INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Assistente Digital de Apoio ao Ensino Musical Orientado a Indivíduos com Deficiência Visual

PEDRO RIBEIRO TINOCO RODRIGUES
Outubro de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Assistente Digital de Apoio ao Ensino Musical
Orientado a Indivíduos com Deficiência Visual

Pedro Ribeiro Tinoco Rodrigues

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Dr. Isabel Praça

Júri:
Presidente:

Vogais:

Porto, 15 de Outubro de 2020

"Music gives a soul to the universe,
wings to the mind,
flight to the imagination,
and life to everything."

Platão

Dedicatória

A todos aqueles que me acompanharam e viabilizaram que esta fase fosse atingível.

Resumo

Apesar da evolução tecnológica se encontrar aliada a vantagens imensuráveis e a constantes reformas da realidade atual, seja pelo desenvolvimento de soluções inovadoras, como pelo aparecimento de inúmeros dispositivos e aplicações, esta nem sempre é aplicada a toda a população. A exclusão digital e social das minorias ainda se faz sentir e, como tal, o conceito de *digital divide* ainda é uma realidade. Na área da aprendizagem musical, a dificuldade existente em indivíduos pertencentes a algumas minorias, mais especificamente, indivíduos invisuais é acrescida. Assim, parte-se desta premissa para a realização de um assistente digital que agilize e auxilie o processo de aprendizagem musical por parte de indivíduos invisuais.

Está presente neste documento uma introdução ao projeto, contextualizando-o, identificando o problema existente e os objetivos que a atual solução pretende endereçar. É feita uma contextualização relativamente a soluções de auxilio musical previamente existentes, dividindo-se entre as soluções que auxiliam a abordagem de ensino tradicional e as soluções intuitivas que utilizam a tecnologia visando descobrir novas formas de auxiliar o processo de aprendizagem. De seguida, introduz-se o conceito de assistentes digitais e explicitam-se os módulos necessários ao seu funcionamento, assim como algumas tecnologias que auxiliam o desenvolvimento de cada um. Posteriormente, é elaborada a análise de valor que determina a validade e a necessidade da existência deste projeto. Posto isto, é analisada e modelada uma solução que integra o conceito de assistentes digitais com algoritmos de análise de som, sendo posteriormente descrita a sua implementação. Por último, de modo a avaliar o potencial funcional, a satisfação e o grau de usabilidade do projeto, apresentam-se métodos de avaliação baseados em testes de *software*, testes de hipóteses e inquéritos.

Palavras-chave: *Digital Divide*, Ensino Musical, Assistente Digital, Usabilidade, Análise de Som

Abstract

The incessant technological evolution is combined with a wide range of solutions that have emerged in order to overcome some existing difficulties. However, this is not the reality for the entire population. The digital and social exclusion of minorities is still felt and, as such, the concept of digital divide is still a reality. In the area of musical learning, the difficulty existing in individuals belonging to some minorities, more specifically, blind individuals, is increased. Thus, this premise is based on the realization of a digital assistant that streamlines and helps the process of musical learning by blind individuals.

An introduction to the project is present in this document, contextualizing it, identifying the existing problem and the objectives that the current solution intends to address. A contextualization is made in what relates to some previously existing musical support solutions, dividing itself between the solutions that help the traditional teaching approach and the intuitive solutions that use technology in order to discover new ways to assist the learning process. Then, the concept of digital assistants is introduced and the needful modules for its operation are explained, as well as some technologies that help the development of each one. Subsequently, a value analysis is carried out that determines the validity and necessity of this project existence. That said, a solution that integrates the concept of digital assistants with sound analysis algorithms is analyzed and modeled, and its implementation is described later. Finally, in order to assess the functional potential, satisfaction and degree of usability of the project, evaluation methods based on software tests and surveys are presented.

Keywords: Digital Divide, Music Education, Digital Assistant, Usability, Sound Analysis

Agradecimentos

Esta tese não teria sido possível sem a cooperação daqueles que, direta e indiretamente estiveram envolvidos ao longo de todo o processo que me traz a este projeto. Expresso os meus sinceros agradecimentos a todos.

Em primeiro lugar, quero agradecer especialmente à minha família, aos que por cá estão e aos que estiveram, à minha namorada e aos meus amigos. Estes foram e continuam a ser o meu maior apoio e influência. Diz-se que somos o que vivemos e parte das pessoas com quem nos relacionamos e, por isto, devo-lhes tudo.

Quero direcionar um agradecimento à professora Isabel Praça por toda a paciência e serenidade que me foi transmitindo tanto no desenvolvimento deste projeto, como ao longo do curso nas mais diversas áreas em que nos fomos encontrando.

Ao Instituto Superior de Engenharia do Porto e a todos os docentes que me foram acompanhando no decorrer da minha formação superior, segue também um agradecimento sentido.

"If I have seen further it is by standing on the shoulders of giants."

Isaac Newton

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Lista de Acrónimos

AAA	Arrange-Act-Assert.
AHP	Analytic Hierarchy Process.
API	Application Programming Interface.
ASR	Automatic Speech Recognition.
ATiA	Assistive Technology Industry Association.
BDD	Behavior Driven Development.
CSAT	Customer Satisfaction Score.
DBaaS	Database as a Service.
FFE	Fuzzy Front End.
FIFO	First In First Out.
MIDI	Musical Instrument Digital Interface.
NCD	New Concept Development.
NLU	Natural Language Understanding.
NPD	New Product Development.
ODM	Object Data Model.
PaaS	Platform as a Service.
RC	Razão de Consistência.
REST	Representational State Transfer.
RGPD	Regulamento Geral sobre a Proteção de Dados.
RMS	Root Mean Square.
RNIB	Royal National Institute of Blind People.
SoC	Separation Of Concerns.
SSD	System Sequence Diagram.
UML	Unified Modeling Language.
USB	Universal Serial Bus.
W3C	World Wide Web Consortium.

Capítulo 1

Introdução

Dada a constante evolução digital, várias soluções aliadas a esta têm surgido de forma a colmatar dificuldades outrora sentidas. A consequente evolução tecnológica tem facilitado a vida das pessoas, fornecendo conveniências sem precedentes tanto a nível comunicativo, educacional, produtivo e pessoal [1]. No entanto, apesar de todas as vantagens inerentes à sua utilização, existe uma quantidade elevada da população que ainda se encontra à margem desta evolução. É com base nesta premissa que se apresenta no presente projeto um produto com valor de inclusão social que atua na área educacional, visando melhorar o processo do ensino musical orientado a indivíduos com deficiência visual.

1.1 Contexto

Os efeitos advindos da interação musical são infindáveis, pelo que a sua utilização explicita várias melhorias na vida das pessoas, independentemente do seu género, idade ou condição de saúde. Assim, considera-se que a música desempenha um papel fundamental no desenvolvimento do ser humano, afetando aspetos fisiológicos como a alteração de pressão arterial, batimentos cardíacos e respiração, assim como aspetos emocionais, envolvendo o humor e até o estado de espírito [2].

Na sequência da realização de vários estudos, a utilização de música como área terapêutica tem vindo a revelar-se positiva, confirmando um efeito relevante que potencia tratamentos relacionados com depressões e ansiedade [3]. Complementando o valor da utilização musical, existem estudos que comprovam os resultados benéficos da sua utilização em contextos de medicina física, indicando que existem evidências científicas de que esta ameniza o desconforto e a dificuldade associada à realização dos exercícios terapêuticos, ajudando e garantindo uma participação mais consistente [4]. A par com a medicina física, existem também evidências relacionadas com a reabilitação física e mental, explicitando que a música poderá servir como estimulante, apoiando e direcionando algumas das funções mentais a serem reabilitadas [4][5].

Uma das forma de contacto com a música é a sua própria aprendizagem, no entanto, apesar do processo de aprendizagem musical ser enriquecedor para o desenvolvimento do ser humano, é também um processo complexo. Com a introdução e o desenvolvimento de novas tecnologias, o paradigma de aprendizagem musical tem vindo a sofrer alterações. Desde a introdução de instrumentos virtuais em detrimento dos instrumentos tradicionais (ou mecânicos) [6] à disponibilização constante de recursos didáticos por meios digitais, que se denota uma mudança radical no decorrer deste processo. O dinamismo dos recursos publicados *online*, acessíveis através de qualquer parte do mundo e a disponibilização constante dos mesmos nas várias plataformas digitais, recorrendo até a recursos áudio-visuais enfatiza

a globalização que se tem sentido e que tanto tem alterado o paradigma de aprendizagem musical.

O conjunto de vantagens das soluções baseadas nas plataformas digitais, aliadas ao aumento constante de indivíduos que acedem à *Internet* [7][8], conforme se pode analisar pela figura 1.1, fazem transparecer uma realidade quase utópica no que diz respeito ao acesso à informação.

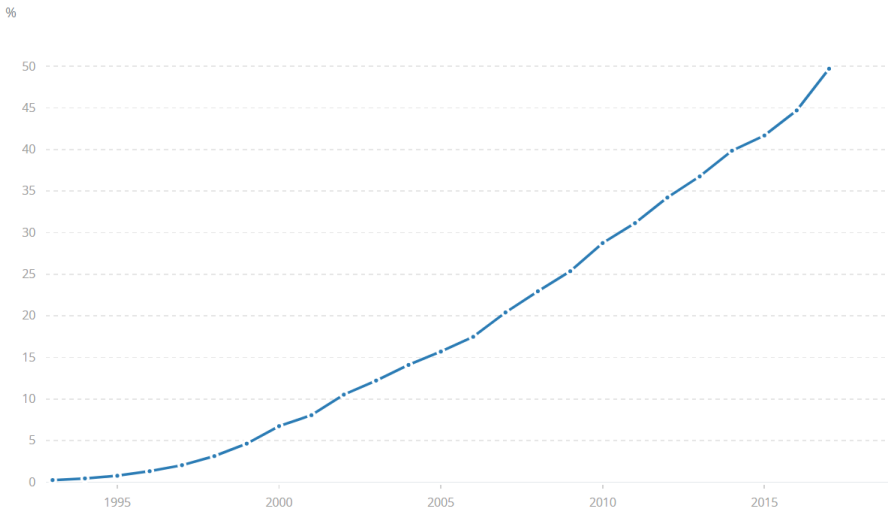

Figura 1.1: Evolução percentual de indivíduos que acedem à *Internet* [7]

Embora a maioria dos indivíduos que acedem à *Internet* usufruam de grande parte do seu potencial, existe ainda um segmento da população que é marginalizado constantemente neste processo. O desenvolvimento de plataformas que não consideram a facilidade de utilização por parte de pessoas com deficiências, pode levar a que estas sejam excluídas. Este fenómeno, que reflete a existência de uma lacuna em termos de acesso e uso da tecnologia da informação e comunicação, denomina-se *Digital Divide* [9]. De forma a atenuar este contraste digital e visando dar resposta à lacuna existente, diminuindo, assim, o *Digital Divide*, várias soluções têm vindo a ser desenvolvidas. Considerando que estas soluções poderão ser concretizadas no desenvolvimento de qualquer recurso, equipamento, programa de software ou produto utilizado para manter, aumentar, ou melhorar os recursos funcionais das pessoas com deficiência, as mesmas são categorizadas, segundo a *Assistive Technology Industry Association (ATiA)* [10], como Tecnologias de Apoio, tendo estas vindo a ser alvo de evolução ao longo dos anos.

1.2 Problema

Considerando que o processo de aprendizagem musical, levado a cabo por um indivíduo sem qualquer condicionante mental ou físico, é bastante complexo e custoso, é quase intuitiva a conclusão de que para indivíduos portadores de deficiências visuais, tal é ainda mais dificultado.

A abordagem tradicional de ensino musical a indivíduos que detenham deficiências visuais é amplamente baseada no sistema *braille*, como explicado na secção 2.1, no entanto, este sistema não é considerado eficaz.

Uma vez que o ensino musical em *braille* implica o estudo de novos símbolos com novos significados, para os indivíduos com deficiência visual, aprender música significa, também, aprender o sistema musical *braille*, sendo esta uma das causas da complexidade acrescida do processo. Dito isto, a falta de conhecimento existente, tanto nos alunos como nos professores, converge numa menor utilização deste tipo de sistema, pelo que se considera que a elevada especialização necessária à sua utilização é, atualmente, a principal barreira imposta para que este sistema sirva o seu propósito de forma integra e eficaz [11].

A utilização de plataformas digitais que visam a disponibilização e consequente recolha de informação tem evoluído, sendo esta, atualmente, a abordagem preferencial de disponibilização de conteúdo. No entanto, o facto de que os recursos didáticos possam ser partilhados em qualquer tipo de formatos, desde o formato de texto ao formato áudio-visual, não implica que os indivíduos invisuais consigam tirar o mesmo partido deste potencial. Assim, denota-se a existência de dificuldade na recolha de informação por parte destes indivíduos.

Em suma, dada a existência de indivíduos que não detêm o conhecimento de leitura musical em *braille*, estando mais dispostos a aprender música do que os novos símbolos existentes na própria linguagem, aliada à falta de plataformas que façam o uso de tecnologias para facilitar a recolha de informação, surge a necessidade de desenvolver mais soluções que apoiem uma metodologia de apoio e ensino alternativa.

1.3 Objetivos

De forma a apoiar a diminuição do *Digital Divide* existente, pretende-se o desenvolvimento de uma plataforma digital que contenha valor pela sua contribuição social, integrando os indivíduos com deficiências visuais na área tecnológica. Posto isto, pretende-se que esta se categorize como uma metodologia de ensino alternativa que apoia a aprendizagem musical por parte do individuo com deficiência visual, complementando o método de ensino tradicional musical.

Assim, os seguintes objetivos mais específicos devem ser tidos em consideração:

- Investigar e analisar uma abordagem de ensino musical que não seja dependente do sentido da visão, potenciando e agilizando o ensino autónomo do instruendo, indivíduo com deficiência visual.
- Desenvolver um sistema baseado na abordagem de ensino investigada capaz de proporcionar um ensino imersivo, apelativo, eficaz e autónomo a indivíduos com deficiência visual.
- Avaliar a eficácia do sistema baseada em dados de utilização reais.

1.4 Estrutura do Documento

Este documento é composto por sete capítulos sendo estes: Introdução, Estado de Arte, Análise de Valor, Análise, *Design*, Implementação e Experimentação e Avaliação. De seguida apresenta-se uma breve descrição dos mesmos:

- Introdução - Introduz e contextualiza o projeto, sendo definido o problema e os objetivos que se pretendem atingir. Por último é indicada a estrutura do documento.

- Estado de Arte - É apresentada uma contextualização relativa às soluções digitais existentes, tanto a nível do apoio à metodologia de ensino tradicional como, ainda, soluções intuitivas para o ensino musical. Aborda-se, também, o tema de assistentes digitais, descrevendo-se o processo de funcionamento dos mesmos. Apresentam-se tecnologias que auxiliam o desenvolvimento de um assistente digital, assim como tecnologias que visam manipular som em ambientes de desenvolvimento *web*. Enumeram-se os critérios a seguir aquando da definição de *frameworks* de desenvolvimento *web*. Por último, para que se conclua a melhor forma de desenvolver um sistema de apoio ao ensino musical, compara-se o desenvolvimento deste sistema através da utilização de assistentes digitais existentes ou através do desenvolvimento de um totalmente novo com integração de um módulo de análise de som.
- Análise de Valor - É retratada a oportunidade identificada, seguida de uma análise que resulta em várias ideias de soluções que, por sua vez, são selecionadas com base num método multicritério. Finalmente, apresenta-se a proposta de valor do presente projeto, assim como o modelo de negócio proposto.
- Análise - São apresentadas e descritas as entidades do domínio em questão, seguidas de uma apresentação dos requisitos funcionais e não funcionais do sistema.
- *Design* - Concebem-se duas alternativas de *design* da solução final seguidas da avaliação e escolha justificada da que mais se adequa. Uma vez definida a solução, esta é detalhada através da exposição de artefactos específicos e adaptados a cada fase.
- Implementação - Neste capítulo explicitam-se os detalhes técnicos da implementação de todo o sistema desenhado.
- Experimentação e Avaliação - Expõe-se as hipóteses, os indicadores, as fontes de informação e o método de avaliação.
- Conclusão - Revelam-se os objetivos alcançados, seguidos pelas limitações do projeto e do trabalho futuro que o enriquecerá.

Capítulo 2

Estado de Arte

Neste capítulo são analisadas soluções de grande relevância no que diz respeito à diminuição do *Digital Divide*, mais especificamente, orientado a pessoas com deficiência visual que pretendem aprender música. Apresentam-se soluções, já existentes, de apoio ao ensino musical orientadas à abordagem tradicional e, ainda, soluções intuitivas de aprendizagem musical.

Seguidamente, introduz-se o conceito de assistente digital, descreve-se o processo responsável pelo diálogo inteligente que estes contêm e apresentam-se algumas tecnologias que auxiliam o desenvolvimento de cada módulo constituinte desse processo. Uma vez abordado o processo e as tecnologias, referem-se boas práticas de *design* de interfaces de voz.

Posto isto, referem-se algumas tecnologias de análise de som orientadas a aplicações *web*, seguido de uma listagem de critérios utilizados na definição das *frameworks* de desenvolvimento *web* candidatas.

Por último, fundamenta-se a decisão quanto à utilização de um assistente digital já existente ou ao desenvolvimento de um novo, concluindo-se o capitulo através da comparação de todas as tecnologias previamente identificadas, justificando a opção tomada.

2.1 Abordagem Tradicional para o Ensino Musical

O alfabeto *braille*, desenvolvido por Louis Braille no ano 1829, é um sistema de representação de símbolos que é, desde então, o sistema de escrita e leitura utilizado por parte de indivíduos total ou parcialmente invisuais. A par do desenvolvimento do código *braille* literário, Louis Braille, ele próprio músico e invisual, desenvolveu um sistema *braille* de notação musical que ganhou grande destaque no inicio do século XX, tendo este servido, posteriormente, como base para a criação do *Music Braille Code*, publicado por *Braille Authority of North America* [12].

A abordagem tradicional adotada para o ensino musical a indivíduos com deficiência visual é amplamente baseada no sistema *braille*, no entanto, embora existam serviços de transcrição de áudio e *braille*, que se encontram disponibilizados por organizações especializadas na produção em *braille* ou, até, através de bibliotecas externas, a disponibilidade de partituras musicais em *braille* continua a ser bastante limitada (c.f secção 1.2).

Dada a existência de soluções digitais que visam apoiar a abordagem tradicional, considera-se que esta abordagem de ensino não se encontra totalmente marginalizada da evolução tecnológica, no entanto, o processo de ensino e produção de pautas musicais em *braille* continua a ser demasiado custoso, em termos de tempo e recursos. Assim, todos os custos

envolventes são mais elevados neste sistema, podendo a impressão de uma partitura em papel, num formato que permita a leitura por músicos com deficiência visual custar até dez vezes mais do que uma impressão resultante de uma conversão de texto em *braille* [11].

Na secção seguinte, apresentam-se soluções já existentes que pretendem facilitar o ensino musical tradicional, não alterando a abordagem de ensino, ou seja, mantendo o *braille*.

2.1.1 Soluções Existentes

Entre as várias soluções digitais existentes que apoiam o ensino tradicional, soluções como *Toccata* [13], *MusiBraille* [14] e *GoodFeel* [15] destacam-se pela sua maior notoriedade e pelas funcionalidades que oferecem.

Todas as soluções previamente enumeradas providenciam um serviço completo de conversão de partituras musicais em *braille*. Para isto, incorporam funcionalidades como: reconhecimento de partituras para posterior tradução para *braille*; Integração de editor de partituras em *braille* e em partitura clássica, permitindo posterior conversão, e sistema embutido de reprodução de notas musicais. Todo o processo de transcrição de notas musicais é facilitado, aumentando, assim, a autonomia das pessoas com deficiência visual na aprendizagem e na produção musical.

Uma vez expostas as funcionalidades que estes disponibilizam, torna-se relevante indicar alguns dos pontos negativos. Assim, de uma perspectiva de *user-experience*, é perceptível o desenho pouco apelativo das interfaces destes sistemas - representados nas figuras 2.1, 2.2 e 2.3. Para além de serem sistemas de apoio pagos, muitas vezes ao ponto de não serem utilizados por falta de recursos [16], todos estes possuem limitações de software quanto à sua portabilidade, na medida em que não foram implementados visando a sua execução *cross-platform*, tendo sidos apenas desenvolvidos para sistemas operativos *Windows*, implicando, inevitavelmente, uma menor aderência do mercado.

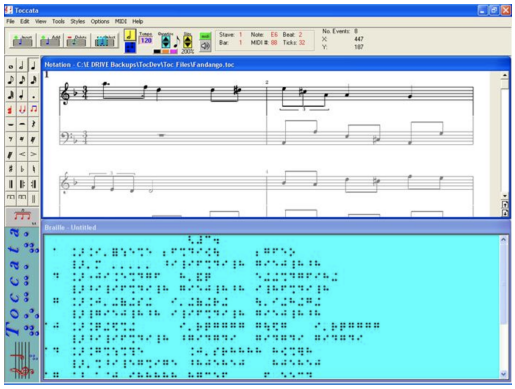

Figura 2.1: Interface gráfica do sistema digital *Toccata* [13]

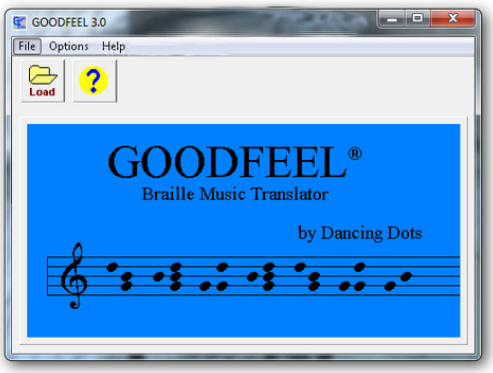

Figura 2.2: Interface gráfica do sistema digital *GoodFeel* [15]

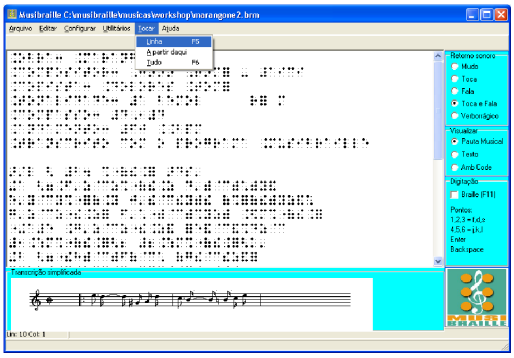

Figura 2.3: Interface gráfica do sistema digital *MusiBraille* [14]

2.2 Soluções Intuitivas para o Ensino Musical

Tanto o processo de aprendizagem musical teórico como o processo de aprendizagem de um instrumento musical específico, implicam, inevitavelmente, um investimento de tempo muito elevado. Efetuando o paralelismo entre indivíduos desprovidos de qualquer deficiência e indivíduos portadores de deficiências visuais, conclui-se que estes últimos estarão sempre mais dependentes de algo ou alguém, pelo que a capacidade de conseguirem levar a cabo uma rotina de ensaio autónoma é menor. Nestes casos, soluções intuitivas que promovam o aumento da autonomia por parte dos mesmos seria uma mais valia.

Em contraposição com este tipo de segmento de estudantes, existe também quem queira levar a cabo um tipo de aprendizagem menos teórico, recorrendo a recursos audiovisuais disponíveis em várias plataformas. No entanto, no caso de deficientes visuais, a acessibilidade a este tipo de informação não é de todo simples, pelo que seriam úteis, também para estes, alguns tipos de soluções digitais que visem agilizar este processo.

Assim, soluções digitais intuitivas que agilizem o processo de aprendizagem, nunca com o objetivo de substituir totalmente o ensino tradicional, tornam-se cruciais, contribuindo para a melhoria da qualidade de vida dos mesmos, promovendo uma diminuição da curva de aprendizagem e viabilizando uma diminuição na tendência que se faz sentir relativamente ao *digital divide*.

Posto isto, nesta secção apresentam-se algumas contribuições que se consideram relevantes, não só a nível tecnológico, mas também numa vertente sócio-cultural, na medida em que, por vezes, são criadas comunidades de entreajuda entre indivíduos sem qualquer deficiência visual e indivíduos invisuais. Dito isto, são soluções que contribuem para uma melhoria significativa na qualidade de vida de qualquer pessoa total ou parcialmente invisual.

2.2.1 Spoken Music

A leitura de música em *braille* implica um conhecimento bastante especializado. Se um individuo sabe ler num sistema *braille*, tal não implica que este saiba, também, ler partituras musicais em *braille*, pelo que este tipo de conhecimento implica um esforço que nem todos estão dispostos a investir (c.f secção 1.2). Isto traduz-se num grande obstáculo para quem pretende iniciar o seu processo de aprendizagem.

É com o principio chave de que todos os indivíduos com deficiências visuais tenham acesso à mesma informação que os indivíduos sem qualquer deficiência visual que, em 2002, surgiu o desenvolvimento do sistema digital *Spoken Music* [11].

Como o próprio nome indica - *Spoken Music* - esta solução tem como principal *moto* a disponibilização de toda a informação de uma partitura de uma forma falada, para todos os tipos de música e instrumentos, substituindo, assim, o *braille* aquando da leitura musical.

2.2.2 Musica Parlata

Tendo por base as mesmas premissas que levaram à implementação da solução *Spoken Music*, foi desenvolvido um sistema denominado *Musica Parlata* [17]. Apesar da evolução do mundo digital, os criadores de *Musica Parlata* reconheceram que essa evolução não se fez sentir nos dez anos que separam esta solução da solução anteriormente referida, pelo que optaram pelo desenvolvimento deste *software*.

Musica Parlata é uma solução digital que visa permitir a aprendizagem musical por parte de indivíduos interessados na área mas que, como a maioria, encontram nos novos símbolos *braille* um enorme obstáculo. Assim, a par com *Spoken Music*, esta solução pretende auxiliar a leitura de partituras musicais, por meio de um sistema de software que disponibiliza toda a informação de uma partitura via auditiva.

Apesar de ser uma solução funcionalmente semelhante à anterior, detém uma grande diferença relativamente à forma como passa a informação de cada nota da partitura, sendo que esta, ao invés de a "falar", "canta-a". Assim, o utilizador da aplicação irá ouvir, não só a nota, mas também a sua afinação, facilitando o processo de memorização de notas musicais quando comparado com o processo de memorização tradicional que implica a leitura de notas musicais numa partitura em *braille*. Devido à possível audição das notas musicais, o utilizador poderá paralelizar esta ação com a interação do seu instrumento. Considera-se, assim, um método imediato, na medida em que não são necessários estudos especializados em qualquer uma das áreas para iniciar o processo de aprendizagem sob esta metodologia.

