo de projeções.

Referências

(2022). Retrieved from Three.js: https://threejs.org/

(2022). Retrieved from PixiJS: https://pixijs.com/

Amazon. (2022). *Amazon Alexa*. Retrieved from Amazon Developer: https://developer.amazon.com/alexa

Amazon. (2022). *Amazon Transcribe*. Retrieved from Amazon Web Services: https://aws.amazon.com/transcribe/

Amorim, P. (26 de julho de 2018). *PROJEÇÃO INTERATIVA: INÚMERAS POSSIBILIDADES PARA O SEU EVENTO*. Obtido de https://www.r1audiovisual.com.br/blog/projecao-interativa-inumeras-possibilidades-para-o-seu-evento

Apple. (2022). *Siri*. Retrieved from Apple: https://www.apple.com/siri/

Cianci, L. (14 de maio de 2021). *Flutter desktop vs Electron: Why to choose Flutter for your next desktop app*. Obtido de https://blog.codemagic.io/flutter-vs-electron/

CineMassive. (2022). *What Type of Video Wall Display Fits Your Needs?* Obtido de CineMassive: https://www.cinemassive.com/compare-video-wall-types/

Dix, A. (2022). *Human-Computer Interaction (HCI)*. Retrieved from Interaction Design Foundation: https://www.interaction-design.org/literature/topics/human-computer-interaction

Dolbey. (2022). *Five Voice Recognition Technology Trends & Applications*. Retrieved from https://dolbeyspeech.com/blog/5-speech-voice-recognition-trends-applications/

Edwards, P. (23 de outubro de 2020). *The technology that's replacing the green screen*. Obtido de https://www.vox.com/21529002/green-screen-mandalorian

Fabric.js. (2022). Retrieved from Fabric.js: http://fabricjs.com/

Google. (2022). *Angular*. Retrieved from https://angular.io/

Google. (2022). *Flutter - Build apps for any screen*. Retrieved from https://flutter.dev/

Google. (2022). *Google Assistant*. Retrieved from Google: https://assistant.google.com/

Google. (2022). *Text-to-Speech*. Retrieved from Google Cloud: https://cloud.google.com/text-to-speech/

H., G. (11 de outubro de 2021). *macOS Performance Comparison: Flutter Desktop vs. Electron*. Obtido de https://getstream.io/blog/flutter-desktop-vs-electron/

Hahn, M. (4 de janeiro de 2019). *Microphone Types: How to Choose the Right Mic for Your Sound*. Obtido de LANDR: https://blog.landr.com/microphone-types/

Koen, P. (2014). *Front End Innovation - FEI*. Obtido de http://www.frontendinnovation.com/fei

Koen, P. A., Ajamian, G. M., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., . . . Seibert, R. (s.d.). Fuzzy Front End: Effective Methods, Tools, and Techniques. Em *The PDMA ToolBook for New Product Development* (pp. 5-35).

Levine, M. (6 de março de 2022). *Different types of microphones and when to use them*. Obtido de Popular Science: https://www.popsci.com/reviews/types-of-microphones/

Lumo Interactive. (2022). *LUMOplay*. Retrieved from https://www.lumoplay.com/

Meireles, A. (16 de dezembro de 2021). *Aquário Vasco da Gama abre sala interativa para atrair mais crianças*. Obtido de Diário de Notícias: https://www.dn.pt/local/aquario-vasco-da-gama-abre-sala-interativa-para-atrair-mais-criancas-14411996.html

Meta Open Source. (2022). *A JavaScript library for building user interfaces*. Obtido de React: https://reactjs.org/

Meta Open Source. (2022). *Create React App*. Retrieved from Create React App: https://create-react-app.dev/

Microsoft. (2022). *Introduction to Microsoft Edge WebView2*. Retrieved from https://docs.microsoft.com/en-us/microsoft-edge/webview2/

Microsoft. (2022). *Speech to text*. Retrieved from Microsoft Azure: https://azure.microsoft.com/products/cognitive-services/speech-to-text/

Nugent, J. (27 de janeiro de 2021). *The 3 main types of microphones (with subtypes) and their best uses*. Obtido de Higher Hz: https://higherhz.com/microphone-types/

OpenJS Foundation. (2022). *Electron*. Retrieved from https://www.electronjs.org/

Osterwalder, A., & Pigneur, Y. (2010). *Business Model Generation.*

Podfeet. (20 de agosto de 2021). *Podfeet PodcastsThe Day the Internet Lost Its Mind about 1Password Becoming an Electron App*. Obtido de Podfeet Podcasts: https://www.podfeet.com/blog/2021/08/1password-electron/

Quality-One. (2022). *Quality Function Deployment (QFD)*. Retrieved from https://quality-one.com/qfd/

Saaty, T. L. (2008). *Decision making with the analytic hierarchy process.*

Scardina, J. (janeiro de 2018). *What is voice recognition (speaker recognition)?* Obtido de https://searchcustomerexperience.techtarget.com/definition/voice-recognition-speaker-recognition

Sonix. (2022). *A short history of speech recognition*. Retrieved from Sonix: https://sonix.ai/history-of-speech-recognition

Swami, U. (2022). *What Are Video Wall Technology Types? LCD Video Wall? LED & Projection Video Walls?* Retrieved from https://headendinfo.com/video-wall-technology/

Wallace, E. (8 de junho de 2017). *WebAssembly cut Figma's load time by 3x*. Obtido de https://www.figma.com/blog/webassembly-cut-figmas-load-time-by-3x/