Apesar de ser uma solução interessante, o desafio de apresentar tal quantidade de informação de forma "cantada", e em tempo real, implica a existência de um conjunto de limitações. Considerando que a própria duração da nota pretende ser transmitida ao utilizador, quando uma nota tem uma duração muito baixa, é impossível transmiti-la em tempo útil, pelo que esta será cortada. Existe ainda uma grande dificuldade na leitura simultânea de múltiplas linhas melódicas.

Figura 2.4: Interface gráfica do sistema digital *Musica Parlata* [17]

2.2.3 A Tangible MIDI Sequencer for Visually Impaired People

A ideia inerente a esta solução surgiu da necessidade que se sentiu de desenvolver um sistema que facilitasse a produção musical por parte de indivíduos portadores de deficiências visuais.

Na área de produção musical, os *softwares* mais cobiçados dependem, maioritariamente, do output gráfico produzido, pelo que, para indivíduos parcial ou totalmente invisuais, tal utilização não é sustentável. Assim, com o objetivo de colmatar a difícil interação com soluções digitais previamente existentes, mais orientadas a indivíduos que não acatem qualquer deficiência visual, desenvolveu-se, no ano 2009, um sequenciador *Musical Instrument Digital Interface (MIDI)* [18] - dispositivo ou *software* que permite ao utilizador gravar e editar uma peça musical sem recorrer a input áudio - orientado a indivíduos invisuais.

Aliada à tecnologia, para que esta solução acrescentasse valor e suprisse os objetivos iniciais, a interação entre o individuo invisual e o *software* deveria ser melhorada. Provida de *feedback* áudio em tempo real, a solução desenvolvida eliminou, por completo, a necessidade de utilização do ecrã digital e do teclado do computador sendo que expôs todos os comandos necessários à sua interação nas próprias teclas do instrumento do utilizador, como se pode verificar pela análise da figura 2.5.

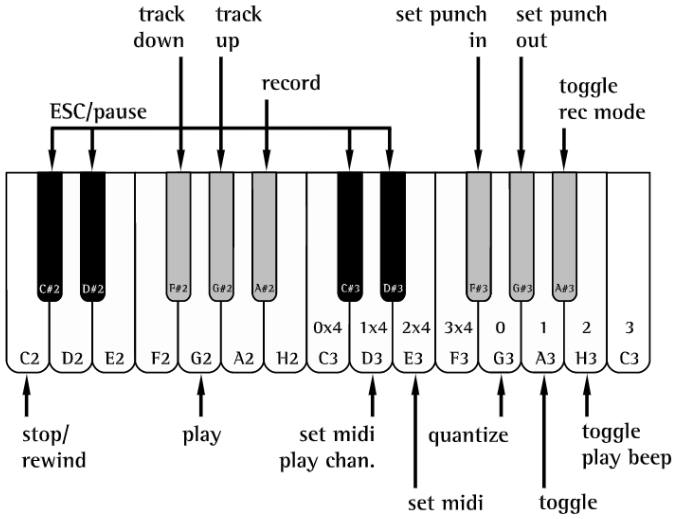

Figura 2.5: Disponibilização de comandos tangíveis na solução *A Tangible MIDI Sequencer* [18]

Apesar de cumprir o objetivo a que se propôs, esta solução é limitada na medida em que exige a existência de um teclado MIDI físico para a sua utilização e devido à sua implementação não ter sido contemplada como *cross-platform*, pelo que apenas existe disponibilizada para sistemas *Windows* e *Linux*.

2.2.4 Collaborative Music Application for Visually Impaired People with Tangible Objects on Table

Apesar de não ter como objetivo o ensino/aprendizagem musical, a solução presente considera-se uma contribuição de grande relevância para a comunidade na medida em que pretende ajudar pessoas com deficiências visuais, utilizando uma abordagem tecnológica e colaborativa, num mundo em que estas são, geralmente, marginalizadas da utilização digital.

Considerando que o trabalho colaborativo entre indivíduos invisuais e indivíduos que não detêm qualquer limitação apresenta resultados positivos no avanço social desses na sociedade, desenvolveu-se, em 2013, um sistema que potencia a colaboração entre os indivíduos através da música - *Collaborative Music Application for Visually Impaired People with Tangible Objects on Table* [19].

Esta solução tem como objetivo ser facilmente utilizada por indivíduos invisuais. Atendendo ao facto de que estes indivíduos possuem o sentido do tato excecionalmente mais apurado, esta foi desenvolvida visando uma interface tangível constituída por objetos físicos que se encontram expostos sob uma mesa.

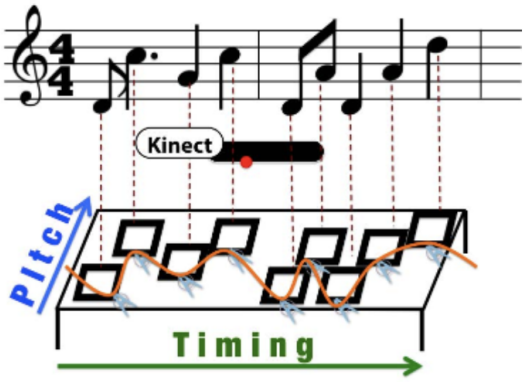

Figura 2.6: Esquema representativo do *hardware* utilizado na presente solução [19]

Como se pode verificar pela analise da disposição de toda a interface na figura 2.6, uma vez que a solução utiliza marcadores físicos aliados a tecnologias de realidade aumentada, o sentido do tato é estimulado, agilizando o processo de aprendizagem e potenciando a utilização intuitiva do sistema. A par com a utilização deste tipo de marcadores, são também utilizados sensores de visão com o objetivo de detetar a posição dos marcadores retornando, em *real-time*, o resultado numa nota musical.

De modo a consolidar esta informação, pela análise da figura 2.6 conclui-se que os conceitos de *Pitch* (tom) e *Timing* (tempo) são abordados de forma interativa e que, cada marcador, em função do tempo e do tom, irá resultar num *feedback* áudio relativo a uma nota musical.

2.2.5 Be My Eyes

> *"When a person is really grateful for something, it's something that money can't buy."[20]*
>
> *Fabio - Voluntário na Be my Eyes*
>
> *"I just used Be My Eyes, and, in just 30 seconds or so, a volunteer came, and she reads the name and the order of all buttons. And these are little buttons, you know, smaller than a fingertip to me."[21]*
>
> *Amir - Invisual na Be My Eyes*

Be My Eyes [22] é uma plataforma digital que visa potenciar a independência de indivíduos com deficiência visual, através do acesso a uma rede de pessoas que possuem o sentido da visão.

Para um sistema baseado neste modelo de negócio suceder, é crucial a existência de uma comunidade ativa, baseada numa associação mutua da qual todos os indivíduos beneficiam. À data, a plataforma conta com cerca de três milhões e meio de voluntários e cerca de duzentos mil invisuais, valores estes particularmente relevantes, pois permitem concluir que, efetivamente, existiu um desenvolvimento bem sucedido de uma comunidade digital que apoia esta causa social, sendo esta constituída por indivíduos necessitados que procuram ajuda e por indivíduos solidários com a causa que se sentem realizados por ajudar.

A funcionalidade principal desta solução é o pedido de ajuda por parte da pessoa invisual e a forma como tudo é processado desde esse momento. A primeira fase do pedido de ajuda trata-se da requisição do pedido. Para isto, os indivíduos invisuais possuem uma interface intuitiva e orientada à sua condição, como se pode verificar pela figura 2.7, sendo esta composta maioritariamente por um botão que permitirá acionar a requisição do pedido. A fase seguinte consiste na aceitação do pedido por parte de um voluntário, que receberá uma notificação com a informação respetiva. Uma vez aceite o pedido de ajuda, a plataforma *Be My Eyes* acede à câmara fotográfica do individuo invisual, que captará o vídeo que será transmitido em tempo real ao voluntário, que analisará o ambiente envolvente, partilhando o seu sentido de visão.

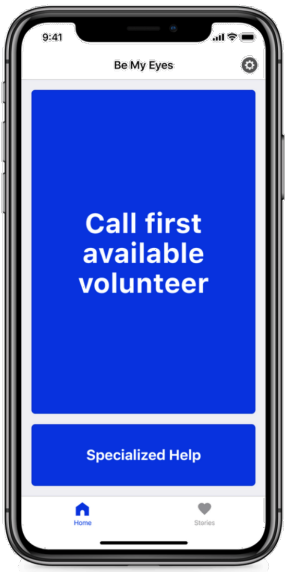

Figura 2.7: Interface para requisição de ajuda [22]

A utilização deste tipo de soluções digitais por parte de ambos os segmentos referidos tende a diminuir a exclusão social e digital que se faz sentir nos indivíduos invisuais, e permite concluir a utilidade das soluções digitais nas tarefas diárias dos dois segmentos.

2.3 Assistentes Digitais de Voz

A evolução natural da tecnologia convergiu na utilização de assistentes digitais no dia a dia, sendo que estes são agentes de *software* que realizam tarefas ou serviços com base em ações que lhes são impostas.

Segundo a *Oracle*, um assistente digital (também denominado de assistente virtual ou assistente de voz), é um *software* considerado avançado que visa a simulação de um diálogo com as pessoas que o utilizam, tipicamente de forma *online* [23].

Na última década o desenvolvimento da tecnologia de voz foi notável, pelo que foram lançados alguns assistentes digitais que são, ainda hoje, pontos de referência e de extensão para programas já existentes. Assim, e como se pode analisar pela figura 2.8, considera-se a década da revolução dos assistentes digitais [24].

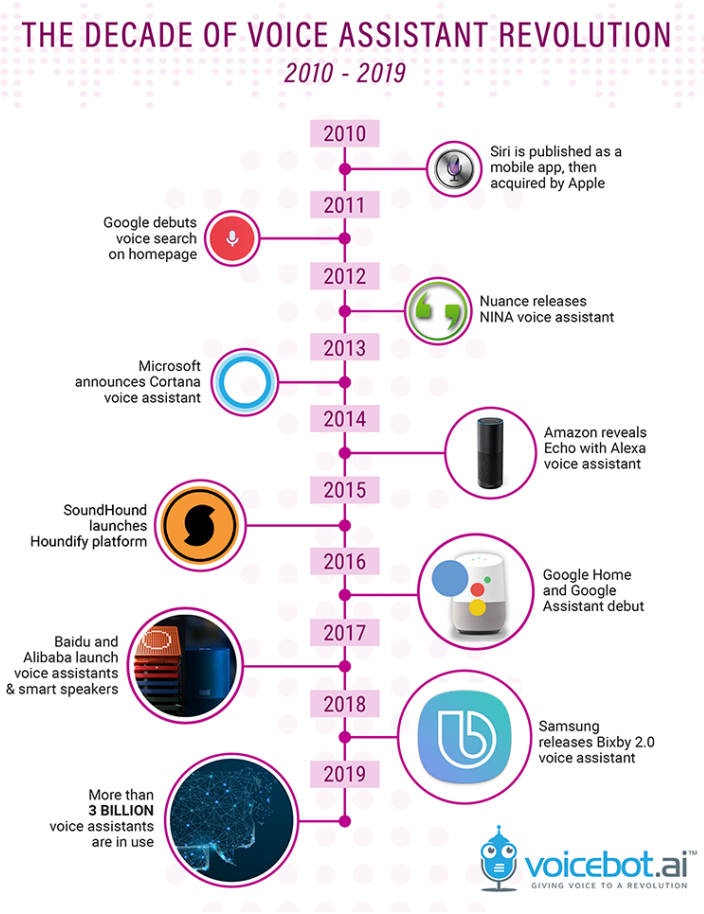

Figura 2.8: *Timeline* representativa da evolução dos assistentes digitais de voz [24]

A análise da figura anterior torna evidente a evolução existente nesta área ao longo dos últimos dez anos, sendo que importa enfatizar alguns dos nomes que contribuíram para esta expansão. Os assistentes digitais mais reconhecidos, desenvolvidos pelas organizações mais dominantes da área tecnológica: *Apple*, *Google*, *Amazon* e *Microsoft*, encontram-se representados na figura 2.9, e são denominados de, respetivamente: *Siri*, *Google Assistant*, *Alexa* e *Cortana*.

A par com o desenvolvimento de assistentes digitais por parte das *Big Tech* (termo utilizado para designar as empresas mais dominantes na industria tecnológica), existe o desenvolvimento de *stacks* tecnologias complementares, que comprovam a utilidade dos assistentes digitais. Conforme é possível analisar pela figura 2.9, de modo a complementar o desenvolvimento dos assistentes digitais, foram desenvolvidas plataformas de comunicação, de reconhecimento de linguagem natural e de *frameworks* de apoio ao desenvolvimento externo de soluções [25].

Figura 2.9: *Bot Ecosystem* [25]

Devido a todo este investimento das *Big Tech*, o conceito de assistentes de voz tem vindo a ser mais cobiçado. Recorrendo à analise de tendências, com base em dados reais presentes na plataforma *Google Trends* [26], relativos às pesquisas efetivadas no motor de pesquisa *Google*, à escala mundial, nos últimos cinco anos, pode comprovar-se a tendência crescente do conceito *Voice Assistant* ou Assistente de Voz (representado na figura 2.10).

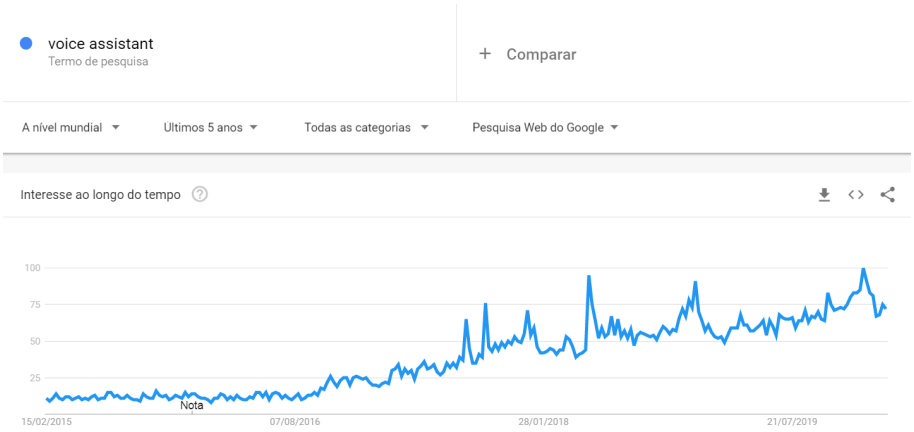

Figura 2.10: Gráfico representativo da tendência de pesquisas *online* pelo conceito *voice assistant* [27]

Para abordar o conceito de assistente como algo físico, envolvendo *hardware*, criou-se o conceito de *smart speaker*. Esta, é a definição utilizada para colunas que contêm sistemas de inteligência artificial integrado, proveniente da integração com o assistente digital respetivo. As colunas inteligentes: *Amazon Echo*, *Apple HomePod* e *Google Home*, são exemplos de colunas inteligentes que integram assistentes digitais - *Alexa*, *Siri* e *Google Assistant*, respetivamente.

2.3.1 Módulos de um Assistente Digital por Voz

Uma vez explicado o conceito e o contexto envolvente, torna-se interessante perceber como se processa o desenvolvimento de um assistente digital de voz. Na figura 2.11, encontra-se descrito o processo, dividido por fases desde o *input* via áudio por parte do utilizador, ao *output*, também via áudio, emitido pelo assistente digital [28].

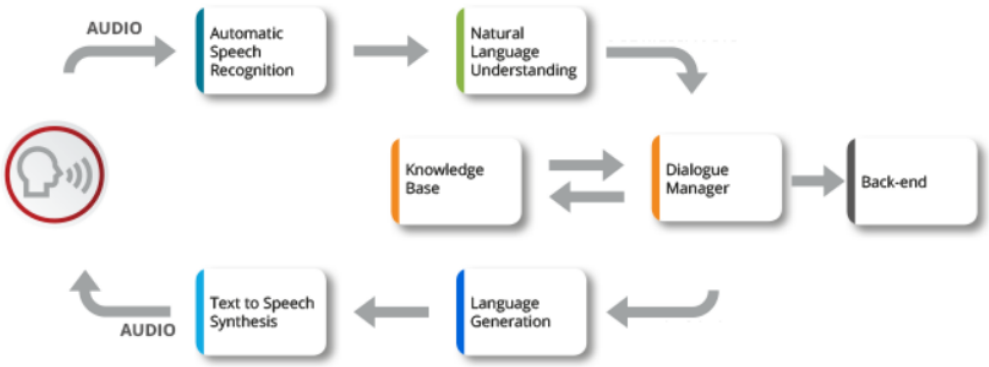

Figura 2.11: Sequência de interação com assistentes digitais por voz [28]

Através da análise da figura 2.11, conclui-se que, para se proceder ao desenvolvimento de um assistente digital por voz, os seguintes módulos devem ser desenvolvidos: Reconhecimento de Voz - *Automatic Speech Recognition (ASR)*, Entendimento de Linguagem Natural

- *Natural Language Understanding (NLU)*, Gestão de Diálogo - *Dialogue Manager*, Base de Conhecimento - *Knowledge Base*, Geração de Linguagem - *Language Generation* e Conversão de Texto para Voz - *Text to Speech*. Posto isto, apresenta-se, de seguida, uma breve descrição relativa aos módulos mencionados.

2.3.1.1 Automatic Speech Recognition - (Speech to Text)

A primeira fase de um assistente digital é o seu módulo de reconhecimento de voz, convertendo-a em texto. Segundo o livro *The Encyclopedia of Applied Linguistics* escrito por John Levis e Ruslan Suvorov em 2002, o reconhecimento de voz é descrito como sendo um processo de descodificação e transcrição de uma fala oral que tipicamente recebe como *input* uma fala, analisa-a recorrendo à utilização de algum tipo de padrão, modelo, ou algoritmo e produz, como *output*, o resultado transcrito em forma textual [29].

2.3.1.2 Natural Language Understanding

É importante enfatizar a diferença existente entre a capacidade de reconhecimento de voz, conforme descrito na secção 2.3.1.1, e a capacidade de a analisar e entender. Segundo a *Gartner* [30], o entendimento de linguagem natural é caracterizado por ser uma área de inteligência artificial que potencia a compreensão, por computadores, da estrutura e significado da linguagem humana, permitindo que os utilizadores interajam com o *software* através da utilização de frases naturais.

2.3.1.3 Dialogue Manager

Após o input do utilizador ser transcrito e entendido pelo sistema, torna-se necessário gerir o diálogo que advém dessa interação. O gestor de diálogo é um componente importante do sistema uma vez que este é o responsável por direcionar o diálogo com o utilizador, promovendo uma interação natural e eficiente controlada dentro dos limites e das capacidades do sistema. Assim, este deve responder à iniciativa de diálogo do utilizador, acompanhando todas as informações por este providenciadas [31].

2.3.1.4 Language Generation

A geração de linguagem natural é uma tecnologia que tem como principal objetivo a transformação de dados numa linguagem [32]. Assim, esta pode ser vista como o processo contrário ao de *NLU*, visto que nesta, o sistema necessita de tomar decisões acerca de como tornar certo conceito em palavras.

2.3.1.5 Text-to-Speech

Em oposição ao reconhecimento automático da fala (*Speech-To-Text*), a conversão de texto em fala (*Text-To-Speech*) é uma tecnologia categorizada como sendo uma tecnologia de apoio (conceito descrito na secção 1.1) que converte qualquer tipo de texto, independentemente da sua linguagem num output de voz natural e perceptível. Com este ultimo módulo, dá-se por terminado o conjunto modular que caracteriza a constituição de um assistente digital de voz.

2.3.2 Princípios de Design de Interfaces de Voz

A interface de voz de um assistente digital por voz é essencial uma vez que será o mecanismo responsável por expor todos os comandos de utilização, orientando o utilizador de uma forma intuitiva e pertinente. Uma vez que esta interface será o ponto de entrada de toda a interação homem-máquina, considera-se relevante a investigação das melhores práticas associadas ao desenvolvimento deste tipo de interfaces.

O conteúdo mais relevante advindo da pesquisa resume-se aos princípios propostos pela *Amazon*, empresa detentora de um dos assistentes digitais por voz mais conceituados do mercado [33], a *Alexa*. Considerando que a *Alexa* é um produto amplamente testado e reconhecido, quer pelo mercado, [34] quer pelos seus utilizadores [35], as práticas e princípios de *design* definidos pela *Amazon* relativamente ao tópico de *design* de interfaces foram consideradas um ponto de referência para o desenho da interface do projeto, uma vez que se aplicam a qualquer interface conversacional.

Deste modo, os princípios apresentados encontram-se divididos em duas fases principais - Obter dados do utilizador e apresentar informação ao utilizador - assim como definido na documentação oficial da *Alexa* [36].

2.3.2.1 Obter dados do utilizador

Os dados de maior relevância aquando da obtenção de dados do utilizador são os que se seguem:

- **Ser explícito relativamente a quando o utilizador deverá responder** - Se é necessário obter uma resposta por parte do utilizador, devem apresentar-se as opções disponíveis. No entanto, apresentar apenas as opções não informa explicitamente o utilizador de que será necessário o mesmo responder. Assim sendo, considera-se uma boa prática apresentar as opções em forma de questão de modo a que o utilizador seja persuadido a devolver uma resposta.
- **Não assumir que o utilizador sabe o que fazer** - Quando um utilizador utiliza pela primeira vez o assistente digital, é uma boa prática apresentar os comandos que existem para cada contexto. Não obstante, e complementando o principio anterior, de modo a que o utilizador saiba em que passo do fluxo se encontra e para que se relembre dos comandos que pode executar, o mesmo deve poder requisitar ajuda a qualquer momento, recebendo uma resposta contextualizada e com toda a informação necessária a que o mesmo não se sinta obrigado a ficar com alguma tarefa pendente porque não sabe qual o passo seguinte.
- **Apresentar as opções de resposta de forma evidente** - Quando se apresentam opções ao utilizador, deve optar-se por uma construção frásica que seja explicita no que diz respeito ao seu objetivo. A titulo exemplificativo, caso se pretenda que o utilizador escolha entre duas cores, é boa prática construir uma frase que não deixe qualquer dúvida relativamente ao tipo de resposta pretendida - Qual a cor pretendida: azul, ou vermelho? - induzindo o utilizador a responder ou "azul"ou "vermelho". No caso de não se considerar este principio de design e se apresentar uma frase menos bem conseguida ao utilizador - Pretende as cores azul ou vermelho? - o mesmo pode ser induzido em erro apresentando, por exemplo, uma resposta de "Sim"e "Não". Posto isto, considera-se relevante ouvir e interagir com todos os comandos aquando

do desenvolvimento de modo a encontrar possíveis construções frásicas que levem o utilizador a responder a algo de forma diferente à pretendida.

- **Evitar apresentação de demasiadas opções** - Quando uma lista de opções é apresentada a um utilizador, deve evitar-se repetição de palavras e deve-se, ainda, otimizar a apresentação das opções para que apenas se exponha o menor número de opções possíveis, que, ainda assim, cubram os casos necessários.
- **Ser sucinto** - Ao contrário do que acontece em interfaces gráficas, onde os utilizadores podem saltar informação que não pretendem analisar e divergir o foco de pesquisa rapidamente, tal não acontece em interfaces de voz. Num sistema visual existem mecanismos que permitem a exposição e ênfase de mais informação num menor intervalo de tempo (seja pela utilização de modais [1], *tooltips* [2], diferentes tipos de estilo, ou quaisquer outros mecanismos de diferenciação), no entanto, as interfaces de voz são lineares, pelo que alterar o foco de pesquisa não é aplicável e saltar informação nem sempre é indicado implicando a que o utilizador acabe por ser forçado a ouvir tudo. Assim, considera-se uma boa prática o estudo da melhor forma de comunicar o máximo possível no menor tempo possível, mantendo a construção frásica clara e concisa. Por último, deve ainda considerar-se a prática de *design* que indica que o sistema apenas deve responder à informação crítica, evitando a prática do erro do excesso de informação.
- **Perguntar apenas questões absolutamente necessárias** - Sempre que possível, o sistema deve inferir e evitar colocar mais uma questão. Caso contrário, estar-se-á a adicionar complexidade ao discurso piorando a sua experiência de utilização.
- **Obter informação de forma faseada** - Por vezes, a quantidade de dados requisitados ao utilizador deve ser realizada passo a passo para que diminua a complexidade do diálogo, e para que o utilizador sinta uma interação menos mecânica com o sistema.
- **Comandos universais** - Independentemente dos comandos suportados pelo sistema, existem alguns comandos universais que devem ser sempre suportados devido à sua utilização praticamente certa por parte do utilizador. Assim, quando aplicável, a implementação de comandos como "Olá", "Adeus", "Próximo", "Ir para o próximo", "Anterior"e "Ajuda"deve ser sempre assegurada. No entanto, apesar de serem comandos universais, tal não significa que o comportamento dos comandos seja sempre exatamente o mesmo, pelo que irá variar conforme o caso de uso a ser executado.

2.3.2.2 Apresentar informação ao utilizador

No que diz respeito aos princípios a considerar relativos à apresentação de dados ao utilizador, enunciam-se de seguida os pontos principais a adotar:

- **Utilizar marcadores de discurso** - A utilização de marcadores de discurso contribuem para a coesão frásica e promovem a humanização da conversa entre o utilizador e o sistema, permitindo ao utilizador perceber em que fase do fluxo se encontra e se, efetivamente, está a ser entendido pelo sistema.

[1] Janela que é exibida de forma sobreposta relativamente ao conteúdo principal da página com a qual se está a interagir.

[2] Mensagem que aparece quando um cursor é posicionado sobre qualquer elemento de uma interface gráfica.

- **Apresentar a informação de forma faseada** - Considerando o fluxo linear de uma interação por voz, considera-se uma boa prática a apresentação de informação de forma faseada de modo a que seja consumível pelo utilizador. Isto é, apenas se deve apresentar a informação que é estritamente necessária, pelo que, aquando da apresentação de listas longas, estas devem ser subdivididas para que sejam posteriormente apresentadas ao utilizador de forma faseada, permitindo questionar ao utilizador se pretende obter o resto da lista. Deste modo, o utilizador sente que tem o controlo da interação, melhorando a sua experiência.
- **Falar apenas quando necessário** - Aquando do *design* de um fluxo conversacional, deve-se considerar que, por vezes, o sistema não precisa de emitir uma resposta de voz para indicar que entendeu o comando, pois, por vezes, a mera execução do comando serve de confirmação (não verbal).
- **Utilizar métodos de confirmações** - Após o utilizador requisitar um comando, o sistema deve confirmar que o comando foi entendido. As confirmações são uma forma de mostrar ao utilizador que o mesmo foi percebido e podem existir sob várias formas, pelo que a utilização de cada uma, depende do contexto em causa. Assim sendo, estudaram-se três tipos de confirmações: Confirmação Explícita, Confirmação Implícita e Confirmação Silenciosa. A confirmação explicita é usada em casos de uso cuja ação final terá grande impacto a nível de negócio, pelo que, nesta, considera-se boa prática a dupla confirmação do comando a executar - como num caso de uso de marcação de viagens. Por outro lado, a confirmação implícita é utilizada quando não existe grande consequência advinda da execução do comando - como num caso de uso em que existirá apenas uma pergunta seguida de uma resposta. Por último, existe a confirmação silenciosa, na qual a execução do comando requisitado pelo utilizador já é uma confirmação em si - como num caso de uso no qual se liga uma lâmpada.
- **Não culpar o utilizador** - Independentemente do comando utilizado e da circunstância em que foi utilizado, o sistema não deve culpar o utilizador sob o risco de que, se o fizer, poderá transparecer hostilidade.
- **Gerir erros** - Aquando de uma interação com uma interface de voz, existem várias formas de detetar exceções. Assim, apresentam-se
- **Latência** - É importante determinar se o sistema irá conter, de forma controlada, algum tipo de atraso ou latência na resposta a qualquer comando inserido pelo utilizador. Caso aconteça, esta latência deve ser controlada, pelo que se deve dar *feedback* ao utilizador antes de obter a informação e depois de a obter.

2.3.3 Tecnologias para Desenvolvimento de Assistentes Digitais

A *stack* tecnológica que auxilia o desenvolvimento de um assistente digital por voz é vasta e encontra-se dividida por módulos. Ainda assim, por vezes, há tecnologias que abstraem o desenvolvimento de mais do que um módulo.

Nas subsecções seguintes, apresentam-se algumas das tecnologias disponíveis, associando-as ao módulo que estas auxiliam.

2.3.3.1 Google Cloud Speech-To-Text

Software desenvolvido pela Google no ano de 2017, que fornece uma *Application Programming Interface (API)* que visa ser de simples utilização. O seu objetivo é a conversão de áudio em texto, sendo que permite o reconhecimento de cerca de cento e vinte idiomas. Dadas as suas funcionalidades, conclui-se que é orientado ao desenvolvimento de módulos de Reconhecimento de Voz (ASR).

2.3.3.2 Web Speech API

A *Web Speech API* é uma tecnologia que permite que aplicações *web* sejam capazes de lidar com dados de voz [37]. Esta, é suportada pelo *World Wide Web Consortium (W3C)*, e permite que os navegadores *web* forneçam um mecanismo de reconhecimento e síntese de fala através de uma API que pode ser usada diretamente no *browser* sem que sejam preocupantes os seus limites relativamente à quantidade de pedidos suportados. Considera-se a mais indicada aquando do desenvolvimento de sistemas que necessitem de um conjunto de recursos simples.

Assim sendo, esta é constituída por dois componentes principais:

- *Speech Recognition* - Fornece a possibilidade de reconhecer voz através de um *input* áudio que será analisado por um serviço de reconhecimento de fala. O resultado desta análise é o conteúdo falado em formato textual. Dito isto, conclui-se que é uma tecnologia orientada ao módulo de Reconhecimento de Voz (ASR).
- *Speech Synthesis* - Componente *Text-to-Speech* que, como o nome indica, possibilita a leitura em linguagem natural através do reconhecimento de conteúdo em forma textual.

2.3.3.3 Google Cloud Text-To-Speech

À semelhança do *Google Cloud Speech-To-Text*, este é um *software* também desenvolvido pela Google que converte texto em fala humana, contando com a existência de suporte para mais de cento e oitenta vozes em mais de trinta idiomas [38]. Dadas as suas funcionalidades, conclui-se que é orientado ao desenvolvimento de módulos de *Text-To-Speech*.

2.3.3.4 Amazon Polly

É uma tecnologia que transforma texto em fala, potenciando o desenvolvimento de sistemas com esta vertente interativa. Conta com a existência de dezenas de vozes realistas aliadas a um vasto conjunto de idiomas. Para além das vozes *Text-to-speech* (TTS) padrão, o *Amazon Polly* disponibiliza, ainda, vozes de conversão neural de texto em fala (*Neural Text-To-Speech*), que se traduzem numa melhoria avançada na qualidade de fala por meio de uma abordagem de *machine learning* [39]. Considerando as funcionalidades transcritas, conclui-se que é um *software* orientado ao módulo de *Text-to-Speech*.

2.3.3.5 DialogFlow

É uma tecnologia desenvolvida pela empresa *Speaktoit*, inicialmente denominada *Api.ai*, tendo sido adquirida em 2016 pela Google que alterou o seu nome para *Dialogflow*. O objetivo desta tecnologia é agilizar o desenvolvimento de sistemas digitais conversacionais,

permitindo a concepção de interfaces que potenciam interações naturais homem-máquina [40].

De modo a que seja possível auxiliar todo o processo correspondente à interação inteligente entre um humano e um sistema digital, esta tecnologia possui vários módulos que visam dar resposta a problemas diferentes. Assim, esta tecnologia contém componentes que visam dar resposta a problemas relativos aos módulos de Reconhecimento de Voz (ASR), NLU, Gestão de Diálogos, Geração de Linguagem e, ainda, *Text-To-Speech*.

2.3.3.6 Amazon Lex

Tecnologia desenvolvida pela *Amazon*, orientada ao desenvolvimento de interfaces conversacionais através da utilização de texto ou som. Disponibiliza funcionalidades avançadas de aprendizagem, utilizadas para converter fala em texto, e para reconhecimento de linguagem natural, sendo, desta forma, orientada a módulos de Reconhecimento de Voz (ASR) e NLU. Como complemento, à semelhança do *Dialogflow*, também contém componentes que auxiliam os módulos de gestão de diálogo, geração de fala e *Speech-To-Text* [41].

2.3.3.7 Microsoft Bot Framework

É um conjunto complexo de serviços desenvolvidos pela *Microsoft* que providenciam uma *framework* completa para viabilizar o desenvolvimento de sistemas digitais inteligentes. Para isto, à semelhança do *Dialogflow* e *Amazon Lex*, contém vários componentes que pretendem dar resposta a vários módulos como Reconhecimento de Voz (ASR), NLU, Gestão de Diálogos, Geração de Linguagem e, também, *Text-To-Speech*. Para além destas funcionalidades, conta com um conjunto vasto de tecnologias complementares que não serão abordadas por não se considerar relevante para o modelo deste projeto [41].

2.4 Tecnologias Web de Análise de Som

De modo a analisar e manipular som em ambientes de desenvolvimento *web*, identificaram-se três tecnologias que apresentam uma solução ao problema: *WebAudioAPI* [42], *Howler.js* [43] e *HTML Audio* [44].

Das três tecnologias identificadas, a primeira a surgir foi a *HTML Audio* que, apesar de se considerar uma tecnologia básica à data de escrita deste documento, foi essencial para que se desenvolvessem novas tecnologias mais avançadas baseadas nesta. Dito isto, esta tecnologia permite manipular som em aplicações *web*, no entanto, considera-se uma tecnologia um pouco limitativa na medida em que apenas permite a incorporação de conteúdo de som na *web*, não permitindo analisá-lo nem alterá-lo.

A *WebAudioAPI* é uma API *javascript* de alto nível e fornece um sistema significativo e versátil para controlar áudio em aplicações *web*. O objetivo principal desta é agregar recursos encontrados em motores de áudio existentes, fornecendo funcionalidades de mistura, processamento, filtragem e análise de som de modo a permitir que estes sejam utilizados na *web*. Esta API permite a manipulação de operações dentro de um contexto de áudio (*audio context*) e permite o encaminhamento modular. Ou seja, todas as operações áudio que se pretendam executar, serão implementadas recorrendo a nós de áudio (*audio nodes*) que se conetam entre si, em cadeia, através dos seus *inputs* e *outputs*, de modo a formar um gráfico de encaminhamento de áudio (*audio routing graph*). Esta modularização aumenta a

flexibilidade da tecnologia e permite o desenvolvimento de funções complexas que envolvam áudio.

Por último, identificou-se a tecnologia *Howler.js*, desenvolvida pela *GoldFire Studios*. Esta é uma biblioteca de áudio *javascript* e *open source*, orientada a aplicações *web*, com suporte para vários navegadores, que permite reproduzir e analisar som. Nesta biblioteca o utilizador tem total controlo uma vez que a análise do som é viável, pois é feito uso da própria *WebAudioAPI*, assim como é feito uso de *HTML Audio* para que seja exequível a reprodução do som.

2.5 Frameworks de Desenvolvimento Web

A *stack* tecnológica de desenvolvimento de aplicações *web* atualmente existente é bastante extensa, pelo que devem existir estudos prévios que fundamentem cada escolha tecnológica. Deste modo, apresenta-se nesta secção uma análise sintetizada de forma a fundamentar as opções tomadas, relativamente às *frameworks* de desenvolvimento a aplicar no projeto atual.

Uma vez que o assistente digital a desenvolver será uma aplicação *web*, é boa prática proceder-se à definição dos critérios e métricas que se procuram, para que se tomem opções viáveis relativamente às *frameworks* a utilizar. Considerando os requisitos funcionais e não funcionais do projeto atual, presentes nas secções 4.2 e 4.3, respetivamente, e com base em estudos previamente dirigidos [45][46], definiu-se a seguinte listagem de critérios e métricas que, uma vez cumpridos, validam a utilização da *framework* candidata:

- Comunidade ativa - A existência de uma comunidade ativa está diretamente ligada à popularidade de uma certa tecnologia pelo que, uma vez existindo, a tecnologia torna-se mais credível. Uma forma de avaliação deste critério passa por se proceder à análise do rácio de perguntas e respostas relacionadas com a tecnologia, em plataformas de perguntas e respostas [46] como o *StackOverflow* [47].
- Maturidade e frequência de atualizações - A maturidade de uma tecnologia é um critério a adotar na medida em que, geralmente, implica uma maior robustez da tecnologia, uma vez que já foi bastante testada e utilizada, conferindo-lhe essa característica. No entanto, a maturidade, por si só, não garante a atualidade da tecnologia, pelo que se considera que a frequência de atualizações também deve ser uma métrica a considerar e a relacionar. A frequência de interações no sistema de controlo de versões das tecnologias representa tanto a participação da comunidade como a inovação e frequência de atualizações do produto [46].
- Integração com *frameworks* de teste - Como qualquer sistema tecnológico deve ser testado, a integração com *frameworks* de teste deve ser um critério a considerar.
- Arquitetura modular - É importante que a *framework* a adotar promova a arquitetura modular para que seja facilitada a futura manutenção do código. A modularidade arquitetural viabiliza um aumento na coesão e permite, ainda, diminuir o acoplamento do código, conforme os padrões GRASP *High Cohesion* e *Low Coupling* [48].
- Qualidade de documentação - A documentação é uma parte importante da captura do conhecimento de um projeto de software, providenciando uma forma de guardar e passar informação adicional [49]. O seu foco é a manutenção e a passagem de conhecimento entre as partes interessadas, pelo que, quando bem executada, torna a

informação mais facilmente acessível facilitando a aprendizagem de novos utilizadores, simplificando o produto e ajudando a reduzir os custos de suporte. Deste modo, considera-se um critério importante a considerar.

- Curva de aprendizagem - A curva de aprendizagem da *framework* não deve ser muito acentuada uma vez que se tal acontecer pode colocar em causa a viabilidade do projeto considerando as *deadlines*.

Uma vez apresentados os critérios, que foram aplicados tanto à pesquisa de *frameworks* de desenvolvimento *frontend*[3] como de *backend*[4], concluiu-se a utilização das *frameworks* *Angular* 2+ e *Node Express*.

2.6 Conclusão

A titulo conclusivo, apesar de existirem várias soluções tecnológicas e mais intuitivas que visam apoiar o ensino musical de modo a diminuir o *digital divide* (c.f secção 2.2), denota-se uma tendência crescente tanto na relevância da utilização de assistentes digitais como na própria pesquisa pelo termo, conforme representado na secção 2.3, pelo que até se considera estar na década dos assistentes digitais [24]. Assim, conforme descrito na secção 2.3, existem vários assistentes digitais previamente implementados por grandes empresas que se encontram integrados em sistemas de colunas inteligentes. O ideal seria desenvolver um sistema que apoiasse o ensino musical e que se encontrasse integrado com estes assistentes digitais. No entanto, por motivos de segurança, todos os assistentes digitais existentes permitem apenas identificação de comandos de voz, pelo que excluem a possibilidade de qualquer tratamento de som. Deste modo, seria impraticável desenvolver um assistente digital de apoio ao ensino musical com base nestes sistemas. Assim sendo, conclui-se que a solução para o problema passa pelo desenvolvimento de um novo assistente digital que, para além de agregar todos os módulos existentes nos assistentes já desenvolvidos (c.f 2.3.1), irá conter ainda um módulo de análise de som baseado nas tecnologias de análise de som referidas na secção 2.4. Deste modo, torna-se possível a existência de um assistente digital que seja capaz de reconhecer comandos de voz, perceber a linguagem natural, gerir discursos de modo a gerar frases que sejam naturalmente percetíveis e que, por último, serão sintetizadas em fala inteligível ao ouvido humano.

Posto isto, para que se desenvolva uma solução que integre todos os módulos de assistente digital e o módulo de análise de som, apresenta-se nas secções 2.6.1 e 2.6.2 uma análise comparativa entre as tecnologias previamente identificadas, seguidas de uma escolha tecnológica justificada.

2.6.1 Comparação de Tecnologias para Desenvolvimento de Assistentes Digitais

No que diz respeito à *stack* tecnológica relativa ao desenvolvimento de assistentes digitais (c.f 2.3.3), apresentam-se de seguida várias tabelas que comparam as tecnologias previamente identificadas e abordadas, agrupadas pelo módulo para o qual são orientadas. Importa referir que todas as tabelas contêm os mesmos critérios de comparação visto que, apesar de serem orientadas a módulos distintos, se tratam todas de soluções tecnologias. Optou-se

[3] Desenvolvimento *frontend* gere tudo o que o utilizador visualiza na sua aplicação *web*.

[4] Desenvolvimento *backend* refere-se ao lado do servidor da aplicação *web* no qual se gere a comunicação entre os componentes internos do sistema.

pela análise de três critérios que se consideram os mais relevantes no âmbito do desenvolvimento do projeto em questão. Assim, tecnologias que sejam inteiramente gratuitas e sem limitações, que contenham uma comunidade *online* que promova a entreajuda e cujo custo-benefício da sua implementação seja positivo, irão prevalecer relativamente às que não cumprirem estes requisitos.

Posto isto, na tabela 2.1, apresenta-se a comparação tecnológica orientada ao módulo de Reconhecimento de Voz.

	Utilização Ilimitada	Existência de Comunidade	Custo-Benefício Positivo
Google Cloud Speech-To-Text		x	
Web Speech API (Speech Recognition)	x	x	x
Dialogflow (Speech-To-Text)		x	
Amazon Lex		x	
Microsoft Bot Framework		x	

Tabela 2.1: Análise Comparativa de Tecnologias relativa ao módulo de Reconhecimento de Voz

Através da análise da tabela anterior, conclui-se que a tecnologia mais indicada para o desenvolvimento do módulo de Reconhecimento de Voz é a *Web Speech API (Speech Recognition)*. Para além de existir uma comunidade *online* especializada [50], esta solução permite a utilização ilimitada do serviço que disponibiliza, uma vez que se encontra integrada em navegadores *web*. Consequentemente, o custo-beneficio da sua implementação é positivo considerando que não será necessário depender de tecnologias disponibilizadas em bibliotecas externas para o desenvolvimento deste módulo.

Na tabela 2.2, apresenta-se a comparação tecnológica orientada aos módulos NLU, *Dialogue Manager* e *Language Generation*.

	Utilização Ilimitada	Existência de Comunidade	Custo-Benefício Positivo
Dialogflow		x	x
Amazon Lex		x	x
Microsoft Bot Framework		x	x

Tabela 2.2: Análise Comparativa de Tecnologias relativas aos módulos *NLU*, *Dialogue Manager* e *Language Generation*

A tabela anterior permite concluir que nenhuma das tecnologias tem utilização ilimitada, no entanto, todas elas possuem comunidades que agilizem o desenvolvimento. O custo-benefício da implementação é positivo em todas as tecnologias uma vez que, apesar de serem sistemas complexos, irão abstrair a implementação de vários módulos. Deste modo, qualquer tecnologia neste caso seria uma boa abordagem, no entanto, considerando o número elevado de informação relativa à tecnologia *Dialogflow online*, esta considera-se a tecnologia mais indicada.

Na tabela 2.3, apresenta-se a comparação tecnológica orientada ao módulo *Text-To-Speech*.

	Utilização Ilimitada	Existência de Comunidade	Custo-Benefício Positivo
Web Speech API (Speech Synthesis)	x	x	x
Google Cloud Text-To-Speech		x	x
Amazon Polly		x	x
Dialogflow (Text-To-Speech)		x	

Tabela 2.3: Análise Comparativa de Tecnologias relativas ao módulo *Text-To-Speech*

No que diz respeito ao módulo *Text-To-Speech*, a tabela anterior permite concluir que a tecnologia que se considera a mais indicada é a *Web Speech API (Speech Synthesis)*, que, à semelhança da tecnologia *Web Speech API (Speech Recognition)*, não só possui comunidade *online* como possibilita uma utilização ilimitada e, ainda, permite obter um custo-benefício de implementação positivo dada a sua integração no navegador *web*.

2.6.2 Comparação de Tecnologias Web de Análise de Som

Uma vez comparadas as tecnologias relativas ao desenvolvimento dos módulos constituintes de um assistente digital, apresenta-se, na tabela 2.4, uma comparação entre as tecnologias de manipulação de som analisadas na secção 2.4.

	Permite Conexão Modular de Nós Áudio	Capacidade de Reproduzir Áudio	Existência de Comunidade Ativa
HTML Audio		x	
WebAudioAPI	x	x	x
Howler.js	x	x	x

Tabela 2.4: Análise comparativa de tecnologias de análise de som

A análise da tabela 2.4 permite concluir que a tecnologia *HTML Audio* não cumpre com todos os requisitos impostos, uma vez que nem permite a conexão modular de nós áudio nem se encontra integrada numa comunidade *online* ativa. A restrição relativa à capacidade modular impossibilita a mistura e a análise de som e a limitação em termos de comunidade dificultaria a sua implementação e manutenção.

Em contrapartida, tanto a API *WebAudioAPI* como a biblioteca *Howler.js* cumprem todos os requisitos definidos. Assim, ambas permitem uma conexão modular, possibilitam a reprodução de áudio e contam com uma comunidade ativa. No entanto, tendo em conta que a *WebAudioAPI* é uma tecnologia que se encontra integrada, por defeito, no mecanismo do *browser* e considerando que a biblioteca *Howler.js* implica o carregamento e a dependência de um módulo externo, conclui-se que a tecnologia mais indicada é a *WebAudioAPI*.

Capítulo 3

Análise de Valor

O processo de criação de valor não é um processo de simples execução, no entanto, para que se identifiquem e previnam custos desnecessários advindos de maus investimentos, torna-se um processo essencial aquando do desenvolvimento de um novo produto/serviço.

Neste capítulo encontra-se representado todo o processo da análise de valor, desenvolvido com base em vários artefactos, através da aplicação de várias técnicas de forma a que se conclua se, efetivamente, existe valor no produto/serviço a desenvolver para o segmento de mercado pretendido.

3.1 Inovação, New Concept Development

O conceito de inovação surge da insatisfação com o atual estado de arte e encontra-se ligado à mudança, seja pela introdução/implementação de novas ideias, ou até pela definição de processos que envolvam atividades cujo objetivo passe por descobrir novas formas de realizar tarefas, melhorando processos.

Considerando o ambiente atual caracterizado pelo ritmo acelerado da evolução tecnológica e pelo aumento da globalização, torna-se necessário inovar constantemente produtos/serviços e/ou processos, sendo que o objetivo principal é sempre o de acrescentar valor tanto à organização como ao cliente, seja pela mudança de comportamentos, lançamento de novos padrões, mudança de rotina ou formas de pensar.

Como proposto por Peter A.Koen e os seus co-autores, num livro publicado em 2002 titulado *The PDMA ToolBook for New Product Development* [51], o processo de inovação, representado na figura 3.1, divide-se em três segmentos distintos e sequenciais.

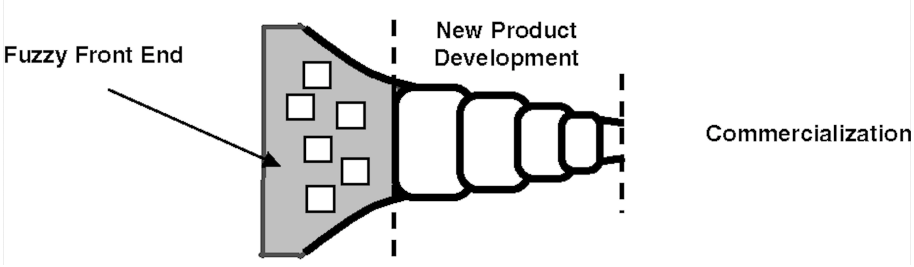

Figura 3.1: Processo de Inovação [51]

Este processo tem como finalidade marcar todo o processo desde a criação à comercialização de um certo produto. Assim, sendo que todo ele funciona de forma sequencial é expectável que só cheguem à fase de comercialização produtos cujo valor tenha sido devidamente

provado e validado nas fases anteriores. Posto isto, pode verificar-se através da análise da figura anterior, as três fases que o caracterizam:

- *Fuzzy Front End (FFE)*;
- *New Product Development (NPD)*;
- Comercialização.

O primeiro passo no processo de inovação é o FFE, termo este concebido por Reinertsen e Smith à data de 1991 [52]. Este tem como objetivo o desenvolvimento de um conceito advindo de uma oportunidade previamente identificada. Para que isto seja concretizável, todo o processo composto pela análise da oportunidade, geração e enriquecimento de ideias, seleção das mesmas e, por último, definição de conceito, tem de ser aplicado.

No passo seguinte do processo encontra-se a fase de NPD. Nesta, existe um conjunto de etapas como o *design*, o desenvolvimento e o *marketing* de bens ou serviços, com o objetivo de aumentar a participação de mercado de uma empresa, satisfazendo a procura por parte do consumidor.

A finalizar o processo, encontra-se a fase da Comercialização, na qual se vende o produto desenvolvido, seja ele completamente novo ou, apenas, uma alteração a um já existente.

De forma a suportar a fase de *FFE*, o modelo *New Concept Development (NCD)* - representado na figura 3.2 - foi desenvolvido por Peter A. Koen em 2002 [51], providenciando uma nomenclatura capaz de definir os componentes chaves desse processo.

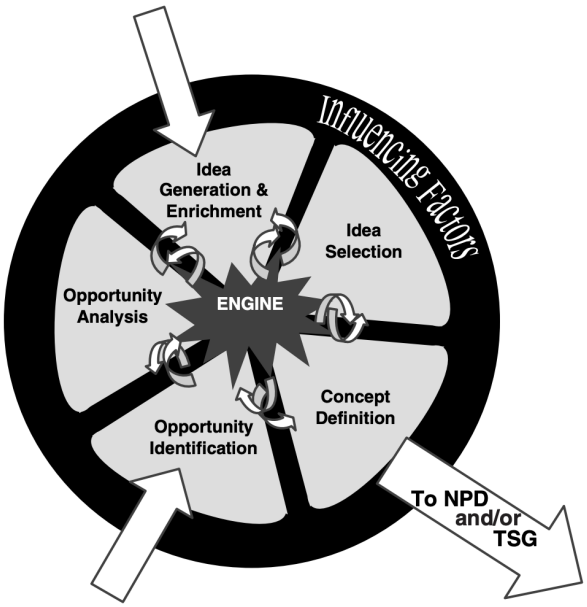

Figura 3.2: *New Concept Development Model* [51]

Conforme representado na figura anterior, o NCD é caracterizado pela existência de três áreas distintas:

- *Engine* (Motor) - Representa a estratégia da organização e a cultura e tem como finalidade mobilizar e impulsionar os cinco elementos chave.

- Cinco elementos chave de atividade - Representam todas as fases que separam o processo desde a identificação de uma oportunidade até à criação de um conceito.
- Fatores influenciadores - Representam os fatores externos e, portanto, não controláveis, que influenciam o processo de inovação (motor e elementos chave).

Através da aplicação das fases previamente enumeradas, torna-se possível analisar em que medida diferentes características do problema são afetadas, tanto por fatores influenciadores externos, como pela componente organizacional. Assim, nas subsecções seguintes, apresenta-se a aplicação da segunda área do modelo, de modo a que se explicite o processo de geração da ideia e da sua seleção.

3.1.1 Identificação de Oportunidade

A evolução digital tem permitido uma melhoria significativa em praticamente todos os processos abrangidos pelo quotidiano, ainda assim, o problema de exclusão digital - fenómeno no qual extensas camadas de sociedade se encontram marginalizados de sistemas digitais - é uma realidade para vários segmentos da população.

De acordo com o *Royal National Institute of Blind People (RNIB)* [53], e no que diz respeito à população portadora de deficiência visual, os valores referentes à quantidade de pessoas desabilitadas que não recebem o apoio devido por parte de serviços governamentais, são elevados. Assim, é ainda uma realidade a exclusão social e digital de indivíduos desabilitados, tanto para a gestão de tarefas básicas do quotidiano, como para usufruir de serviços *online* na busca de informação e/ou, até, para apoio e estabilidade emocional na fase de adaptação e ajuste à perda de visão.

Com base na informação referida previamente, identificou-se, particularmente, uma grande dificuldade na aprendizagem musical por parte destes indivíduos. Assim, é na convergência dos problemas identificados na secção 1.2 que se identifica a oportunidade, visando contribuir para a diminuição da lacuna digital sentida por estas pessoas através de uma solução digital que assista na aprendizagem musical (seja esta formal ou informal).

3.1.2 Análise de Oportunidade

Uma vez identificada a oportunidade, torna-se necessário fundamentá-la e analisá-la para que se concluam as vantagens e desvantagens advindas da sua implementação. Considerando que a oportunidade identificada é de índole digital, torna-se pertinente averiguar se as pessoas com deficiências visuais possuem uma atitude positiva face à utilização da tecnologia, analisar a possível existência de algum tipo de resistência face à sua utilização e, por último, estudar a curva de evolução do conceito relativo a assistentes digitais com base nas tendências de pesquisa.

De acordo com o RNIB, num estudo realizado em 2015 [54] a atitude face à utilização de tecnologias encontra-se segmentada por idades, sendo que 86% das pessoas com idades compreendidas entre os 18 e os 29 revelam uma atitude bastante positiva e uma certa proatividade no que diz respeito à sua utilização. Este valor percentual vai decrescendo à medida que se avança nas faixas etárias, sendo que apenas 25% dos indivíduos cuja idade seja de 75 anos ou mais revelam essa mesma atitude. Com isto, conclui-se que a atitude positiva face à utilização de tecnologias não será preocupante nem impeditiva.

Para além da atitude face à tecnologia, deve analisar-se também a resistência à utilização da mesma. A existência de plataformas digitais como a *Be My Eyes*, conforme desenvolvido na secção 2.2.5, que conta com um valor de cerca de 3.3 milhões de voluntários e mais de 180 mil pessoas portadoras de deficiências visuais [22] demonstram a capacidade de criação de uma comunidade neste segmento de mercado, permitindo tirar conclusões positivas face ao analisado.

Uma vez analisadas as vertentes sociais, deve-se analisar, na figura 3.3, o gráfico relativo à tendência de pesquisas *online* pelo conceito "assistente digital"(*digital assistant*).

Figura 3.3: Gráfico representativo da tendência de pesquisas *online* pelo conceito *digital assistant* [55]

Para que isto fosse possível, recorreu-se à utilização da ferramenta *Google Trends* [26], porque, apesar de apenas conter um subconjunto relativo ao total de pesquisas realizadas no motor de busca *Google*, tal é suficiente na medida em que, neste, são processados milhares de milhões de pedidos diariamente [56]. Dito isto, o subconjunto de dados que abastecem a base de dados utilizada são fiáveis e conclusivos. Sendo assim, a análise da figura 3.3 permite concluir que, a nível mundial, num intervalo de tempo constituído pelos últimos cinco anos, o conceito em questão tem sido mais pesquisado, demonstrando um aumento de interesse por parte da população, traduzindo-se assim, num argumento positivo face à implementação da oportunidade identificada.

3.1.3 Geração de Ideias

Uma vez demonstrada a validade da oportunidade identificada, foi necessário proceder-se à geração de ideias que possam vir a ser promovidas a soluções. Para isto, para que se concluísse a melhor abordagem de desenvolvimento de um sistema digital que cumprisse os requisitos, agendou-se uma sessão de *brainstorming* entre o Orientador do projeto e o mestrando. As possíveis soluções segmentaram-se, concluindo-se que se poderia seguir uma abordagem baseada no desenvolvimento de um assistente digital por voz ou de um *chatbot*. Apesar de ambos terem o poder de desenvolver o papel de assistentes virtuais digitais, estes diferem no meio de interação, sendo que um deles é via áudio e outro é via texto.

Posto isto, as ideias finais obtidas foram as seguintes:

- Desenvolvimento de um assistente digital *custom*;

- Utilização de um assistente digital já existente (como colunas inteligentes);
- *Chatbot*.

3.1.4 Seleção de Ideias

Uma vez identificadas as ideias que pretendem dar solução à oportunidade identificada, torna-se necessário decidir, de forma fundamentada, qual implementar. Sendo uma escolha crucial ao desenvolvimento do projeto e dada a sua complexidade, optou-se pela utilização de um método de apoio à decisão multicritério. Devido à sua abordagem racional, baseada na utilização de técnicas matemáticas, e no cruzamento entre as alternativas e os critérios existentes, optou-se pela utilização do método *Analytic Hierarchy Process (AHP)*, proposto por *Thomas L. Saaty* em 1980 [57].

Existem quatro fases deste método que devem ser abordadas aquando da sua utilização. Assim, a primeira fase consiste no desenvolvimento de uma árvore hierárquica de decisão, definindo-se o problema inicial, seguido dos critérios e das alternativas utilizadas para a decisão final. Como referido na secção 3.1.3, o objetivo final para o qual se torna necessário a utilização deste método é definir qual a abordagem de desenvolvimento de um assistente virtual mais adequada. Para que se chegue à solução final, esta deve ser avaliada com base nos seguintes critérios:

- Possibilidade de integração com sistemas de gestão de diálogo;
- Capacidade de reconhecimento de voz;
- Capacidade de reconhecimento de som.

O primeiro ponto, referente à integração com sistemas de gestão de diálogo, é imprescindível dado que potencia o desenvolvimento de um assistente devidamente inteligente. De seguida, no que diz respeito à capacidade de reconhecimento de voz, esta é importante na medida em que viabilizará uma via de comunicação verbal, agilizando e facilitando o processo de comunicação. Por último, é fulcral a possibilidade de reconhecimento de som para que existam automatismos relacionados com os sons de instrumentos musicais.

Considerando o objetivo final pretendido e os critérios supramencionados, as três ideias que foram identificadas na secção 3.1.3 serão promovidas a alternativas visando, no final do processo, encontrar a possível solução.

Na figura seguinte, figura 3.4, apresenta-se um gráfico representativo da estrutura hierárquica do método AHP a aplicar.

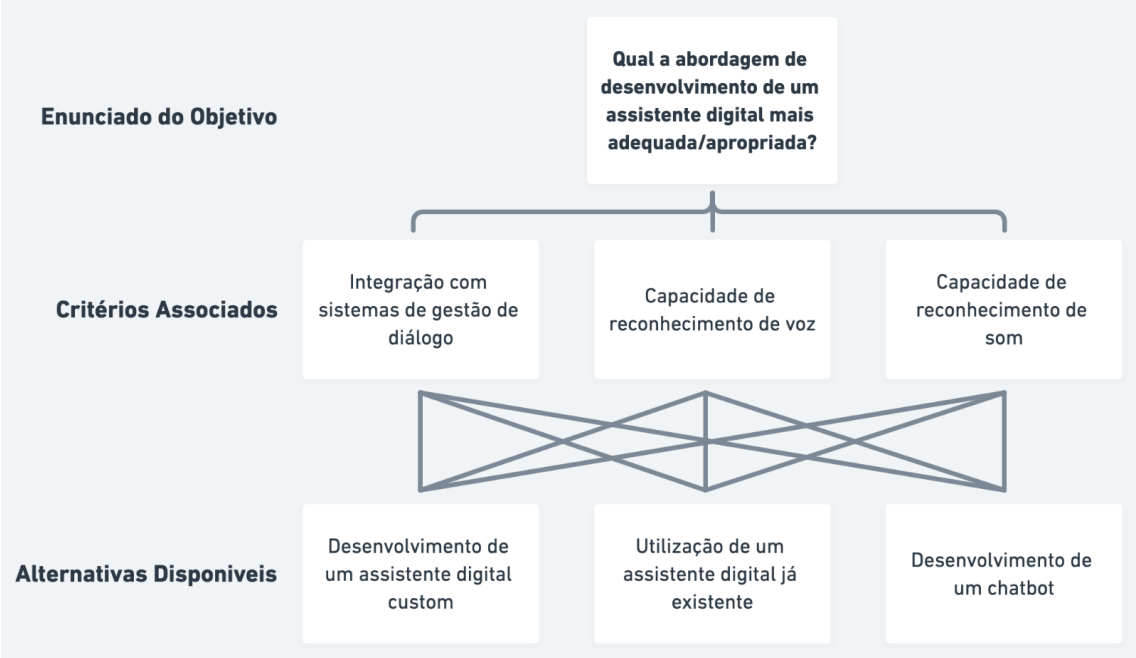

Figura 3.4: Estrutura hierárquica do método AHP [57]

A segunda fase do método AHP é a comparação de alternativas e critérios. Inicialmente, deve ser determinada a escala de valores a utilizar de modo a definir prioridades. Para isto, utilizou-se a escala de Saaty [1].

Após definição da escala a utilizar, passou-se ao desenvolvimento da matriz de comparação entre critérios, como se encontra representado na tabela 3.1.

Considerando:

- A: Integração com sistemas de gestão de diálogo;
- B: Capacidade de reconhecimento de voz;
- C: Capacidade de reconhecimento de som.

	A	B	C
A	1	0.33	0.2
B	3	1	0.33
C	5	3	1

Tabela 3.1: Matriz de comparação entre critérios

Analisando esta matriz de comparação, pode concluir-se que o critério relativo ao reconhecimento de voz é levemente mais importante e prioritário que a integração com sistemas de gestão de diálogo. Também se pode inferir que a capacidade de reconhecimento de som prevalece quando comparado com os restantes critérios, sendo que é levemente mais importante que a capacidade de reconhecimento de voz e fortemente mais importante que a integração com sistemas de gestão de diálogo.

[1] Escala proposta por *Thomas L. Saaty* aquando da apresentação do método multicritério AHP [57]

A terceira fase do método AHP passa por identificar a prioridade relativa de cada critério. Para que isto seja possível, torna-se necessário normalizar os valores da matriz de comparações representada na tabela 3.1 e obter o vetor de prioridades respetivo. Na tabela 3.2 encontra-se a matriz normalizada, seguida do vetor de prioridades respetivo, sendo este obtido pelo cálculo do valor da média aritmética por cada linha da matriz normalizada.

	A	B	C	Prioridade Relativa
A	$\frac{1}{9}$	$\frac{1}{13}$	$\frac{3}{23}$	0,11
B	$\frac{1}{3}$	$\frac{3}{13}$	$\frac{5}{23}$	0,26
C	$\frac{5}{9}$	$\frac{9}{13}$	$\frac{15}{23}$	0,63

Tabela 3.2: Matriz normalizada de comparação entre critérios e vetor de prioridades relativas

A quarta fase implica avaliar a consistência das prioridades relativas. Para isto, é necessário calcular a Razão de Consistência (RC) segundo a fórmula seguinte e verificar se este valor é menor que 0,1.

$$RC = \frac{IC}{IA}$$

sendo que IC corresponde ao índice de consistência e IA ao índice aleatório obtido através da tabela de índice aleatório presente na tabela 3.3 .

Dimensão da matriz	1	2	3	4	5	6	7	8	9	10
Índice de consistência	0,00	0,00	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49

Tabela 3.3: Matriz com valores aleatórios de consistência

Sendo que a dimensão da matriz em questão é três, o valor de IA será 0,58.

Para se calcular IC é necessário aplicar a fórmula que se segue:

$$IC = \frac{(\lambda_{max} - n)}{(n-1)}$$

Posto isto, multiplicando a matriz de comparação com o vetor de prioridades, obtém-se a seguinte matriz de valores:

A	0,32
B	0,79
C	1,94

Tabela 3.4: Matriz de valores obtida

Assim, torna-se possível o cálculo do valor de λ_{max} que é dado pela média dos valores obtidos na matriz dividido pelos valores do vetor de prioridades relativas.

$$\lambda_{max} = \frac{0{,}32/0{,}11 + 0{,}79/0{,}26 + 1{,}94/0{,}63}{3} = 3,04$$

Obtido o λ_{max}, passa-se ao cálculo do IC:

$$IC = \frac{(\lambda_{max} - n)}{(n-1)} = \frac{(3{,}04-3)}{(3-1)} = 0,02$$

Para finalizar, aplica-se a fórmula da razão de consistência:

$$RC = \frac{IC}{IA} = \frac{0,02}{0,58} = 0,03$$

Como 0,03 < 0,1, conclui-se que os valores das prioridades relativas estão consistentes.

A quinta fase do método AHP consiste no desenvolvimento das matrizes de comparação cruzando todas as alternativas existentes com cada critério de forma individual.

Considerando:

- A1: Desenvolvimento de um assistente digital *custom*;
- B1: Utilização de um assistente digital já existente (como colunas inteligentes);
- C1: Desenvolvimento de um *chatbot*.

	A1	B1	C1	Vetor Prioridade
A1	1	$\frac{1}{3}$	1	0,20
B1	3	1	3	0,60
C1	1	$\frac{1}{3}$	1	0,20

Tabela 3.5: Matriz de comparação para o critério A

	A1	B1	C1	Vetor Prioridade
A1	1	1	9	0,47
B1	1	1	9	0,47
C1	$\frac{1}{9}$	$\frac{1}{9}$	1	0,05

Tabela 3.6: Matriz de comparação para o critério B

	A1	B1	C1	Vetor Prioridade
A1	1	7	9	0,78
B1	$\frac{1}{7}$	1	3	0,15
C1	$\frac{1}{9}$	$\frac{1}{3}$	1	0,07

Tabela 3.7: Matriz de comparação para o critério C

Através da análise das tabelas 3.5, 3.6 e 3.7, é possível identificar a importância relativa de cada uma das alternativas. Assim, pode concluir-se que, no que diz respeito à integração com sistemas de gestão de diálogo, a solução B1 seria a mais adequada. Relativamente ao critério referente à capacidade de reconhecimento de voz, tanto a alternativa A1 como a B1 seriam adequadas. Por último, relativamente ao critério que explicita a capacidade de reconhecimento de som, a alternativa A1 seria a mais adequada, contendo uma priorização relativamente elevada quando comparada com B1 e C1.

A sexta fase consiste na obtenção da prioridade composta para as alternativas. Para isto, podem dispor-se os vetores de prioridades, provenientes das matrizes de comparação das alternativas por critérios, numa só matriz:

	A	B	C
A1	0,20	0.47	0.78
B1	0,60	0,47	0.15
C1	0,20	0,05	0,07

Tabela 3.8: Matriz de prioridades relativas das alternativas

Multiplicando esta matriz pelo vetor das prioridades relativas dos critérios, presente na tabela 3.2 obtém-se o seguinte vetor de prioridades compostas:

A1	**0,64**
B1	0,29
C1	0,08

Tabela 3.9: Prioridades compostas

Por último, a sétima fase consiste na escolha da alternativa mais apropriada. Assim, em função dos critérios definidos, é possível concluir que o desenvolvimento de um assistente digital *custom* será a abordagem de desenvolvimento de um assistente digital mais adequada.

3.2 Proposta de Valor

Uma proposta de valor serve um papel fulcral na conceção e no desenvolvimento de um novo produto, pois visa apresentar o produto e/ou os serviços oferecidos, quais os problemas que estes tencionam colmatar e, ainda, o que é oferecido ao cliente, de modo a que este atinja os resultados pretendidos. Relativamente ao cliente, os benefícios da obtenção do produto e as dificuldades que este vem colmatar devem, também, ser explícitas.

De modo a organizar as ideias envolvidas no processo de criação de valor, visando entender os problemas existentes para desenvolver um produto que seja realmente necessário, existe uma ferramenta utilizada mundialmente em *start-ups* e empresas de grande dimensão que se deve utilizar nesta fase de conceção - *Value proposition canvas* [58].

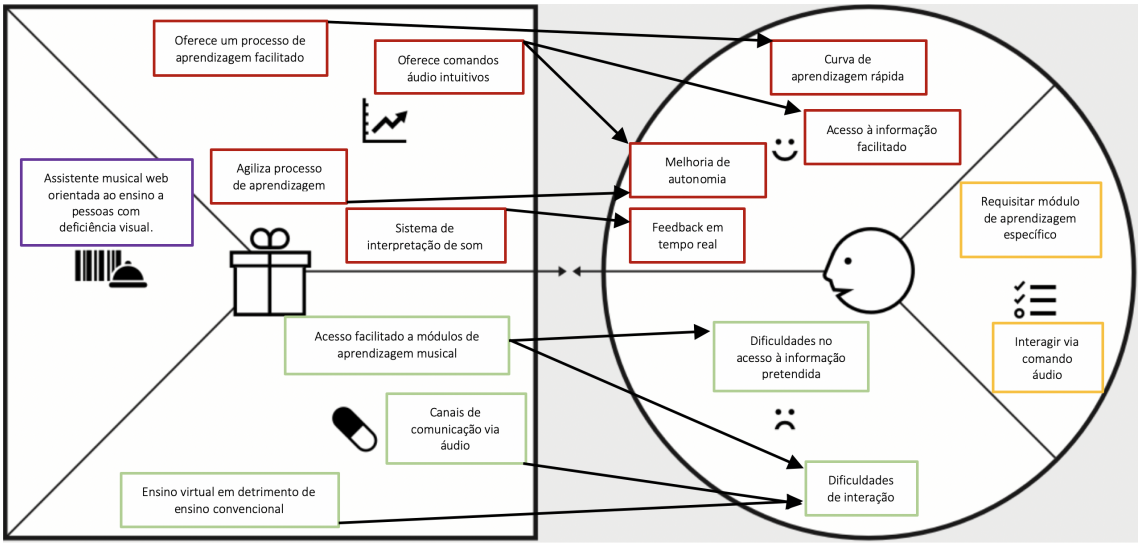

Figura 3.5: Value proposition canvas (adaptado de [58])

Através da análise do mapa apresentado na figura anterior (figura 3.5), verifica-se a existência de uma conexão clara entre o produto/serviços a desenvolver e o que é esperado pelo cliente, verificando-se em que medida o produto a ser desenvolvido simplifica a vida do cliente, criando benefícios.

A proposta de valor deste projeto consiste no desenvolvimento de uma plataforma digital que permita uma melhoria evidente no acesso à informação por parte das pessoas portadoras de deficiências visuais, existindo uma fácil interação com a mesma através de comandos via áudio, simplificando e agilizando a aprendizagem de módulos musicais.

3.3 Business Model Canvas

O *Business Model Canvas*, originalmente proposto por Alexander Osterwalder à data de 2004 na sua tese de doutoramento *The Business Model Ontology a Proposition in a Design ScienceApproach* [59] e posteriormente publicado no livro *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers* [60], é uma ferramenta de gestão estratégica, que permite desenvolver e esboçar um modelo de negócio novo ou existente. Segundo o autor, este modelo de negócio deve descrever a lógica de como uma organização cria, entrega e captura valor, sendo que esta descrição deve ser simples, relevante e facilmente perceptível, de modo a facilitar a sua descrição e comunicação.

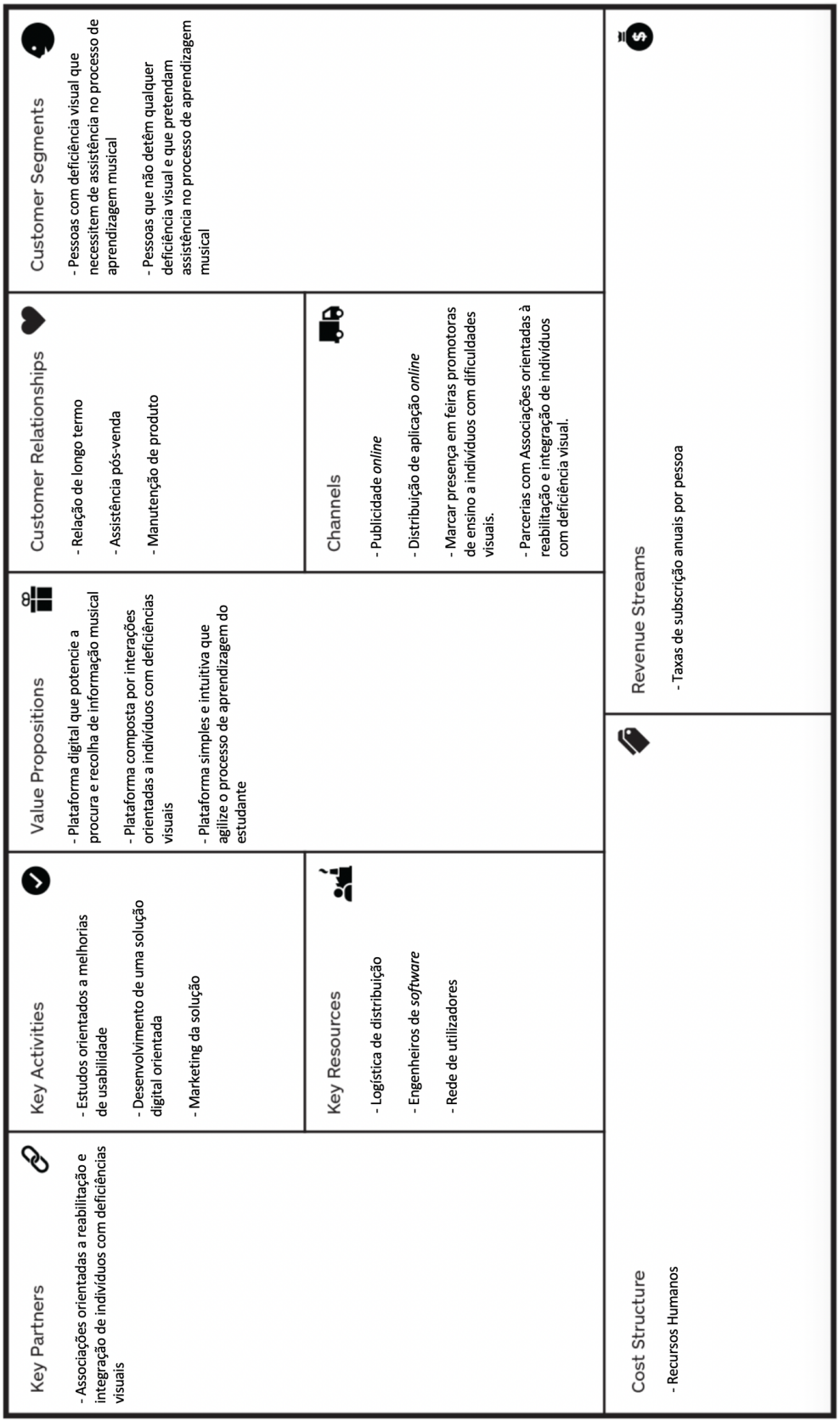

Figura 3.6: *Business Model Canvas* proposto

Na figura 3.6, apresenta-se um possível modelo de negocio a aplicar ao contexto do presente projeto. Através da análise do modelo desenvolvido, observa-se a existência de nove blocos distintos:

- Parcerias Chave (*Key Partners*) - Neste bloco encontram-se representados todos os parceiros externos à organização que contribuem para a concretização de um produto ou projeto. No âmbito deste projeto, as associações e organizações orientadas à reabilitação e integração de indivíduos com deficiências visuais serão os parceiros chave, uma vez que estas representam e defendem os direitos deste visando a construção de uma sociedade mais inclusiva [61].
- Atividades Chave (*Key Activities*) - As atividades chave descrevem quais as ações mais importantes que uma organização deve executar para que o seu modelo de negócio suceda. Para o projeto em questão, o desenvolvimento progressivo da plataforma assistente é a base de todas as ações, no entanto, para dar a conhecer o produto e o valor que dele advém, devem ser aplicadas atividades de *marketing* orientadas à solução. Por último, relembrando a grande importância da interação simples e intuitiva com o sistema, estudos orientados à melhoria da usabilidade serão também necessários.
- Recursos Chave (*Key Resources*) - Os recursos chave representam todo o tipo de recursos sem os quais se torna impossível a concretização do modelo de negócio proposto, permitindo a uma organização a criação e oferta de valor, alcançar mercados, obter fontes de retorno e manter relações com clientes. Assim, é relevante a existência de uma equipa de desenvolvimento composta por engenheiros de *software* para que se implemente a solução final. Uma vez que exista uma versão desta solução pronta a comercializar, importa distribui-la recorrendo à logística de distribuição. Por último, a existência de uma rede de utilizadores é fulcral porque o modelo de negócio só poderá existir com tal fonte de retorno.
- Proposta de Valor (*Value Proposition*) - Este bloco descreve o conjunto de produtos ou serviços que criam valor para um determinado segmento de clientes. Assim, uma proposta de valor expõe os fatores diferenciadores, visando ser a razão pela qual um cliente opta por uma organização em detrimento de outra. Para este projeto, conforme descrito na secção 3.2, a atividade diferenciadora da proposta de valor consiste no desenvolvimento de uma solução digital que auxilie na execução de processos de aprendizagem musical de forma simples e intuitiva.
- Relação com os Clientes (*Customer Relationships*) - O bloco da relação com clientes explicita , essencialmente, o tipo de relação que uma organização estabelece com o cliente. Assim, no presente projeto pretende-se uma relação de longo termo com o cliente considerando que se pretende uma melhoria na vida dos utilizadores. Como em qualquer solução digital, a constante evolução e manutenção, assim como a assistência pós-venda são também relações a implementar.
- Canais (*Channels*) - Neste bloco é descrita a forma como uma organização alcança um cliente, definindo-se canais de comunicação e venda e distribuição de um produto ou serviço. Para este projeto, existe um foco na publicidade *online*, para que se dê a conhecer a plataforma a indivíduos que, efetivamente, utilizem tecnologia. As parcerias com associações e a presença em feiras especializadas são, também, abordagens a adotar na divulgação do produto. Relativamente ao canal de distribuição e venda, pretende-se optar por uma distribuição *online*.

- Segmentos de Clientes (*Customer Segments*) - Neste bloco, definem-se os diferentes segmentos de clientes que se espera que obtenham um certo produto ou serviço. Neste projeto, apesar de o mesmo ser principalmente orientado a todos os indivíduos portadores de deficiências visuais que pretendam assistência na fase de aprendizagem musical, indivíduos que não sofram com qualquer deficiência visual podem, também, adquirir o produto e usufruir das suas funcionalidades.
- Estruturação de Custos (*Cost Structure*) - A estruturação de custos visa descrever todos os custos advindos da implementação do modelo de negócio. Para o modelo de negócio definido, os custos irão centrar-se nos custos de recursos humanos, seja a engenheiros responsáveis pelo desenvolvimento como a técnicos especializados para a distribuição.
- Fontes de Retorno (*Revenue Streams*) - Neste bloco representa-se o as fontes de retorno advindas de um produto ou serviço. No âmbito deste projeto, pretende-se que todo o retorno seja uma consequência das taxas de subscrições anuais que se pretendem aplicar a cada utilizador.

Capítulo 4

Análise

Neste capítulo, apresenta-se o modelo de domínio no qual se encontram representados os conceitos de negócio. Seguidamente, apresentam-se os requisitos funcionais, representados num diagrama de casos de uso, e os requisitos não funcionais do projeto, representados sob o modelo FURPS+.

4.1 Modelo de Domínio

Um modelo de domínio é uma representação visual estruturada entre conceitos interligados. Assim, representa classes conceptuais e rigorosamente organizadas que contêm interesse para o negócio [48].

De forma a melhorar a compreensão relativa ao domínio do sistema, apresenta-se, na figura 4.1, o modelo de domínio recorrendo à notação Unified Modeling Language (UML), no qual se encontram representadas as classes de domínio e respetivas cardinalidades.

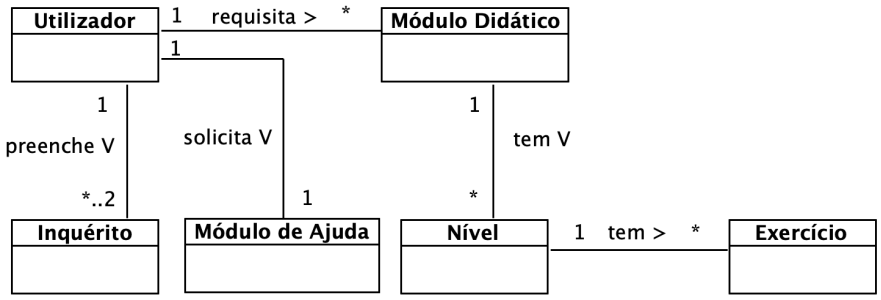

Figura 4.1: Modelo de domínio do assistente digital

Os conceitos representados no modelo de domínio da figura 4.1 encontram-se descritos de seguida:

- **Utilizador** - Representa a entidade Utilizador que irá dar uso ao sistema desenvolvido. Este tem a possibilidade de requisitar módulos didáticos, que por sua vez necessitam da escolha de um nível e da opção por um exercício. Para além disto, o utilizador pode preencher inquéritos e, ainda, solicitar ajuda.
- **Módulo Didático** - Entidade representativa dos módulos didáticos, ou instrumentos musicais, que o utilizador pode requisitar de modo a obter exercícios. Cada módulo didático será subdividido em níveis que, por sua vez, contêm exercícios.

- **Módulo de Ajuda** - Entidade representativa do módulo de ajuda que o utilizador pode solicitar. Este é responsável por indicar todas as funcionalidades possíveis ao utilizador.
- **Nível** - Entidade representativa do nível de dificuldade que agregará a existência de exercícios.
- **Exercício** - Entidade representativa do exercício requisitado pelo utilizador após requisitar módulo didático e escolher nível de dificuldade. O exercício selecionado poderá ser teórico ou prático.
- **Inquérito** - Entidade representativa dos inquéritos de satisfação e usabilidade que deverão existir de forma integrada no sistema. O utilizador poderá responder a todos ou a nenhum. Esta entidade existe como forma de método de avaliação de modo a avaliar indicadores de usabilidade e satisfação (c.f secção 7.5.2 e secção 7.5.3).

4.2 Requisitos Funcionais

De forma a representar os requisitos funcionais do sistema, recorrendo-se à utilização da notação UML, apresenta-se na figura 4.2 o diagrama de casos de uso.

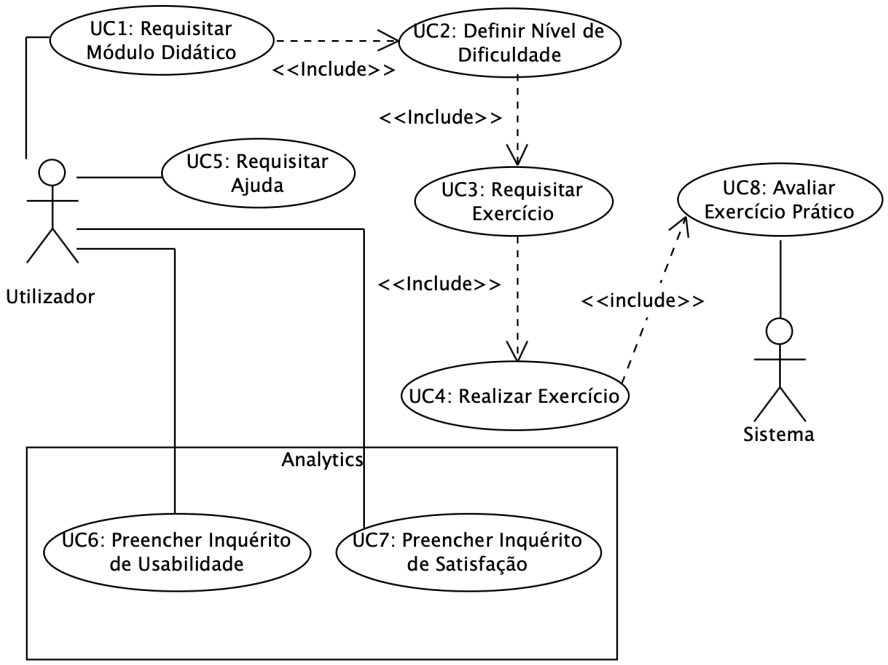

Figura 4.2: Diagrama de casos de uso do sistema

A análise do diagrama anterior permite concluir que irão existir dois atores no sistema. O Utilizador poderá iniciar vários casos de uso, entre estes, existem dois que se encontram agrupados num módulo de *Analytics* que corresponde ao módulo estatístico de avaliação da usabilidade e satisfação da aplicação. O sistema será responsável pela avaliação dos exercícios outrora executados pelo Utilizador. Os casos de uso encontram-se descritos de seguida através de uma breve descrição na qual se segue o fluxo normal de execução de cada um.

- **UC1 - Requisitar Módulo Didático**: O Utilizador inicia o processo de requisição de um módulo didático. O sistema reconhece o pedido, apresenta os módulos didáticos existentes e solicita escolha. O Utilizador escolhe o modulo didático, que é reconhecido pelo sistema que, por sua vez solicita requisição de nível de dificuldade.
- **UC2 - Definir Nível de Dificuldade**: O Utilizador inicia o processo de definição do nível de dificuldade. O sistema reconhece o pedido e indica quais os níveis de dificuldade existentes para o módulo didático definido. O Utilizador indica o nível de dificuldade, que é reconhecido pelo sistema que, por sua vez, solicita requisição de exercício.
- **UC3 - Requisitar Exercício**: O Utilizador inicia o processo de requisição de Exercício. O sistema reconhece o pedido e indica quais os exercícios existentes para o nível de dificuldade definido. O Utilizador define um exercício e o sistema solicita a sua realização.
- **UC4 - Realizar Exercício**: O Utilizador inicia o processo de realização de exercício. No caso de ser um exercício teórico, o conteúdo é apresentado ao utilizador e o caso de uso termina. No caso de ser um exercício prático, o sistema solicita dados advindos do instrumento, avalia os dados obtidos e oferece *feedback*.
- **UC5 - Requisitar Ajuda**: O Utilizador inicia o processo de requisição de ajuda. Após o Utilizador requisitar ajuda, o sistema reconhece o comando requisitado, calcula a ação pretendida e descreve todos os comandos áudio constituintes do sistema.
- **UC6 - Preencher Inquérito de Satisfação**: O Utilizador inicia o processo de preenchimento do inquérito de satisfação. O Sistema reconhece o pedido, e solicita resposta de cada pergunta individualmente até ao término da realização do inquérito.
- **UC7 - Preencher Inquérito de Usabilidade**: O Utilizador inicia o processo de preenchimento do inquérito de usabilidade. O Sistema reconhece o pedido, reproduz o guião com as tarefas a executar, e solicita resposta de cada pergunta, individualmente, referente a cada tarefa proposta, até ao término da realização do inquérito.
- **UC8 - Avaliar Exercício Prático**: O Sistema inicia o módulo de avaliação do exercício prático, analisa o som proveniente da realização do exercício prático com base no método apresentado na secção 6.2.3 e, por último, oferece *feedback* ao utilizador.

Importa referir que todas as interações entre o Utilizador e o Sistema são realizadas via áudio para que se melhore o processo de interação e a experiência de usabilidade dos indivíduos invisuais.

Uma vez descritos os casos de uso, considera-se relevante a apresentação da estruturação funcional dos casos de uso arquiteturalmente mais relevantes. Para esta representação, recorreu-se à utilização de diagramas *System Sequence Diagram (SSD)*, nos quais o componente representativo do Sistema é visto como uma "caixa preta", pois não se representa em detalhe as ações que por ele são executadas [62].

A complexidade inerente ao caso de uso **UC4 - Realizar Exercício**, devido a todas as pré-condições necessárias à sua execução, fazem com que este seja um dos casos de uso arquiteturalmente mais relevantes. Assim, de modo a contextualizar o caso de uso **UC4 - Realizar Exercício**, apresenta-se também o caso de uso **UC3 - Requisitar Exercício** que o antecipa. Nas figuras 4.3 e 4.4, apresenta-se o SSD referente a cada caso de uso.

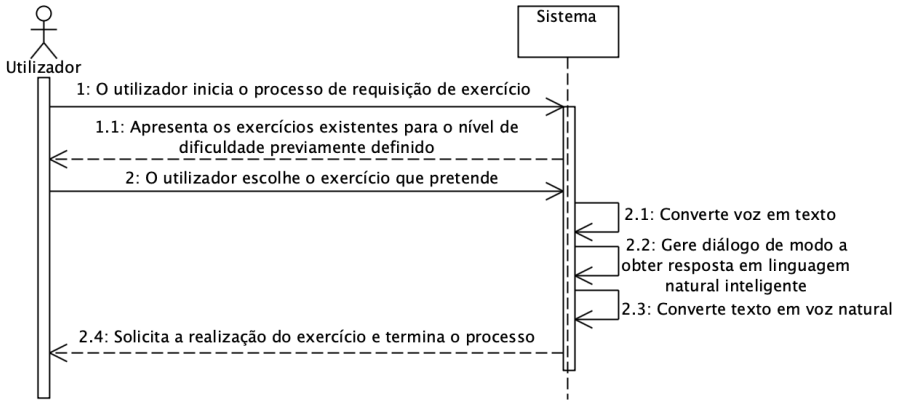

Figura 4.3: *SSD* do caso de uso Requisitar Exercício

No caso de uso representado na figura 4.3, o sistema é responsável por selecionar todos os exercícios existentes para o nível de dificuldade previamente definido, apresentando-os ao utilizador que, por sua vez, deverá escolher qual o exercício que posteriormente irá realizar. Posto isto, a análise da figura anterior ainda permite concluir que a cada interação entre o Utilizador e o Sistema, existe um conjunto de passos (c.f secção 2.3.1) que são executados automaticamente pelo sistema de modo a permitir o diálogo inteligente.

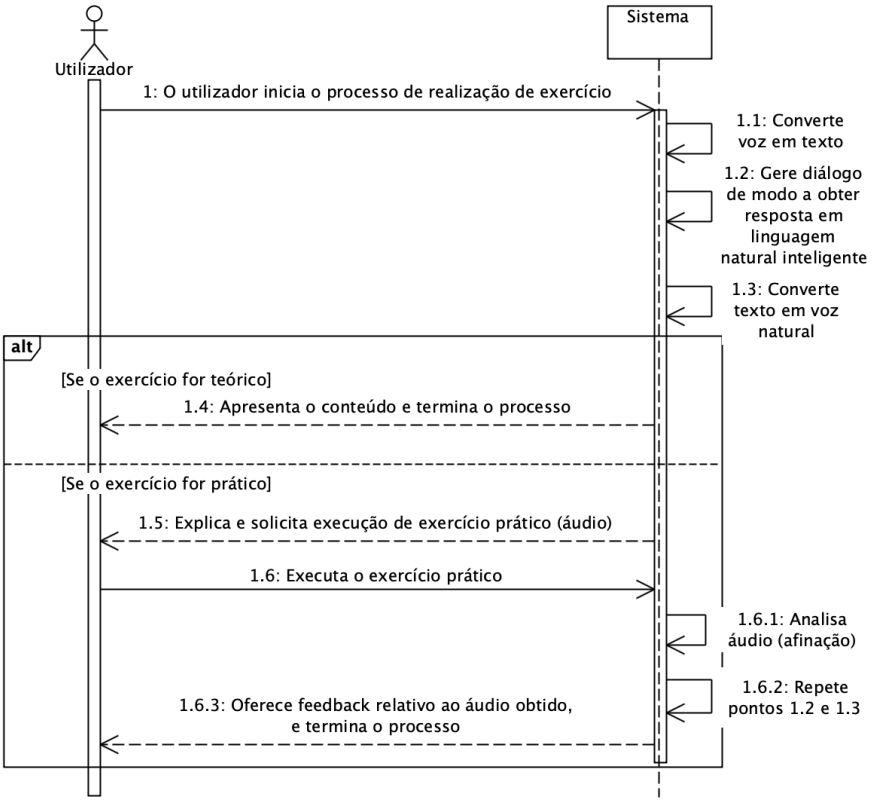

Figura 4.4: *SSD* do caso de uso Realizar Exercício

Uma vez executados os casos de uso necessários como pré-condições: UC1, UC2 e UC3, o

utilizador pode iniciar o processo de realização de exercício, que se encontra representado na figura 4.4. Assim, caso o exercício solicitado seja teórico, o sistema apresentará o módulo teórico e terminará, consecutivamente, o processo. No entanto, se o exercício solicitado for prático, o sistema irá requisitar a execução do exercício visando analisar o áudio recebido através de um componente de afinação - Lógica executada no caso de uso **UC8 - Avaliar Exercício Prático**. Este caso de uso tem como objetivo concluir se o áudio recebido se encontra em conformidade com os requisitos pretendidos, para que, deste modo, seja possível oferecer *feedback* ao utilizador relativo à execução do exercício.

Considerando a existência de um componente orientado a análise estatística, representado por *Analytics* na figura 4.2, considera-se relevante apresentar uma representação funciona de um dos casos de uso desse componente. Assim, na figura 4.5 apresenta-se o SSD do caso de uso **UC6 - Preencher Inquérito de Satisfação**, no qual se apresentam as interações entre o Utilizador e o Sistema.

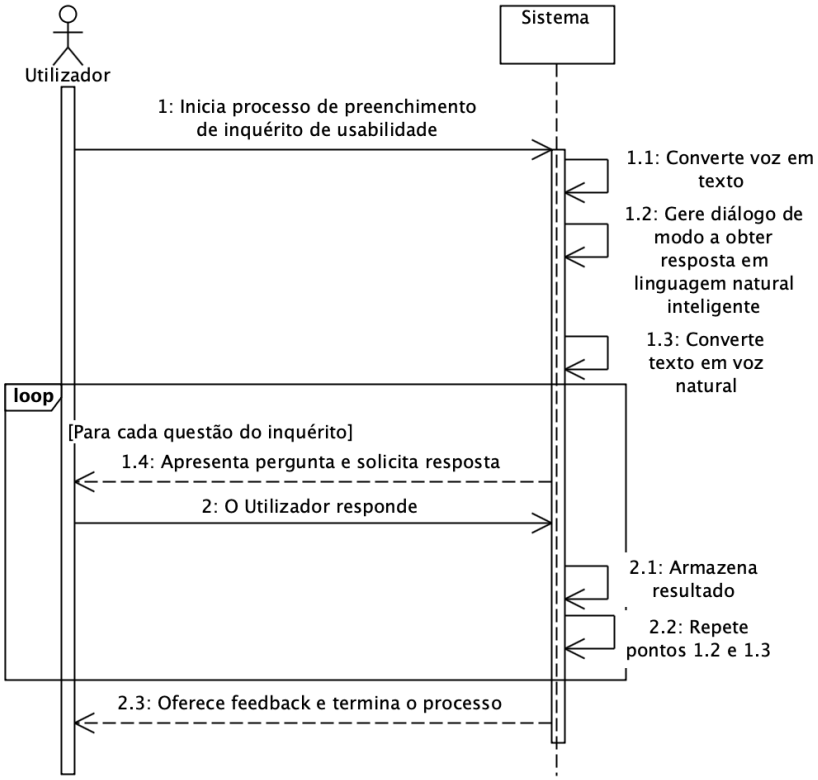

Figura 4.5: SSD do caso de uso Preencher Inquérito

A análise do diagrama da figura anterior permite concluir que, à semelhança do caso de uso representado na figura 4.4, cada interação implica a execução de vários módulos de modo a que seja possível dialogar com o sistema de forma inteligente. Para além disso, permite concluir que o sistemas de inquéritos é integrado no sistema, facilitando a interação por parte dos indivíduos invisuais.

4.3 Requisitos Não Funcionais

Os requisitos não funcionais são caracterizados por definirem a forma como um determinado sistema irá executar a sua funcionalidade. Assim, estes não estão diretamente relacionados com as funcionalidades de um sistema mas impõem-lhe algumas restrições.

De modo a representar os requisitos não funcionais, recorreu-se à utilização do modelo FURPS+ [48]. Este, é um acrónimo para *functionality* (funcionalidade), *usability* (usabilidade), *reliability* (confiabilidade), *performance* (desempenho) e *supportability* (suportabilidade) onde o + assume a função de identificar outros requisitos resultantes da evolução do modelo inicial FURPS, sendo estes: Restrições de design, restrições de implementação, restrições de interface e, por último, restrições físicas.

Nas subsecções seguintes, apresentam-se os requisitos não funcionais identificados.

4.3.1 Usabilidade

Considerando o problema (c.f secção 1.2) e os objetivos previamente definidos (c.f secção 1.3), a usabilidade é o requisito não funcional de maior relevância, pelo que a interação com o sistema tem de ser bastante intuitiva, permitindo a usabilidade do sistema por parte de pessoas invisuais através de comandos áudio.

4.3.2 Desempenho

O tempo de resposta deve ser adequado para não comprometer a experiência do utilizador, confirmando a experiência de um diálogo inteligente em tempo real.

4.3.3 Suportabilidade

O sistema deve ser compatível e adaptável a qualquer tipo de dispositivo móvel e fixo.

4.3.4 Restrições de Implementação

O sistema deve ser desenvolvido sob forma de uma aplicação *web*, devendo adotar normas de implementação que agilizem futura manutenção.

4.3.5 Restrições de Interface

A interface do sistema deve ser bastante intuitiva sendo que deve ser totalmente interativa via comandos áudio.

4.3.6 Restrições Físicas

O sistema deve ser adaptável a qualquer tipo de dispositivo, seja ele móvel ou fixo, desde que este contenha um componente de captura de voz, de emissão áudio e que seja, ainda, dotado de placa de rede que permita acesso à *internet*.

Capítulo 5

Design

O *design* de *software* representa o processo de criação de um artefacto de *software* que se propõe a atingir um certo objetivo, com base em certas restrições [63].

5.1 Abordagens Possíveis e Escolha de Arquitetura

Nesta secção apresentam-se duas abordagens de *design* alternativas, representadas através de diagramas de componentes recorrendo à notação UML, de modo a evidenciar as relações existentes entre os componentes do sistema [64]. Ambas as abordagens são apresentadas numa perspetiva de *design* de granularidade grossa, considerando que se pretendem representar as decisões arquiteturais sem entrar em detalhe sobre especificidades de cada componente. As tecnologias utilizadas para as duas abordagens são advindas da análise e comparação efetuada na secção 2.6.

5.1.1 Assistente Digital com Segregação de Responsabilidades

Na figura 5.1 encontra-se a representação dos componentes constituintes do sistema proposto com segregação de responsabilidades, assim como as interações entre os seus componentes.

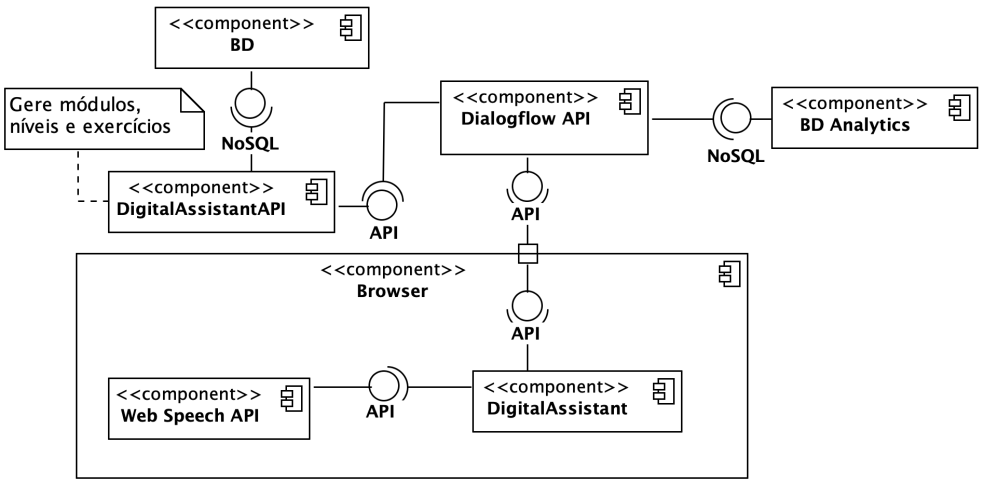

Figura 5.1: Diagrama de componentes do sistema com segregação de responsabilidades

A análise da figura anterior permite concluir a existência de vários componentes que se encontram descritos de seguida:

- *DigitalAssistant* - Aplicação *web* a desenvolver, que será responsável por agregar toda a lógica de integração, garantindo a sequência necessária ao funcionamento do assistente digital. Assim, esta comunica com a API do *DialogFlow*, de modo a gerir o discurso inteligente que pode até ser constituído por módulos, níveis, exercícios e inquéritos. Deste modo, é composta por um componente de *software* responsável pela análise de som advindo do instrumento musical e tem, ainda, a responsabilidade de garantir a existência de um discurso inteligente através da integração com as tecnologias de conversão de fala, gestão de diálogo, geração de linguagem e de conversão de texto.
- *Web Speech API* - Componente representativo da API que se encontra integrada no *browser*, e que é responsável pela conversão de voz em texto (*Speech-To-Text*) e de texto em fala natural (*Speech-To-Text*) (c.f secção 2.3.3.2).
- *Dialogflow* - Tecnologia responsável pela centralização da lógica relativa à gestão de diálogos e à geração de linguagem (c.f secção 2.3.3.5).
- *DigitalAssistantAPI* - Componente representativo da API a desenvolver, na qual irá residir a responsabilidade de alojamento e gestão de todos os módulos, níveis e exercícios existentes.
- BD - Componente representativo da base de dados que irá alojar os módulos, os níveis e os exercícios.
- BD *Analytics* - Componente representativo da base de dados que irá alojar os inquéritos e os seus resultados.

Assim, detona-se a existência de segregação de responsabilidades na medida em que a responsabilidade de apresentação de entidades de domínio se encontra abstraída pela utilização de uma API (*DigitalAssistantAPI*) que persiste os dados numa base de dados distinta da BD *Analytics*.

5.1.2 Assistente Digital com Centralização de API

Na figura 5.2 encontra-se a representação dos componentes constituintes do sistema com centralização de API, assim como as interações entre os seus componentes.

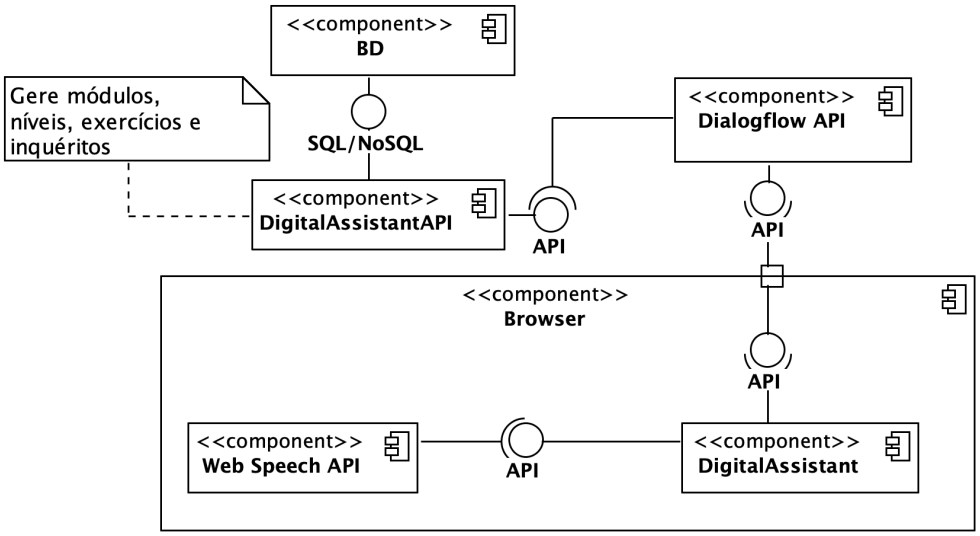

Figura 5.2: Diagrama de componentes do sistema com centralização de API

O diagrama representado na figura anterior contém grande parte dos componentes já apresentados na secção 5.1.1, no entanto, nesta abordagem alternativa de *design*, o componente **DigitalAssistantAPI** centraliza a lógica de gestão tanto das entidades de domínio como, também, dos inquéritos. Consequentemente, o componente **BD** que representa a base de dados, não só persistirá os módulos, os níveis e os exercícios mas, também, os inquéritos e os seus resultados.

5.1.3 Avaliação e Escolha de Solução de Design

A centralização da gestão das entidades de domínio e dos inquéritos numa só API e base de dados (c.f secção 5.1.2) pode ser interessante na medida em que facilita a logística de alojamento dos serviços. No entanto, esta centralização implica a utilização da mesma tecnologia de persistência de dados tanto ao nível das entidades de domínio como ao nível dos inquéritos de usabilidade e satisfação, e implica uma dificuldade acrescida aquando de manutenção futura.

No que diz respeito à solução que implementa a segregação de responsabilidades, tal não acontece. Nesta, apesar da logística de alojamento dos serviços distribuídos ser mais complexa, o ganho obtido relativo à escalabilidade e à manutenção futura prevalece. A criação de uma API cuja responsabilidade é unicamente gerir os resultados dos inquéritos facilita a integração com módulos de análise estatística no futuro e potencia, ainda, a utilização de tecnologias de persistência distintas que permitam uma maior escalabilidade e *performance*. Esta segregação de responsabilidades encontra-se em conformidade com o principio de *design Separation Of Concerns (SoC)*[1], que visa subdividir a complexidade de um problema em várias camadas de fácil gestão e manutenção, reduzindo o risco advindo de futuras mudanças funcionais, considerando que estas se encontram isoladas a um único componente [65]. Assim considerando-se a abordagem de segregação de responsabilidades a mais adequada (c.f secção 5.1.1), apresenta-se, de seguida, o *design* detalhado com base nessa arquitetura.

[1]Principio de *design* de *software* que defende a existência de componentes distintos para funcionalidades distintas, sobre os quais exista o mínimo acoplamento possível [65].

5.2 Design Detalhado

O design detalhado é o conceito de design no qual se pormenoriza a solução a desenvolver. O objetivo principal deste é o desenho ao detalhe de uma solução segundo a arquitetura definida no design arquitetural, conforme definido na secção 5.1.1, promovendo classes de domínio a classes de software, diminuindo a tendência a eventuais mudanças, apoiando a arquitetura sempre de forma a satisfazer os requisitos propostos [66].

5.2.1 Diagrama de Componentes Detalhado do Sistema

Uma vez definida a abordagem arquitetural a seguir, conforme analisado na secção 5.1.3, torna-se relevante detalhar a estrutura a implementar, passando de uma granularidade grossa a uma granularidade fina. Deste modo, na figura 5.3 encontram-se representados os componentes constituintes do sistema, assim como as suas interligações e dependências.

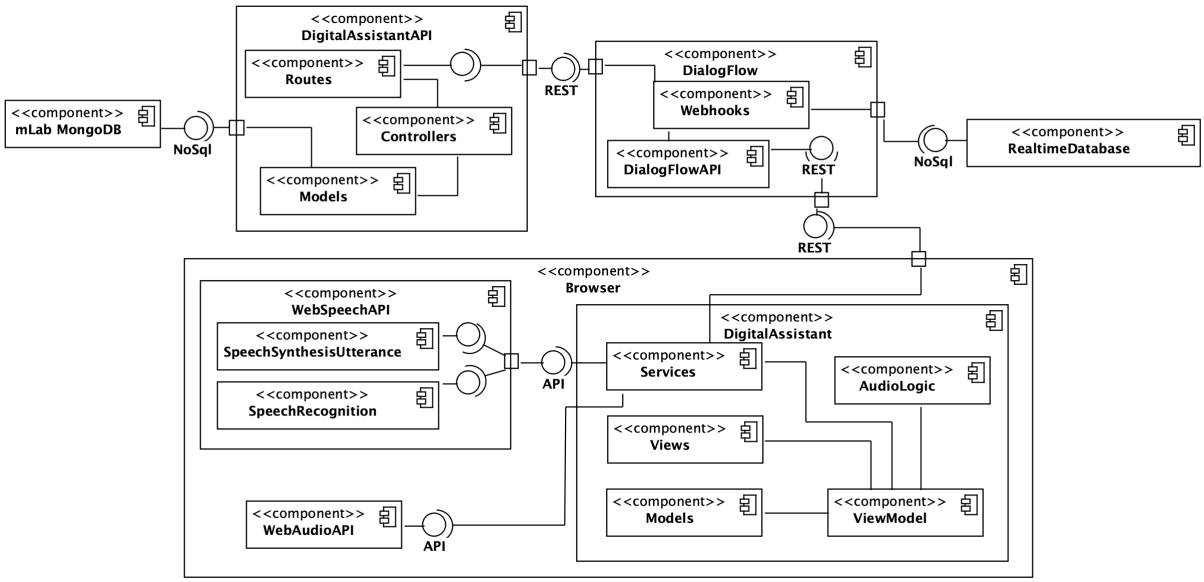

Figura 5.3: Diagrama de componentes existentes no sistema

A figura 5.3 ilustra a existência de componentes já analisados na secção 5.1.1, nomeadamente: *WebSpeechAPI*, *DigitalAssistant*, *DigitalAssistantAPI* e *DialogFlowAPi*. No entanto, dado o aumento do nível de detalhe, e com base nas decisões tecnológicas efetuadas na secção 2.6, passa-se a explicar a relevância inerente a cada componente interno que os constitui:

- **WebSpeechAPI**
  - *SpeechRecognition* - Interface constituinte da API *Web Speech API*, que providencia a possibilidade de reconhecimento de texto através de um input áudio.
  - *SpeechSynthesisUtterance* - Interface constituinte da API *Web Speech API*, responsável pela síntese de fala inteligível.
- **WebAudioAPI** - Componente representativo da API que se encontra integrada no *browser*, que disponibiliza um sistema versátil e completo, no que diz respeito à infraestrutura de controlo de áudio na *web*.

- **DigitalAssistant**
  - *Services* - Artefacto representativo da camada de serviços existente na aplicação *web DigitalAssistant*, responsável por gerir todas as interações existentes entre o componente *DigitalAssistant* e serviços externos.
  - *Models* - Artefacto representativo da camada de modelos existente na aplicação *web* em questão, responsável pelo alojamento da estrutura das entidades de domínio existentes, conforme descritas na secção 4.1.
  - *Views* - Artefacto representativo da camada de vistas existente na aplicação *web* em questão, responsável pela apresentação de conteúdo ao utilizador. Apesar de ser uma aplicação cuja interação é executada via áudio, a existência da camada de *Views* é necessária devido a implicações técnicas da *framework* utilizada.
  - *ViewModel* - Artefacto representativo da camada de modelos existente na aplicação *web* em questão, responsável pela interação entre as Vistas e os Modelos.
  - *AudioLogic* - Componente existente na aplicação, responsável pela análise de som proveniente do instrumento musical, através do qual se dará *feedback* ao utilizador após execução de um exercício prático.
- **DigitalAssistantAPI**
  - *Routes* - Artefacto representativo da camada de rotas existentes na API, que será responsável por orientar os pedidos recebidos, associando cada um ao seu *Controller* respetivo.
  - *Controllers* - Artefacto representativo da camada de controladores, na qual existem funções responsáveis pela lógica associada a cada pedido, persistindo e retornando dados da base de dados.
  - *Models* - Artefacto representativo da camada de modelos que contém a estrutura de cada entidade de domínio que será persistida na base de dados não relacional *MongoDB*.
- **MongoDB** - Componente representativo da base de dados não relacional a implementar, que será consumida pela API *DigitalAssistantAPI*.
- **DialogFlow**
  - *DialogFlow API* - Artefacto representativo da API disponibilizada pela tecnologia *DialogFlow*, servindo de ponto de entrada para pedidos solicitados por aplicações externas.
  - *Webhooks* - Artefacto representativo da tecnologia disponibilizada pela plataforma *DialogFlow* que permite extensão ou alteração de comportamento previamente existente, através da execução de pedidos customizados a serviços externos, consumindo dados externos e retornando automaticamente quando certos critérios são preenchidos [67].
- **RealTimeDatabase** - Componente representativo da base de dados não relacional exposta por um serviço externo já existente, responsável por alojar toda a informação relativa ao módulo de inquéritos.

A título conclusivo, é possível inferir pelo diagrama da figura 5.3 o fluxo de interligações existente, não só entre os componentes principais da aplicação, mas também entre os seus componentes internos. Conclui-se, portanto, que a aplicação *web DigitalAssistant*, disponível através de um *Browser*, interage com APIs *built-in* no *Browser* e, também, com a plataforma do *DialogFlow* através da camada de serviços. O *DialogFlow*, por sua vez, interage com a base de dados na qual serão persistidas todas as informações relativas a inquéritos e com a aplicação *DigitalAssistantAPI*. Esta é responsável pela disponibilização de rotas que satisfaçam os requisitos funcionais e, ainda, pela persistência das entidades de domínio.

5.2.2 Diagrama de Implantação do Sistema

De forma a garantir que existe consistência nas dependências físicas do sistema e para que exista uma gestão das máquinas a serem utilizadas e do número de componentes alojados em cada uma, para que se evite uma possível utilização desnecessária de recursos, desenhou-se o diagrama apresentado na figura 5.4, no qual se representa a estrutura física do sistema a desenvolver.

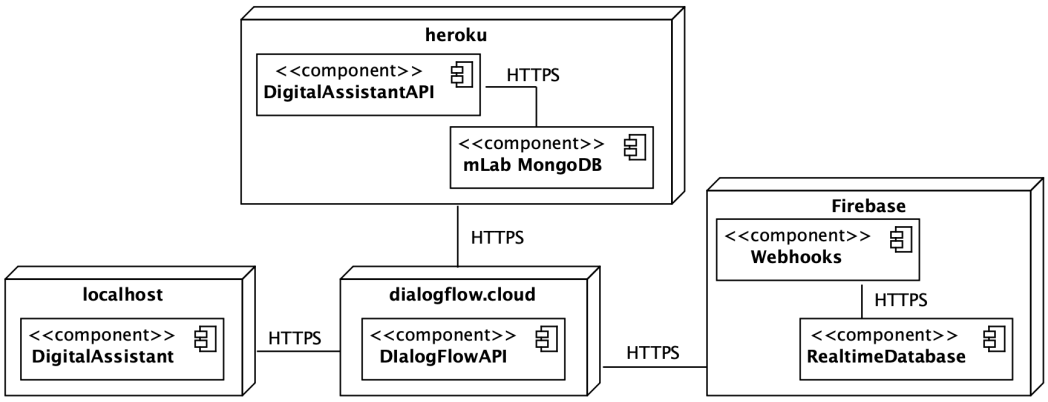

Figura 5.4: Diagrama de implantação do sistema

Os componentes físicos responsáveis por alojar o sistema são os seguintes:

- *heroku* - Sistema *Platform as a Service (PaaS)* [2], que permite o alojamento de aplicações na *cloud* de forma gratuita. Assim, a implantação da API *DigitalAssistantAPI* será feita utilizando este serviço. Uma vez que a aplicação em questão terá de persistir as entidades de domínio (c.f 5.1.1), torna-se necessária a utilização de um serviço *Database as a Service (DBaaS)* [3] que substitua a base de dados outrora necessária aquando de um cenário local, sendo este o componente que se encontra representado como *mLab MongoDB* [70]. A implantação da API *DigitalAssistantAPI* foi de índole obrigatória uma vez que a sua implantação é um pré-requisito para que esta possa ser consumida pela plataforma *Dialogflow*.

[2]Modelo de computação *cloud* que detém uma infraestrutura na qual se hospeda o hardware e o software. Como resultado, o *PaaS* agiliza o processo de implantação de serviços uma vez que abstrai o processo de instalação de *hardware* e *software* internos para desenvolver ou executar um novo serviço [68].

[3]Serviço de computação *cloud* que permite o acesso e utilização de um sistema de base de dados abstraindo tanto o processo de configuração de hardware e software da base de dados como, também, a sua gestão e manutenção [69].

- *localhost* - Como não se tornou necessária a sua implantação em serviços *online*, *localhost* representa o sistema local no qual se encontra implantada a aplicação *web DigitalAssistant*.
- *dialogflow.cloud* - Domínio no qual se encontra implantado o sistema de *DialogFlow*, assim como a API que disponibiliza.
- *Firebase* - Plataforma de desenvolvimento de aplicações móveis e *web* que disponibiliza serviços variados de forma a auxiliar o desenvolvimento. Os *Webhooks* (c.f secção 6.2.1.3) e a *RealtimeDatabase* (c.f 5.2.4) são dois dos sistemas disponibilizados pelo *Firebase* a utilizar no projeto em questão.

A estrutura de implantação resultante, demonstra uma modularidade no sistema, pois pretende-se agilizar as etapas posteriores tanto de manutenção como de atualização tecnológica, pelo que, assim, se torna possível a modificação ou substituição individual de cada um dos componentes, sem que exista algum impedimento que reflita uma reestruturação do sistema.

5.2.3 Vista de Cenários do Sistema

A Vista de Cenários visa representar as relações entre os componentes do sistema, sendo representada através de um diagrama de sequência UML, que retrata a interação funcional existente entre todos os componentes constituintes do sistema.

Na figura 5.5, encontra-se uma representação das interações existentes entre os componentes do sistema relativamente ao caso de uso UC3 - Requisitar Exercício, podendo, ainda, identificar-se o fluxo de processamento existente. O utilizador interage com o sistema via componente *View* cuja lógica é gerida no componente *ViewModel*. De modo a identificar a fala do utilizador, recorre-se à utilização da *WebSpeechAPI(Speech Recognition)*. Após identificação do comando proferido pelo Utilizador, o sistema executa uma ação em conformidade com o pedido, que irá selecionar todos os exercícios existentes para o nível de dificuldade previamente selecionado, de modo a que estes sejam apresentados ao Utilizador. De modo a representar esta ação, recorreu-se à utilização de *Interaction Use*[4], podendo verificar-se esta interação no diagrama representado na figura 5.6.

[4] Conceito inerente à utilização de diagramas de sequência UML, que representa uma interação do sistema que, uma vez utilizada, permite a modulação e reutilização de diagramas, simplificando os diagramas mais complexos, tornando-os mais legíveis [71]

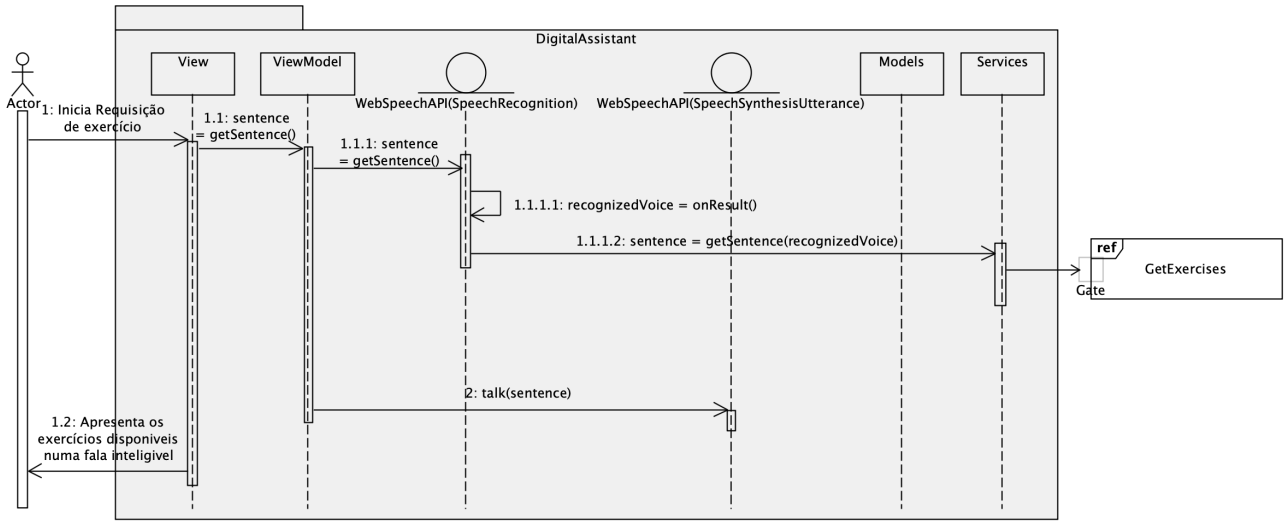

Figura 5.5: Diagrama representativo do processo de interação do utilizador com o assistente digital

No bloco de interação representado na figura seguinte - figura 5.6 - encontra-se representada a ação de seleção de exercícios. Nesta, verifica-se a dependência existente entre a plataforma *DialogFlow* e a API *DigitalAssistantAPI*, uma vez que o *DialogFlow* tem a responsabilidade de consumir os dados persistidos na base de dados gerida pela *DigitalAssistantAPI*. Para isto o agente do *DialogFlow* utiliza um *WebHook* (c.f secção 6.2.1.3) que irá comunicar com a camada de *Routes* que associará o *Controller* respetivo ao pedido efetuado e este, por último, comunica com a camada de *Models* na qual se encontra a estrutura dos modelos a persistir ou a consumir, conforme explicado na secção 5.2.1. Uma vez obtidos os exercícios, recorre-se à API *WebSpeechAPI(Speech SynthesisUtterance)* de modo a sintetizar fala que será retornada ao utilizador.

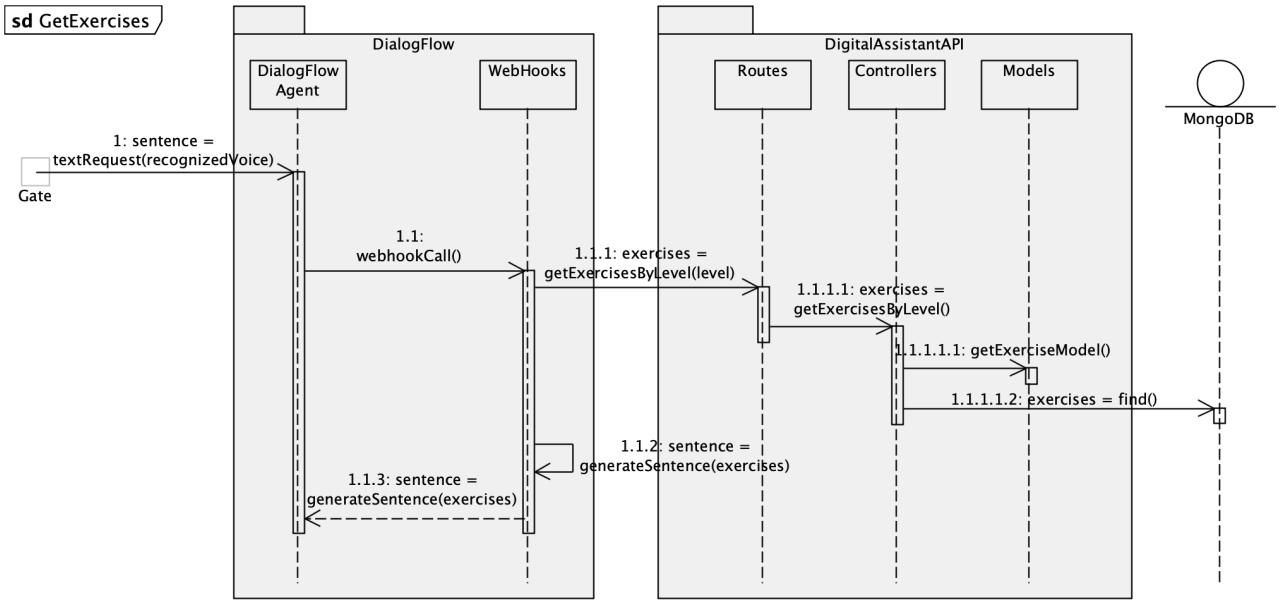

Figura 5.6: Diagrama representativo do processo de interação do utilizador com o assistente digital

5.2.4 Modelação de Dados

Como referido na secção 5.2.1, existem duas bases de dados a desenvolver sendo que a primeira, identificada por *MongoDB* será responsável pelo alojamento estruturado das entidades de negócio (c.f 4.1) e a segunda, identificada por *RealtimeDatabase*, será responsável por alojar os inquéritos e as suas respostas.

O *MongoDB* é uma base de dados não relacional orientada a documentos que armazena valores em formato *JSON*, isto é, ao contrário das bases de dados relacionais, em *MongoDB* não existe necessidade de criar tabelas com relações entre si, podendo um documento conter toda a informação necessária. Uma vez agrupados, estes documentos dão lugar a coleções que formam a base de dados [72].

De forma a persistir as entidades de negócio identificadas na secção 4.1, e considerando as possibilidade auferidas pela utilização desta tecnologia, optou-se por agregar toda a estrutura de persistência das entidades num só documento que terá inicio no objeto Módulo Didático. Deste modo, apresenta-se no Excerto de Código 5.1 uma representação em formato *JSON* do documento que contém a estrutura que persistirá toda a informação.

```
{
    "name" : "Guitarra",
    "theoreticalIntroduction" : "...",
    "levels" : [
        {
            "name" : "Iniciante",
            "exercises" : [
                {
                    "name" : "inqueritoUsabilidadeCordas soltas",
                    "howTo" : "..."
                    "description" : "..."
                },
                {
                    (...)
                }
            ]
        },
        {
           (...)
        }
    ]
}
```

Excerto de Código 5.1: Representação JSON da estrutura do documento do Módulo Didático

Apesar de ser uma base de dados não relacional, tal não invalida a viabilidade da utilização de valores simples como números, *strings* e datas e, ainda, de vetores de valores ou objetos. Assim, pela análise do Excerto de Código 5.1, conclui-se que a estruturação da informação se inicia num Módulo Didático, cujo nome é Guitarra, que contém um vetor de objetos representativos dos níveis de dificuldade que, por sua vez, contém também um vetor de objetos que representam os exercícios. Em suma, a base de dados *MongoDB* será constituída por uma coleção de documentos de Módulos Didáticos nos quais existirá toda a informação a persistir.

A *RealtimeDatabase*, à semelhança do *MongoDB*, é também uma base de dados não relacional [73] e surge neste projeto devido à necessidade de existir forma de gerir os inquéritos

no próprio sistema. Deste modo, tanto para os inquéritos de satisfação como para os inquéritos de usabilidade, pretende-se uma estrutura como a do documento em formato *JSON* apresentado no Excerto de Código 5.2, no qual se define o nome do inquérito, a escala de cada questão e as próprias questões. A propriedade *script* é referente ao guião utilizado nos inquéritos e apenas será preenchida para inquéritos de usabilidade uma vez que não se pretende que inquéritos de satisfação sigam um guião com tarefas previamente planeadas.

```
{
    "id" : "...",
    "name" : "Inquerito  Satisfacao",
    "scale" : "...",
    "script" : "...",
    "questions" : [
        {
            "questionId" : "...",
            "questionDescription" : "..."
        },
        {
            (...)
        }
    ]
}
```

Excerto de Código 5.2: Representação JSON da estrutura do documento dos inquéritos

Uma vez definida a estrutura de armazenamento das questões, torna-se necessário definir uma estrutura de persistência para os resultados obtidos. Assim, no Excerto de Código 5.3, apresenta-se a estrutura dos documentos em formato JSON que associam o identificador da pergunta à resposta obtida. Importa ainda referir que nenhuma informação do utilizador é guardada de forma a que a base de dados esteja em conformidade com o Regulamento Geral sobre a Proteção de Dados (RGPD).

```
{
    "id": "...",
    "questionId" : "...",
    "questionResult" : "..."
}
```

Excerto de Código 5.3: Representação JSON da estrutura do documento das respostas dos inquéritos

Em síntese, a *RealtimeDatabase* irá conter dois documentos representativos dos inquéritos existentes (c.f secção 4.1) e tantos documentos quantas respostas existirem a esses mesmos inquéritos.

5.2.5 Comandos de Voz

Considerando os requisitos não funcionais impostos na secção 4.3, o sistema a desenvolver seguirá uma abordagem com base em sistemas conversacionais de pergunta e resposta pelo que a cada ação do utilizador, irá seguir-se uma resposta por parte do sistema. Assim sendo, apresenta-se na figura seguinte um diagrama representativo da interação homem-máquina do assistente digital a desenvolver.

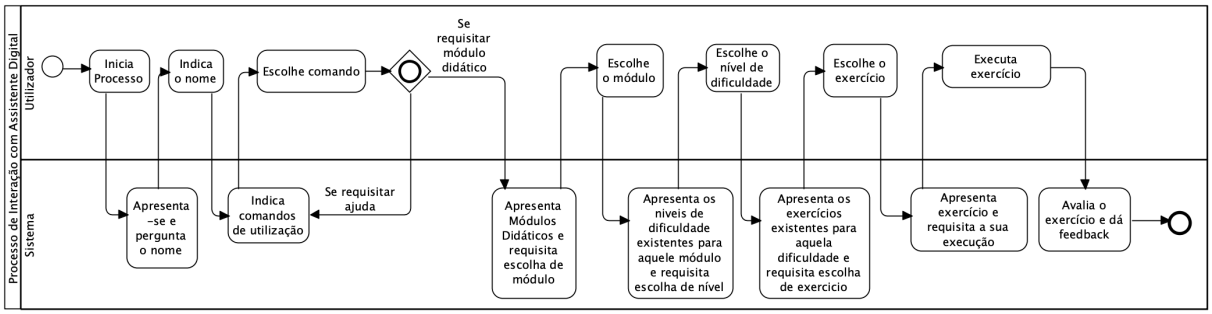

Figura 5.7: Diagrama representativo do processo de interação do utilizador com o assistente digital

Através da análise da figura 5.7 verifica-se o fluxo normal do sistema desde que se inicia o processo até que se avalia um exercício e se oferece *feedback*. No entanto, apesar do sistema se encontrar representado como tendo um inicio e fim quando se oferece *feedback*, importa referir que não se pretende que esse comportamento seja linear, pelo que é possível que o utilizador, uma vez que tenha indicado a sua identificação ao sistema, inicie o fluxo conversacional numa ação que não seja necessariamente a que se encontra representada no inicio do diagrama apresentado, assim como é possível continuar a utilização do sistema após receber *feedback* de um exercício. Deste modo, pretende-se melhorar a experiência do utilizador, inferindo a ideia de que existe inteligência no sistema com o qual está a interagir.

A lógica de apresentação e preenchimento de inquéritos não se encontra representada porque é idêntica à lógica apresentada para o fluxo normal do resto do sistema.

Capítulo 6

Implementação

Neste capitulo é abordada a fase da disciplina de implementação deste projeto. Assim, abordam-se, de seguida, os requisitos arquitecturalmente relevantes, e apresentam-se, como complemento, alguns fragmentos do código produzido.

6.1 Convenções Adotadas

A primeira fase da implementação foi a definição de algumas diretrizes a adotar de forma a existir uma escrita de código que o tornasse legível e formal, potenciando modificações futuras e que fosse provada como sendo uma boa forma de implementação. Deste modo, adotaram-se algumas medidas definidas pela *Sun Microsystems* [74]:

- Comentários de Documentação - Comentários utilizados para que se descrevam componentes, interfaces, construtores, métodos e, até, propriedades. Têm como principal utilidade dar a conhecer a especificação da implementação sem que seja necessário o conhecimento do código fonte.
- Comentários de Implementação - Comentários utilizados para que se explicite uma parte específica da implementação. Apesar da existência de vários tipos de comentários de implementação, pretende-se que a utilização destes comentários seja restringida a comentários de linha única (que explicitam o comportamento na linha exatamente anterior) e comentários em bloco (que seguem a lógica dos comentários anteriormente referidos mas permitem múltiplas linhas).
- Convenções Nominais - Convenções adotadas para a definição de métodos, variáveis e constantes.
  - Métodos - A constituição dos nomes dos métodos deve ser ou um verbo ou inicializada por um verbo, e a sua capitalização deve seguir a regra *LowerCamelCase* (na qual todas as palavras são capitalizadas com exceção da primeira, que se mantém com letra minúscula).
  - Variáveis - Os nomes dados às variáveis devem ser curtos e significativos e a sua capitalização deve seguir a regra *LowerCamelCase*.
  - Constantes - O nome dado a constantes deve conter apenas caracteres capitalizados e, no caso de conter várias palavras, devem ser divididas por *underscores*.
- Organização de Ficheiros - Todos os ficheiros criados devem conter menos de duas mil linhas de código, pelo que se considera esse o limite de forma a manter a legibilidade e facilidade de manutenção de um ficheiro.

De modo a facilitar a leitura, a navegação e a edição dos ficheiros, definiu-se uma ordem especifica para os membros de uma classe, assim sendo, propriedades privadas devem anteceder propriedades publicas, seguidas dos métodos privados que antecedem os métodos públicos. A par com estas convenções, ainda que menos relevantes, impuseram-se outras regras de estilo como limite máximo de caracteres por linha e restrições da existência de espaços em branco em cada inicio e fim de linha, de modo a uniformizar o código, melhorando a sua legibilidade.

6.2 Digital Assistant

Nesta secção aborda-se o processo de implementação do projeto *Digital Assistant*, considerando todas as fases constituintes de um assistente digital, e ainda, a integração com o módulo de análise de som desenvolvido.

6.2.1 Fases do Assistente Digital

Considerando as várias fases constituintes de um assistente digital, conforme enunciadas na secção 2.3.1, apresenta-se nesta secção uma explicação detalhada da disciplina de implementação, através da utilização das tecnologias analisadas e definidas na secção 2.6.

6.2.1.1 WebSpeechAPI (Speech Recognition)

A primeira fase a implementar aquando da construção de um assistente digital é a fase do reconhecimento de voz. Conforme referido na secção 2.3.3, a *WebSpeechAPI* providencia duas áreas funcionais, sendo que a área *Speech Recognition* aborda o problema do reconhecimento de voz e apresenta uma solução, pelo que se optou pela sua utilização.

No Excerto de Código 6.1, apresenta-se a implementação da função *speechRecognition()*, que será a responsável pela integração com a *WebSpeechAPI* no projeto *DigitalAssistant*, considerando-se o ponto inicial no processo de utilização do assistente digital.

```
1  export interface IWindow extends Window {
2      webkitSpeechRecognition: any;
3  }
4  const { webkitSpeechRecognition }: IWindow = <IWindow>window;
5  const PT_LANGUAGE = 'pt-PT';
6
7  public speechRecognition(): void {
8      if ('webkitSpeechRecognition' in window) {
9          this.vSearch = new webkitSpeechRecognition();
10         this.vSearch.continuous = false;
11         this.vSearch.interimresults = true;
12         this.vSearch.lang = PT_LANGUAGE;
13         this.vSearch.start();
14         this.vSearch.onspeechend = (e) => {
15            this.vSearch.start();
16         };
17         this.vSearch.onerror = function (e) {
18             this.vSearch.stop();
19         };
20         this.vSearch.onresult = (e) => {
21             this.result = e.results[e.results.length - 1][0].transcript;
22             this.getResponse(this.result);
23         };
24     }
25 }
```

Excerto de Código 6.1: Função *speechRecognition* que integra a *WebSpeechAPI*

A análise do Excerto de Código 6.1 permite concluir a existência de vários parâmetros e funções expostos pela *WebSpeechAPI*, que alteram o modo de funcionamento da API. Deste modo, explicam-se de seguida, as funções e os parâmetros utilizados.

- *start()* - Função exposta pela API, presente na linha 13 e 15, que permite iniciar o processo de reconhecimento de voz.
- *stop()* - Função exposta pela API, presente na linha 18, que permite terminar o processo de reconhecimento de voz.
- *continuous* - Parâmetro exposto pela API, presente na linha 10, que permite controlar a identificação continua, ou não, de resultados. Para o caso do *DigitalAssistant*, pretende-se que a API retorne um resultado caso identifique fala, daí a utilização do parâmetro com o valor *false*.
- *interimresults* - Parâmetro exposto pela API, presente na linha 11, que permite controlar a identificação de resultados intermédios. Para o caso específico, pretendem-se apenas os resultados finais (ou seja, no final de cada comando de voz), pelo que se utiliza este parâmetro com o valor *true*.
- *lang* - Parâmetro exposto pela API, presente na linha 12, que permite a definição da linguagem pela qual se pretende reconhecer os comandos de voz. Para o caso específico utiliza-se a linguagem portuguesa.
- *onspeechend* - Parâmetro exposto pela API, presente na linha 14, que executa quando os comandos de voz reconhecidos pelo sistema de reconhecimento deixam de ser detetados. Neste caso, quando o sistema reconhece que o comando de voz terminou, pretende-se voltar a iniciar o sistema de reconhecimento de voz dada a continuidade

de interação que se pretende que exista. Deste modo, aquando da execução deste *handler*, requisita-se, novamente, a função *start()*.

- *onerror* - Parâmetro exposto pela API, presente na linha 17, que executa quando ocorre um erro no sistema de reconhecimento. Caso isto aconteça, requisita-se a execução da função *stop()* para que se termine o processo.
- *onresult* - Parâmetro exposto pela API, presente na linha 20, que executa quando o serviço de reconhecimento retorna um resultado. O resultado retornado é do tipo *SpeechRecognitionResultList*, sendo que este contém um vetor de objetos do tipo *SpeechRecognitionResult*. Cada objeto do tipo *SpeechRecognitionResult* contém objetos do tipo *SpeechRecognitionAlternative* que, por sua vez, agregam o resultado final na propriedade *transcript* e, ainda, a propriedade *confidence*, que indica, numa escala de 0 (zero) a 1 (um) o grau de confiança do resultado retornado. Neste cenário, apenas se pretende obter o resultado proveniente na propriedade *transcript* pois este contém a informação textual do comando de voz reconhecido. Uma vez obtido o resultado, passa-se à próxima fase da implementação de um assistente digital.

6.2.1.2 DialogFlow

Uma vez obtido o resultado do sistema de reconhecimento de voz, pretende-se compreender o comando de voz que foi reconhecido (NLU), gerir o discurso de modo a direcionar, de forma controlada, o diálogo do utilizador (*Dialogue Manager*) e, posteriormente, gerar discurso inteligente (*Language Generation*). Para isto, conforme definido e explicado na secção 2.6, utilizou-se a tecnologia *DialogFlow*.

De modo a integrar a tecnologia *DialogFlow* no projeto *DigitalAssistant*, utilizou-se o cliente *ApiAiClient*, que se encontra representado pela variável *client* no Excerto de Código 6.2.

```
1  private readonly token: string = environment.dialogflow.angularBot;
2  private readonly client = new ApiAiClient({ accessToken: this.token });
3
4  public async getResponse(text: string): Promise<void> {
5      this.client.textRequest(text).then(async res => {
6          if (res != null && res.result.fulfillment != null)
7              (...)
8              this.talk(res.result.fulfillment.speech);
9              (...)
10         }
11     });
12 }
```

Excerto de Código 6.2: Cliente *DialogFlow*

Antes de se iniciar a implementação do código da integração dos sistemas, deve-se, inicialmente, criar um agente de *DialogFlow* que não é mais do que uma representação de um agente virtual que irá ser treinado de forma a lidar com certos cenários controlados dentro do negócio do sistema. Uma vez criado o agente, tem-se acesso ao *token* único que o identifica, permitindo instanciar o cliente do *DialogFlow* com esse *token* de acesso, conforme representado nas linhas 1 e 2 do Excerto de Código 6.2.

Após a instanciação do cliente do *DialogFlow*, encontra-se representada a função *getResponse()*, que é chamada uma vez que seja obtido o resultado advindo do sistema de reconhecimento de voz (conforme representado na linha 22 no Excerto de Código 6.1 referente

ao módulo de reconhecimento de voz). Esta função utiliza o cliente do *DialogFlow* de modo a obter o resultado textual que será devolvido como resposta ao *input* enviado. Para que isto seja possível, o cliente do *DialogFlow* disponibiliza a função *textRequest()* que será responsável por analisar a árvore de *Intents* (conforme secção 6.2.1.3 existente no *DialogFlow*, retornando o conteúdo textual que servirá de resposta ao comando de voz enviado. Uma vez obtido o resultado final, executa-se a função *talk()*, que contém comportamento relacionado com a síntese de fala inteligente e, como tal, será abordada na secção 6.2.1.4.

6.2.1.3 Intents

Intents são ações especificas que são invocadas pelo utilizador caso este utilize comandos que contenham os termos definidos na consola do *DialogFlow*. Um agente necessita de *intents* para que possa responder aos comandos enviados, pelo que se podem definir vários *intents* para cada agente de *DialogFlow*.

Cada *intent* é constituído por:

- Contextos - Equivalente aos contextos existentes em discursos entre seres humanos. É um conceito importante pois permite o controlo total sob o fluxo da conversa, uma vez que possibilita que certas respostas advenham de decisões tomadas como consequência de respostas anteriores.
- Frases de treino - Frases de exemplo que contêm o que o utilizador poderá dizer. Caso alguma frase advinda do utilizador se pareça com uma frase de treino, o *intent* correspondente será acionado. Dado o sistema de *machine learning* integrado no *DialogFlow*, não é necessário definir exatamente todas as alternativas de frases que possam ser proferidas pelo utilizador pois esta lista será expandida de forma automática pela própria tecnologia.
- Ações e Parâmetros - Quando um *intent* é acionado, é executada uma ação previamente configurada para o *intent* correspondente. Uma vez executado o *intent*, o *DialogFlow* extrai valores do comando de entrada sob forma de parâmetros. Estes são constituídos por nome e tipo, pelo que se consideram dados estruturados que podem facilmente ser utilizados na execução de lógica interna.
- Resposta - Equivalente a uma resposta existente em discursos de seres humanos. Pode ser uma resposta definida de forma estática para um certo *intent* ou uma resposta dinâmica.

Uma vez explicada a sua constituição, apresenta-se de seguida, a titulo exemplificativo, a configuração do *intent* responsável pela seleção do módulo didático **Guitarra**.

Na figura 6.1, apresenta-se a vista de contextos na qual se definem o nome e os contextos de entrada e saída de cada *intent*. Para o caso específico do *intent* **Guitarra**, criou-se um contexto de saída para que só se possa prosseguir para o próximo passo do sistema conversacional caso este contexto exista como sendo o contexto de entrada do *intent* seguinte. Assim sendo, na figura 6.2 apresenta-se a configuração de contextos para o *intent* **Iniciante**, referente a um dos níveis de dificuldades existentes, pertencente à fase de seleção do nível de dificuldade. Este *intent* contém como contexto de entrada o contexto de saída do *intent* **Guitarra**, de forma a controlar totalmente o fluxo conversacional para que só seja possível definir um nível de dificuldade após se definir o módulo didático.

Deste modo, cumpre-se o fluxo do processo de interação do utilizador com o assistente digital conforme demonstrado no diagrama presente na figura 5.7 da secção 5.2.5.

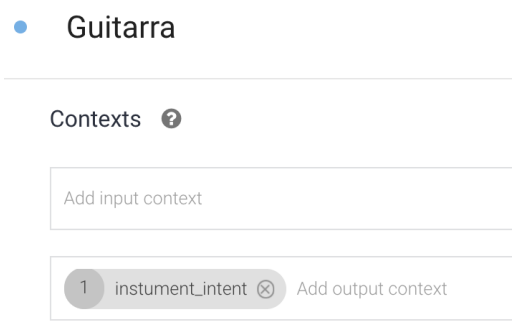

Figura 6.1: Contexto de saída de *intent* Guitarra

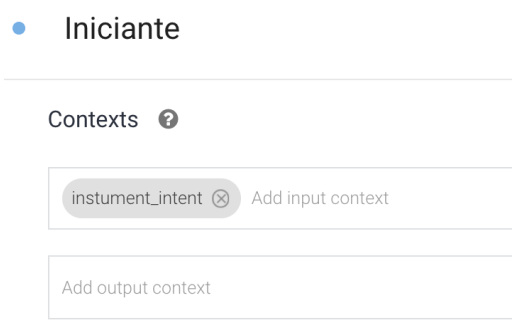

Figura 6.2: Contexto de entrada de *intent* Iniciante

Uma vez definidos os contextos do *intent*, definem-se as frases de treino que espoletam a sua execução. Na figura 6.3, apresentam-se algumas frases de treino presentes na configuração do *intent* **Guitarra**, que serão processadas e, posteriormente, irão ser responsáveis por enquadrar e associar os comandos de voz recebidos aos *intents* existentes. De modo a estruturar os dados de entrada, criou-se a Entidade[1] *Instrument* que irá aceitar apenas uma lista de instrumentos previamente definidos.

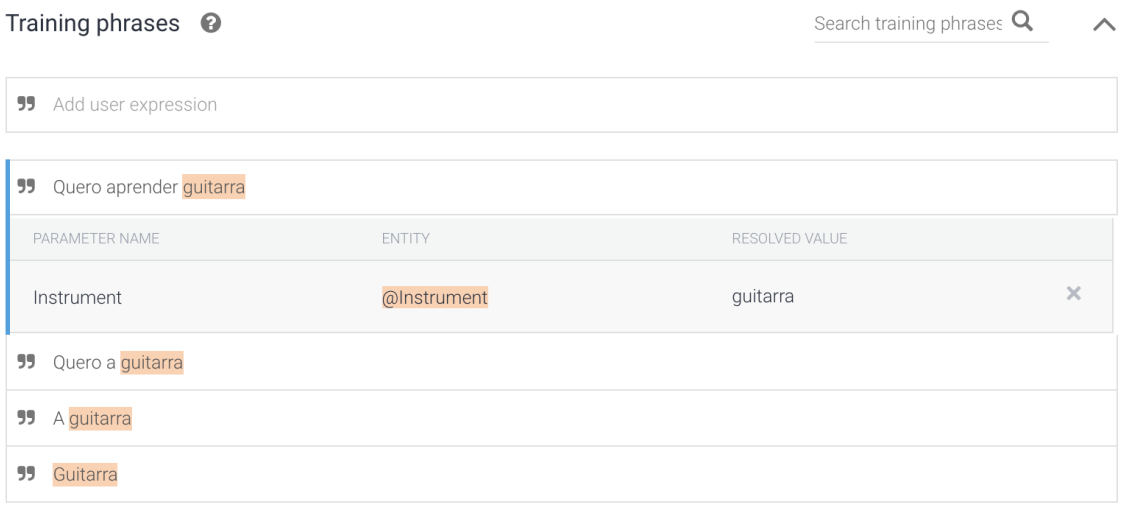

Figura 6.3: Vista de Frases de Treino da consola do *DialogFlow*

A fase seguinte é a definição das ações e dos parâmetros. Na figura 6.4 encontra-se representada a ação que guarda o valor recebido da Entidade *Instrument* na variável *$Instrument*.

[1] Conceito que define o tipo de informação a adquirir de um comando de entrada [75].

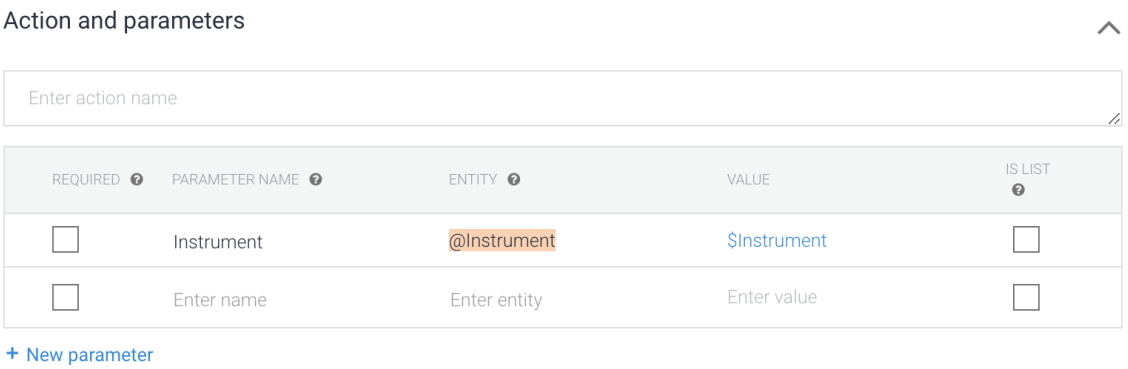

Figura 6.4: Vista de Ações e Parâmetros da consola do *DialogFlow*

Por último, define-se a resposta a retornar. Para isto, na figura 6.5 apresenta-se a vista de Resposta do *DialogFlow* na qual é possível definir respostas estáticas. Para o caso em análise, esta vista encontra-se vazia uma vez que não se pretende que a resposta a este *intent* seja de índole estática mas sim dinâmica.

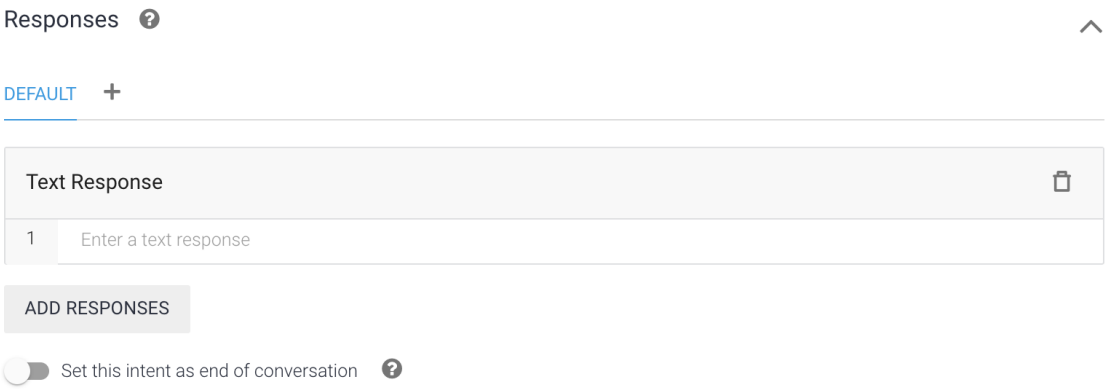

Figura 6.5: Vista de Resposta da consola do *DialogFlow*

Quando existe o requisito de retornar uma resposta dinâmica para um certo *intent*, é necessário ativar *Fulfillment* para esse mesmo *intent* de modo a que a resposta do *Dialogflow* seja uma chamada a um serviço especificamente definido pelo utilizador e não uma resposta estática definida na vista de Respostas, como presente na figura 6.5.

Na figura 6.6, encontra-se representado o passo referente à ativação do *fulfillment* para o *intent* **Guitarra**, uma vez que se pretende que a resposta retornada por este *intent* contenha todos os níveis de dificuldade existentes na API *DigitalAssistantAPI*.

Figura 6.6: Vista de *Fulfillment* da consola do *DialogFlow*

Uma vez ativa a funcionalidade, implementou-se a chamada ao serviço externo que terá a responsabilidade de retornar todos os níveis de dificuldade existentes para o módulo didático **Guitarra**. Para isto, utilizou-se o editor existente na consola do *Dialogflow* [76] que permite a definição de funções que serão automaticamente convertidas para *Google Cloud Functions*[2], permitindo logo a sua execução através de pedidos HTTP por parte do *DialogFlow*.

```
1  const functions = require('firebase-functions');
2  const {WebhookClient} = require('dialogflow-fulfillment');
3  const axios = require('axios');
4
5  exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request
       , response) => {
6      const agent = new WebhookClient({ request, response });
7      (...)
8      function getLevelsForInstrument(agent) {
9          return axios.get('https://calm-retreat-23110.herokuapp.com/
       didacticModules')
10             .then(res => {
11               const module = res.data.find(module => module.name === agent
       .parameters.Instrument);
12               const levels = module.levels.map(level => level.name);
13               agent.add(`Qual o nivel de dificuldade: ${levels.slice(0,
       levels.length - 1).join(', ') + ", ou " + levels.slice(-1)} ?`);
14             }
15         );
16     }
17     (...)
18     let intentMap = new Map();
19     intentMap.set('Guitarra', getLevelsForInstrument);
20     (...)
21     agent.handleRequest(intentMap);
22 });
```

Excerto de Código 6.3: Função implementada para o *intent* Guitarra

No Excerto de Código 6.3, apresenta-se a implementação de *Fulfillment* para o *intent* **Guitarra**, que, por requisito da própria plataforma, se encontra implementado em *Node.js* [76]. Inicialmente, instancia-se um *WebhookClient* que será o agente para o qual se irá associar os dados relativos aos *intents* e respetivas funções de *Fulfillment*, conforme representado nas linhas 2 e 6. De seguida, implementa-se a função que será responsável por alojar toda a lógica referente à frase de resposta que se pretende retornar. Para isto, implementou-se a função *getLevelsForInstrument()*, presente na linha 8. Como se pretende consumir um serviço externo alojado na API *DigitalAssistantAPI* (c.f secção 6.3) para que se obtenham todos os níveis de dificuldade, utilizou-se a biblioteca *axios* [78] para que se executem pedidos HTTP em tecnologias *Node.js* (linhas 3 e 9). Uma vez executado o pedido com sucesso, passa-se à obtenção de todos os níveis de dificuldade existentes para o módulo didático previamente selecionado e guardado na variável *$Instrument*, conforme representado na figura 6.4. Esta lógica encontra-se representada nas linhas 11 e 12. Na linha 13 através da função *add()* disponibilizada, associa-se a frase de retorno ao agente para que esta seja a frase de retorno quando o *intent* **Guitarra** for acionado. Por último, é ainda necessário definir-se quais as funções que devem ser executadas quando um certo *intent* é acionado.

[2]Ambiente de execução que proporciona a construção e conexão de serviços *cloud* e que permite programação *serverless*, uma vez que abstrai toda a gestão de infraestrutura, facilitando o processo de desenvolvimento, agilizando o processo de execução e *scaling* de código [77].

Para isto, representado pela variável *intentMap*, encontra-se a estrutura responsável por associar o *intent* à função de *Fulfillment* respetiva. Assim, conforme representado na linha 19, associa-se o *intent* **Guitarra** à função *getLevelsForInstrument()*. Uma vez implementada a estrutura de associação, utiliza-se a função *handleRequest()* para associar a estrutura ao agente atual.

6.2.1.4 WebSpeechAPI (Speech Synthesis)

Uma vez obtido o resultado advindo do *Dialogflow*, passa-se à ultima fase do assistente digital, na qual se sintetiza esse resultado num *output* de voz natural. Conforme referido na secção 2.3.3, optou-se pela utilização de *WebSpeechAPI (Speech Synthesis)*. No Excerto de Código 6.4, apresenta-se a implementação da função *talk()*, que agregará a lógica de integração com a *WebSpeechAPI* e que será executada uma vez que seja obtido o resultado advindo do *Dialogflow*, conforme representado na linha 8 do Excerto de Código 6.2.

```
1 public talk(speech: string): void {
2     const msg = new SpeechSynthesisUtterance();
3     msg.volume = 1;
4     msg.rate = 1;
5     msg.pitch = 1;
6     msg.text = speech;
7     this.response = speech;
8     speechSynthesis.speak(msg);
9 }
```

Excerto de Código 6.4: Função *talk* que integra a *WebSpeechAPI*

- *volume* - Parâmetro exposto pela API, presente na linha 3, que permite controlar a intensidade do volume sob o qual a frase será emitida. Este valor é variável entre 0 e 1 pelo que se pretende utilizar o valor máximo para esta intensidade.
- *rate* - Parâmetro exposto pela API, presente na linha 4, que permite controlar a velocidade de fala da frase em questão. Este valor é variável entre 0,1 e 10, pelo que se utiliza o valor 1 considerando que este é o valor definido por defeito.
- *pitch* - Parâmetro exposto pela API, presente na linha 5, que permite controlar o tom no qual a frase será falada. Este valor é variável entre 0 e 2, sendo que se utiliza o valor 1 considerando que este é o valor definido por defeito.
- *text* - Parâmetro exposto pela API, presente na linha 6, que permite definir o texto que será sintetizado aquando da fala da frase.
- *speak()* - Função exposta pela API, presente na linha 8, que permite adicionar uma frase à *queue* de frases a falar. Funciona como uma *queue First In First Out (FIFO)*[3], sendo que apenas será sintetizada uma frase em fala quando a frase imediatamente anterior tiver sido, também ela, processada.

6.2.2 WebAudioAPI

Conforme referido na secção 5.2.1 aquando da demonstração detalhada dos componentes do sistema, de forma a processar o áudio, utilizou-se a *WebAudioAPI*.

[3]Sigla que remete a um algoritmo de escalonamento em que o primeiro processo a chegar ao processador será o primeiro processo a ser executado.

Conforme explicado na secção 2.4 quando se comparou e se definiu a tecnologia escolhida, a *WebAudioAPI* envolve a manipulação de operações dentro de um contexto de áudio (*audio context*) e possibilita o encaminhamento modular através de nós de áudio.

Tipicamente o gráfico de encaminhamento de áudio inicia-se pela definição de um nó de áudio *source*, que providenciará dados na forma de vetores de intensidades. O resultado advindo do nó de áudio *source* tanto pode ser computado pela própria API como pode ser advindo de ficheiros áudio previamente gravados ou, ainda, proveniente de *streams* áudio em tempo real.

De seguida, na figura 6.7, apresenta-se uma representação do gráfico de encaminhamento de áudio do projeto em questão.

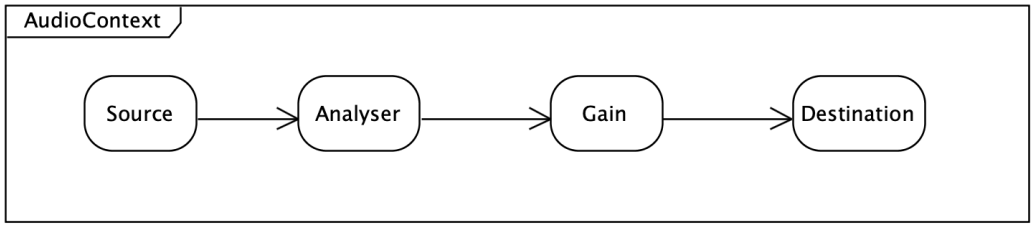

Figura 6.7: Diagrama representativo do gráfico de encaminhamento de áudio

Através da análise da figura 6.7 verifica-se a existência dos nós áudio que se seguem:

- *Source* - Nó de áudio responsável por providenciar som à cadeia do gráfico de encaminhamento.
- *Analyser* - Nó de áudio que permite a análise do som.
- *Gain* - Nó de áudio responsável pelo controlo da intensidade do som.
- *Destination* - Nó de áudio correspondente ao dispositivo que irá renderizar o áudio final.

Uma vez descritos os nós envolventes, passa-se à explicação da sua relevância no projeto, assim como à explicação dos pontos de implementação mais relevantes. De modo a capturar o som advindo de qualquer tipo de dispositivo de entrada, seja ele o próprio microfone da máquina onde se encontra implantado o sistema ou até uma interface de áudio externa, implementou-se o sistema de forma a que o nó de áudio *Source* fosse um nó do tipo *MediaStreamAudioSourceNode*, uma vez que se pretende que este opere como uma fonte de áudio cujos dados advenham de um objeto *MediaStream*.

A declaração do nó de áudio *Source* encontra-se representada na linha 2 do Excerto de código 6.5, que demonstra a inicialização do processo de encaminhamento modular dos nós áudio.

```
1  private audioContext: AudioContext = new AudioContext();
2  private microphone: MediaStreamAudioSourceNode;
3  private analyser = this.audioContext.createAnalyser();
4  private gainNode = this.audioContext.createGain();
5
6  navigator.getUserMedia({
7      audio: { (...) }
8  }, (stream) => {
9      this.stream = stream;
10     this.microphone = this.audioContext.createMediaStreamSource(stream);
11     this.microphone.connect(this.analyser);
12     this.analyser.connect(this.gainNode);
13     this.gainNode.connect(this.audioContext.destination);
14     requestAnimationFrame(this.dispatchAudioData);
15 }, (err) => {
16     (...)
17 });
```

Excerto de Código 6.5: Inicialização do processo de encaminhamento modular de nós áudio

Dada a flexibilidade que se pretende dar relativamente ao dispositivo de entrada, utiliza-se o método *enumerateDevices()* exposto pela interface *MediaDevices* da API *Navigator* para se pesquisar um dispositivo especifico que se pretende que seja o dispositivo responsável pela captura do som. Uma vez identificado o dispositivo a utilizar, recorreu-se à utilização da API de *streaming Navigator* tirando partido da funcionalidade que esta tem em retornar um objeto do tipo *MediaStream* na sua *callback* de sucesso aquando da utilização do método *getUserMedia()* que irá executar sob o dispositivo previamente selecionado. Esta pesquisa do dispositivo e a obtenção do objeto *MediaStream* através da utilização da API de *streaming* mencionada encontra-se representada no Excerto de Código 6.6.

```
1  const DEVICE_ID = 'USB Audio CODEC  (08bb:2900)';
2
3  await navigator.mediaDevices.enumerateDevices().then((devices) => {
4      myDevices = devices.filter((d) => d.kind === 'audioinput' && d.label
       === DEVICE_ID);
5  });
6
7  navigator.getUserMedia({
8      audio: {
9          deviceId: myDevices[0].deviceId,
10     }
11 }, (stream) => {
12     (...)
13 }, (err) => {
14     (...)
15 });
```

Excerto de Código 6.6: Inicialização do processo de encaminhamento modular de nós áudio

Uma vez identificado o dispositivo e após se obter o objeto *MediaStream* que irá conter toda a informação produzida externamente e captada pelo dispositivo de entrada selecionado, inicia-se a construção do gráfico de encaminhamento de áudio, iniciando-se pela instanciação do objeto *MediaStreamAudioSourceNode* através do objeto *MediaStream* (representado no Excerto de Código 6.5 na linha 10). Este primeiro passo refere-se à criação do nó de áudio

Source demonstrado na figura 6.7. Uma vez definido o nó *Source* passa-se à conexão do nó *Analyser* ao nó *microphone* (*Source*), seguido da conexão do nó *Gain* ao nó *Analyser* e, finalmente, do nó de destino ao nó *Gain*, prefazendo o gráfico representado na figura 6.7.

A estrutura demonstrada do gráfico de encaminhamento é fundamental para a implementação do projeto uma vez que, para além de se ter definido que o nó *Source* irá atuar sob dados advindos de um objeto *MediaStream* (que irá conter os dados relativos ao som produzido pelo instrumento externo), ainda se definiu a existência do nó *Analyser* que irá permitir a análise do som advindo do instrumento de modo a retornar *feedback* ao utilizador após a execução de exercícios práticos.

6.2.3 Análise do Som (Auto-correlação)

De modo a analisar o som proveniente do instrumento musical, desenvolveu-se o componente *AudioLogic* (c.f secção 5.2.1), que tem a responsabilidade de gerir toda a lógica de análise de som e retorno de *feedback* ao utilizador. No âmbito do projeto em questão, este algoritmo apenas dará suporte a instrumentos de cordas como a guitarra, como prova de conceito.

Tendo em conta que não se encontra nos objetivos deste projeto o desenvolvimento de um algoritmo de identificação áudio inovador, optou-se pela adaptação de um algoritmo previamente existente que se baseia no método da auto-correlação [79]. Este método permite a comparação de uma onda de áudio com ela mesma em vários momentos do tempo diferentes. Caso se encontre uma correspondência, tal significa que se encontrou onde a onda se repete. Uma vez encontrado em que ponto a onda se repete então é descoberta a sua frequência.

A função central desta análise, e tornando-se assim a parte principal da análise de áudio é a função *autocorrelateAudioData* que é invocada através da função *dispatchAudioData*. No Excerto de Código 6.7, encontra-se representado o inicio da implementação da função *autocorrelateAudioData*.

```
1  private frequencyBuffer = new Float32Array(this.frequencyBufferLength);
2
3  public autocorrelateAudioData(time) {
4      let rms = 0;
5      const rmsMin = 0.051;
6      (...)
7
8      this.analyser.getFloatTimeDomainData(this.frequencyBuffer);
9
10     for (let d = 0; d < this.frequencyBuffer.length; d++) {
11         rms += this.frequencyBuffer[d] * this.frequencyBuffer[d];
12     }
13
14     rms = Math.sqrt(rms / this.frequencyBuffer.length);
15
16     if (rms < rmsMin) {
17         return 0;
18     }
19
20     (...)
21 }
```

Excerto de Código 6.7: Função *autocorrelateAudioData* - redução de ruído

A implementação desta função inicia-se pela utilização da função *getFloatTimeDomainData* disponibilizada pelo nó *analyser* existente na API *WebAudioAPI* (c.f secção 6.2.2). Esta função copia o valor da onda no domínio do tempo para o vetor passado por parâmetro que terá de ser do tipo *Float32Array*, assim como representado na linha 8 e declarado na linha 1 do Excerto de Código 6.7 [80].

Uma vez obtido o valor advindo da função *getFloatTimeDomainData*, torna-se possível analisar os dados provenientes do instrumento. Assim, a primeira análise que se faz é à existência, ou não, de ruído. Para isto, calcula-se o *Root Mean Square (RMS)*[4] através do cálculo da raiz quadrada média dos valores dos dados obtidos, conforme representado nas linhas 8 a 14 do Excerto de Código 6.7. Para terminar a fase de análise de ruído, verifica-se se a qualidade do sinal é, ou não, suficiente para inferir com um grau de certeza aceitável, a frequência da onda de som captada, conforme representado na linha 16.

Se existir qualidade suficiente no sinal captado, procede-se então a inferir qual a corda tocada pelo utilizador, a partir da frequência da onda de som.

Para iniciar esta fase, tornou-se necessário o desenvolvimento da classe *Guitar* que irá agregar informação relativa a todas as cordas existentes, as suas frequências e os seus *offsets*, conforme representado na tabela 6.1.

Corda	**Frequência(Hz)**	**Offset(Hz)**
E2	82.4069	535
A2	110.000	401
D3	146.832	300
G3	195.998	225
B3	246.942	179
E4	329.628	134

Tabela 6.1: Associação entre cordas, frequências e *offsets* a 44.1Khz

Na tabela 6.1, na coluna "Corda" encontra-se a representação universal, ordenada, das cordas de uma guitarra convencional, sendo que a letra representa a nota e o número que a segue representa a oitava[5] (i.e. E2 significa o mesmo que nota Mi na segunda oitava, visto que a letra "E" é a representação universal para a nota Mi.). Na coluna "Frequência" encontram-se representados os valores constantes das frequências de cada uma das cordas. Na coluna "*Offset*" encontra-se a representação dos valores dos intervalos que serão utilizados aquando da procura pela corda que mais se parece com a frequência da corda tocada. Estes valores dos *offsets* são variáveis pois dependem do *sample rate*[6] utilizado (i.e. O valor do *offset*

[4] Medida estatística que permite calcular a potência média fornecida, encontrando-se relacionada, proporcionalmente, com a potência do sinal [81].

[5] Intervalo de notas no qual a nota mais alta tem uma frequência de vibração de onda sonora duas vezes maior do que a da sua nota mais baixa. Assim, referir que uma nota se encontra uma oitava acima significa que a nota é a mesma mas num tom mais agudo [82].

[6] Valor que define quantas vezes por segundo serão colecionadas amostras do sinal analógico enviado por um microfone ou instrumento.

equivale à divisão do *sample rate* pela frequência da nota.). Para o caso em questão utilizou-se um *sample rate* de 44.1Khz, pois este é o valor definido por defeito na criação de nós áudio na *WebAudioAPI* [83].

Posto isto, no Excerto de Código 6.8, encontra-se uma representação da classe *Guitar*, de forma a demonstrar as suas propriedades e a estrutura definida para guardar a informação referente aos *offsets* de cada corda.

```
1  export class Guitar {
2      public strings;
3      constructor(audioContext) {
4          this.strings = {
5              e2: {
6                  offset: Math.round(audioContext.sampleRate / 82.4069),
7                  difference: 0
8              },
9              a2: {
10                 offset: Math.round(audioContext.sampleRate / 110),
11                 difference: 0
12             }
13             (...)
14         };
15     }
16 }
```

Excerto de Código 6.8: Classe *Guitar* implementada

Ainda com base na análise da classe *Guitar*, importa referir que a propriedade *strings*, é uma estrutura constituída por *key* e *value*. A *key* remete-nos à identificação da própria corda, conforme demonstrado na tabela 6.1. O *value*, é constituído pela propriedade *offset* e pela propriedade *difference*. A propriede *offset* existe para que se guardem os *offsets* de cada corda, também definidos na tabela 6.1. A propriedade *difference*, presente nas linhas 7 e 11, representa a diferença existente entre a frequência da onda obtida e a frequência do *offset* ideal para cada corda. Esta propriedade é inicializada a zero para todas as cordas servindo assim como variável de controlo para que posteriormente permita selecionar a corda com menor diferença. Salienta-se que a frequência de cada corda, ao contrário dos *offsets*, não é guardada na estrutura *strings* uma vez que esta pode ser calculada pela divisão do *sample rate* e o *offset* atual da corda.

Conhecidas todas as cordas e as suas características relevantes, definidas na classe *Guitar*, inicia-se o processo de procura pela corda candidata que o utilizador poderá ter tocado, isto é, a corda que tem uma correspondência mais próxima com a que o utilizador tocou. Este processo encontra-se representado no Excerto de Código 6.9.

```
1  private instrument: Guitar = null;
2  public autocorrelateAudioData(time) {
3      const searchSize = this.frequencyBufferLength * 0.5;
4      (...)
5      for (let o = 0; o < this.stringsKeys.length; o++) {
6          offsetKey = this.stringsKeys[o];
7          offset = this.instrument.strings[offsetKey].offset;
8          difference = 0;
9
10         (...)
11
12         for (let i = 0; i < searchSize; i++) {
13             difference += Math.abs(this.frequencyBuffer[i] -
14                 this.frequencyBuffer[i + offset]);
15         }
16
17         difference = difference / searchSize;
18
19         this.instrument.strings[offsetKey].difference += (difference *
       offset);
20     }
21     (...)
22     this.stringsKeys.sort(this.sortStringKeysByDifference);
23
24     const assumedString = this.instrument.strings[this.stringsKeys[0]];
25     (...)
26 }
27
28 public sortStringKeysByDifference(string1, string2) {
29     return this.instrument.strings[string1].difference - this.instrument.
       strings[string2].difference;
30 }
```

Excerto de Código 6.9: Função *autocorrelateAudioData* - Pesquisa da corda candidata

Inicia-se o processo obtendo, para cada corda (linha 5), o *offset* previamente definido. Com este, calcula-se a diferença existente entre a frequência recebida (alojada no vetor *frequencyBuffer*) e o *offset* que se considera o valor ideal para cada corda (linhas 12 a 17). Uma vez obtida a diferença final de cada corda, persiste-se essa informação na estrutura *strings* para a corda atual da iteração (linha 19). Seguidamente, ordenam-se as cordas (linha 22) através da utilização da função *sortStringKeysByDifference* (linhas 28 a 30) que ordena as cordas existentes com base na diferença obtida. A corda com menor diferença será considerada a corda tocada pelo utilizador. Por último, após a ordenação da estrutura *strings*, considera-se que a primeira corda da estrutura é a corda candidata, uma vez que contém a menor diferença calculada (linha 24).

Na fase seguinte, detendo o candidato para a correspondência mais próxima, torna-se necessário descobrir exatamente o quão longe da frequência ideal a corda candidata se encontra. Este processo ainda pertence à função *autocorrelateAudioData* e encontra-se representado no Excerto de Código 6.10.

```
public autocorrelateAudioData(time) {
    (...)
    const assumedString = this.instrument.strings[this.stringsKeys[0]];
    const searchRange = 10;
    const searchStart = assumedString.offset - searchRange;
    const searchEnd = assumedString.offset + searchRange;
    let actualOffset = assumedString.offset;
    let smallestDifference = Number.POSITIVE_INFINITY;

    for (let s = searchStart; s < searchEnd; s++) {

        difference = 0;

        for (let i = 0; i < searchSize; i++) {
            difference += Math.abs(this.frequencyBuffer[i] -
                this.frequencyBuffer[i + s]);
        }

        difference = difference / searchSize;

        if (difference < smallestDifference) {
            smallestDifference = difference;
            actualOffset = s;
        }
    }
    return this.audioContext.sampleRate / actualOffset;
}
```

Excerto de Código 6.10: Função *autocorrelateAudioData* - Cálculo da frequência exata

Conforme se pode verificar, após a descoberta da corda candidata, são definidas algumas propriedades relevantes:

- *searchRange* - Constante utilizada para que a pesquisa pela frequência exata seja limitada, diminuindo o numero necessário de iterações a executar para que a frequência seja descoberta. Se estamos perante uma situação em que sabemos a corda cujo valor do *offset* é o mais próximo do executado pelo utilizador, tal significa que não é necessário iterar por todos os valores dos *offsets* existentes, uma vez que se sabe que a frequência estará próxima do *offset* da corda candidata.
- *searchStart* / *searchEnd* - Constantes que utilizam a constante *searchRange* previamente definida, de forma a delimitar a pesquisa do *offset* ideal. A constante *searchStart* representa o limite inicial da pesquisa. A constante *searchEnd* representa o limite final da pesquisa.
- *actualOffset* - Variável que é inicializada com o valor do *offset* ideal da corda candidata previamente encontrada. Apesar de se ter descoberto a corda cujo *offset* se encontra mais perto da frequência tocada, esta variável tem como principal objetivo ser posteriormente substituída pelo valor exato do *offset* da corda tocada pelo utilizador.
- *smallestDifference* - Variável inicializada com um valor infinito positivo [84] para que permita encontrar a menor diferença.

Uma vez feita a pesquisa pelo *offset* exato, retorna-se o valor da divisão do *sample rate* pelo *offset* final, que corresponde ao valor da frequência obtida. Assim, a função *autocorrelateAudioData* termina o seu propósito, retornando esta frequência para a função que a invocou,

a função *dispatchAudioData*, para que esta, com base na frequência calculada, infira a nota e a oitava que foi tocada pelo utilizador.

De forma a descobrir a nota e as oitavas através da frequência obtida, torna-se necessário entender o conceito da frequência de afinação padrão da nota Lá 440, padronizada como ISO 16 [85] (ou A440, ou ainda A4 se pretendermos a representação universal, uma vez que a letra "A" representa a nota Lá e o número 4 representa a quarta oitava). Este padrão especifica que a frequência da nota Lá deve ser 440Hz, servindo esta como a base de afinação de todas as outras notas. Se tomarmos como exemplo a tabela 6.1, reparamos que a corda A2 de uma guitarra tem uma frequência de 110Hz. Quando se sobe uma oitava, tal significa que a frequência respetiva duplica [86] (i.e. A nota A2 tem uma frequência de 110Hz, pelo que a nota A3 terá uma frequência de 220Hz). Assim sendo, considerando que o valor padrão definido é de 440Hz para A4, e concluindo que a diferença de A2 para A4 são duas oitavas, tal justifica o valor 110Hz representado na tabela 6.1 para A2, visto que este é o resultado da divisão por dois por cada oitava. E deste modo, seguindo este padrão, todas as notas obedecem ao mesmo sistema, sendo que até existe uma tabela de frequências que contem as notas existentes e as suas frequências com base neste sistema [87].

No Excerto de Código 6.11, encontra-se representada a função *dispatchAudioData*.

```
1 public dispatchAudioData() {
2     (...)
3     const frequency = this.autocorrelateAudioData(time);
4     const octavesFromA4 = Math.log2(frequency / 440);
5     const semitonesFromA4 = 12 * octavesFromA4;
6     let octave = 4 + ((9 + semitonesFromA4) / 12);
7     octave = Math.floor(octave);
8     const note = (12 + (Math.round(semitonesFromA4) % 12)) % 12;
9 }
```

Excerto de Código 6.11: Função *dispatchAudioData*

A análise do Excerto de Código 6.11 permite concluir que, após obter a frequência emitida pelo instrumento do utilizador, a função *dispatchAudioData* utiliza o sistema A440 para determinar a quantas oitavas de distância se encontra a nota obtida da nota A4 (linha 4) [88]. Uma vez obtida a quantidade de oitavas de distância, pretende-se obter a quantidade de semitons que constituem essa distância. Um semitom é considerado o menor intervalo existente entre notas [89]. Assim, e como se pode analisar na figura 6.8, uma oitava é constituída por doze semitons.

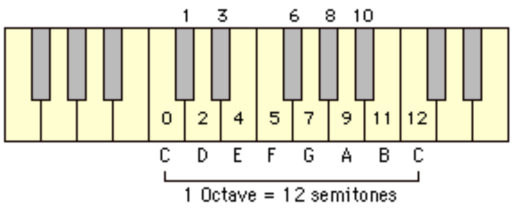

Figura 6.8: Representação de uma oitava e dos semitons que a constituem num teclado musical [89]

Deste modo, para que se obtenha o número de semitons que distanciam a frequência obtida da nota A440, multiplica-se as oitavas pelo número de semitons existentes numa oitava,

conforme representado na linha 5.

Com este valor, é então possível calcular a nota tocada pelo utilizador e a oitava relativa a essa nota. O cálculo da oitava faz uso do valor constante quatro (referente ao valor da oitava atual uma vez que se está a utilizar o sistema padrão A4), e do valor constante nove, uma vez que este valor se refere à quantidade de semitons que são necessários descer de modo a que se desça para A3 (terceira oitava da notá Lá). Dito isto, através da fórmula que se encontra presente na linha 6 do Excerto de Código 6.11, calcula-se a oitava [86]. Por último, ainda através do valor correspondente à distância de semitons até A4, calcula-se a nota respetiva conforme a fórmula presente na linha 8 [86].

Uma vez obtido o valor da nota e da oitava através da frequência emitida pelo instrumento do utilizador, torna-se possível o retorno de *feedback* com base em exercícios que identifiquem se as notas e as oitavas foram tocadas em conformidade, criando assim um algoritmo que sirva como prova de conceito para que se teste se é relevante a existência de um assistente digital integrado com algoritmos de análise de som.

6.3 DigitalAssistant API

Nesta secção aborda-se o processo de implementação da API desenvolvida que assenta no padrão arquitetural *Representational State Transfer (REST)* e que, conforme explicado na secção 5.1.1, terá a responsabilidade de alojar e gerir todos os módulos didáticos, níveis de dificuldade e exercícios que serão consumidos pela API do *DialogFlow* para que possam ser expostas e utilizadas pelo *DigitalAssistant*.

De modo a manter a consistência tecnológica entre esta API e as funções de *Fulfillment* implementadas aquando da definição de respostas dinâmicas no *Dialogflow*, utilizou-se, também, a tecnologia *Node.js* para a implementação desta API. De forma a complementar esta tecnologia, e com o objetivo de agilizar o processo de implementação, utilizou-se a biblioteca *Express* [90]. Considerando as normas e boas práticas para a utilização desta biblioteca [91], e conforme explicado na secção 5.2.1, esta API encontra-se modelada nas camadas: *Models*, *Controllers* e *Routes*. Assim sendo, apresentam-se de seguida as secções que abordam cada uma das camadas constituintes a par com Excertos de Código relevantes.

6.3.1 Models

A camada de modelos é a camada responsável por alojar toda a lógica de estruturação das entidades de domínio que serão persistidas na base de dados. Para que exista esta interação com a base de dados, torna-se necessária a definição do tipo de abordagem que se deve seguir aquando da implementação de tal integração. Tipicamente, as alternativas viáveis ou contemplam a utilização da linguagem de *queries* nativa da camada de base de dados ou a utilização de um *Object Data Model (ODM)*[7]. Apesar do melhor desempenho que se pode obter ao utilizar a primeira opção, a utilização de um ODM geralmente resulta em menos custos de desenvolvimento e manutenção, abstraindo a linguagem nativa da base de dados [93], pelo que se optou por esta abordagem, utilizando-se a tecnologia *Mongoose*[8].

[7] Modelo de dados que trata conjuntos de dados como sendo objetos, atribuindo-lhes propriedades e valores, estruturando-os [92].

[8] Tecnologia ODM desenhada para sistemas não relacionais [94].

Uma vez explicada a definição da abordagem e da tecnologia a utilizar, demonstra-se no Excerto de Código 6.12, a titulo exemplificativo, o código responsável pela definição da estrutura da entidade Módulo Didático.

```
1  var mongoose = require('mongoose');
2  var Schema = mongoose.Schema;
3
4  const levelSchema = require('./levelModel');
5
6  var didacticModuleSchema = new Schema({
7    name: { type: String },
8    theoreticalIntroduction: { type: String },
9    levels: [levelSchema]
10 });
11
12 module.exports = mongoose.model('DidacticModule', didacticModuleSchema);
```

Excerto de Código 6.12: Classe representativa da modelação da entidade Módulo Didático

Inicialmente, nas linhas 1 e 2, define-se o ODM a utilizar e a variável *Schema* que será utilizada para modelar a entidade. De seguida, nas linhas 6 a 10, define-se a estrutura que se pretende que a entidade Módulo Didático contenha (c.f secção 4.1). Deste modo, nas linhas 7, 8 e 9, define-se o nome, a introdução teórica do módulo e os níveis que o constituem, respetivamente. Como o Nível é também uma entidade, esta encontra-se modelada num ficheiro à parte pelo que tem de ser importado especificamente (conforme presente na linha 4) para que possa ser utilizado (conforme presente na linha 9).

6.3.2 Controllers

Esta camada tem a responsabilidade de alojar as funções que serão invocadas aquando da definição de rotas (c.f 6.3.3). De modo a exemplificar a definição de um ficheiro controlador, assim como as funções que este aloja, demonstram-se de seguida excertos dessa implementação.

```
1  var mongoose = require('mongoose'),
2      DidacticModule = mongoose.model('DidacticModule');
3
4  const Controller = {};
```

Excerto de Código 6.13: Inicialização do *controller domainLogicController*

Inicialmente, conforme representado no Excerto de Código 6.13, declara-se a utilização de *Mongoose* e define-se a variável *DidacticModule* como sendo o modelo definido na camada de *Models*, como demonstrado na linha 12 do Excerto de Código 6.12. De seguida, definem-se as funções que serão expostas à camada de rotas (c.f 6.3.3).

Para exemplificar a definição destas funções, encontram-se representadas no Excerto de Código 6.14 duas funções controladoras, sendo que uma tem como objetivo listar módulos didáticos e outra criar.

```
1  (...)
2
3  // Method used to get all didactic modules
4  Controller.get_all_didacticModules = function (req, res) {
5      DidacticModule.find({}, function (err, didacticModule) {
6          if (err) {
7              res.status(500).end();
8          } else {
9              res.json(didacticModule);
10         }
11     });
12 };
13
14 // Method used to create a didactic module
15 Controller.create_a_didacticModule = function (req, res) {
16     var new_DidacticModule = new DidacticModule(req.body);
17     new_DidacticModule.save(function (err, didacticModule) {
18         if (err) {
19             res.status(500).end();
20         } else {
21             res.json(didacticModule);
22         }
23     });
24 };
25
26 (...)
```

Excerto de Código 6.14: Funções constituintes do *controller domainLogicController*

A primeira função, definida e iniciada na linha 4 do Excerto de Código 6.14 utiliza a função *find()* para que sejam retornados todos os módulos didáticos existentes na base de dados. Esta função recebe como primeiro parâmetro a filtragem que se pretende associar ao pedido. Caso se envie um parâmetro vazio, como o caso em análise (presente na linha 5), serão retornados todos os dados, sem qualquer tipo de filtragem [95]. O segundo parâmetro da função *find()* trata-se da função *callback* que se pretende que seja executada assim que exista uma resposta. Deste modo, conforme representado nas linhas 6 a 10, caso ocorra algum erro durante a execução da função *find()*, retorna-se erro. Caso contrário retornam-se, em formato *JSON*, os módulos didáticos existentes.

A segunda função, definida e iniciada na linha 15, do Excerto de Código 6.14 utiliza a função *save()* que, à semelhança da função *find()*, recebe uma função *callback* que será executada assim que seja gravado o módulo didático na base de dados. Ainda seguindo a estrutura definida na primeira função, e conforme representado nas linhas 18 a 22, retorna-se um erro caso a execução da função *save()* falhe ou, caso suceda, retorna-se o módulo didático que acabou de ser criado na base de dados.

6.3.3 Routes

Esta camada remete a uma secção de código *Express* que tem como propósito associar um verbo *HTTP* a uma função existente num controlador correspondente, servindo como ponto de entrada para pedidos externos. Assim, é considerada a ponte de comunicação entre o

cliente e o servidor, uma vez que irá ser responsável por expor todos os *endpoints*[9] existentes no sistema.

Os *endpoints* disponibilizados pelo *DigitalAssistantAPI* são os que se seguem:

- GET /didacticModules - Retorna todos os módulos didáticos existentes.
- POST /didacticModules - Permite a criação de módulos didáticos com níveis de dificuldade e exercícios.
- GET /didacticModules/:didacticModuleId - Retorna o módulo didático correspondente ao identificador *didacticModuleId*.
- GET /didacticModules/:didacticModuleId/levels - Retorna todos os níveis de dificuldade existentes para o módulo didático identificado pelo identificador *didacticModuleId*.
- GET /didacticModules/:didacticModuleId/levels/:levelId - Retorna o nível de dificuldade correspondente ao identificador *levelId* que pertença ao módulo didático correspondente ao identificador *didacticModuleId*.

Uma vez identificados os *endpoints* existentes, demonstra-se, no Excerto de Código 6.15, a declaração de dois *endpoints* que servirão como exemplo quanto à sua implementação.

```
1 var domainLogicController = require('../controllers/domainLogicController'
      );
2
3 app.route('/didacticModules')
4     .get(domainLogicController.getAllDidacticModules)
5     .post(domainLogicController.createDidacticModule);
6 (...)
```

Excerto de Código 6.15: Declaração de *endpoints* referentes à *DigitalAssistantAPI*

Inicialmente, conforme representado na linha 1, define-se o *Controller* (c.f secção 6.3.2) para o qual se irá associar os pedidos recebidos. De seguida, definem-se as rotas e os *controllers* aos quais as associar. Deste modo, na linha 3, encontra-se representado um exemplo no qual se define a rota *didacticModules* que irá ser associada ao método *getAllDidacticModules*, do *controller domainLogicController*, aquando de pedidos *GET* e ao método *createDidacticModule*, do *controller domainLogicController*, aquando de pedidos *POST*.

[9] Ponto de entrada num canal de comunicação no qual dois sistemas interagem [96].

Capítulo 7

Experiências e Avaliação

Neste capítulo, definem-se os indicadores de avaliação, as suas hipóteses e as metodologias de avaliação. Por último, após a atribuição dos métodos de avaliação aos indicadores de avaliação correspondentes, avaliam-se os resultados obtidos conforme as hipóteses formuladas.

7.1 Indicadores de Avaliação

Os identificadores de avaliação têm como objetivo auxiliar a avaliação do projeto em questão. Deste modo, nesta secção indicam-se os identificadores de avaliação considerados:

- Cumprimento de Requisitos Funcionais e Não Funcionais - O cumprimento dos requisitos permitirá concluir se os requisitos identificados foram alcançados.
- Usabilidade e satisfação do utilizador - Segundo a norma ISO 9241-11 a usabilidade é o que permite medir se um produto pode ser usado por utilizadores específicos para atingir metas específicas com eficácia, eficiência e satisfação num contexto de utilização especifico. Assim, esta é um atributo de qualidade de *software* constituinte do modelo FURPS+ (c.f secção 4.3) que visa identificar a facilidade de interação entre o utilizador e o produto, assim como a sua satisfação.

7.2 Definição de Hipóteses

A definição de hipóteses tem como propósito avaliar os indicadores previamente definidos. Assim, foram estabelecidas as seguintes hipóteses:

- Cumprimento dos requisitos da solução.
- Satisfação dos utilizadores igual ou superior a 80%.
- Grau de usabilidade igual ou superior a 80%.

A primeira hipótese apresentada remete para o indicador relativo ao cumprimento dos requisitos funcionais e não funcionais, uma vez que se pretende avaliar se estes foram alcançados.

A segunda hipótese está relacionada com o indicador de satisfação dos utilizadores e indica que a satisfação dos utilizadores deve ser igual ou superior a 80%. Este valor advém da utilização do sistema *Customer Satisfaction Score (CSAT)* que é um indicador chave da satisfação dos utilizadores [97]. Este indica que a satisfação dos utilizadores é alcançada quando se atinge um grau igual ou superior a 80% pelo que, quanto maior a percentagem, maior a satisfação alcançada.

Relativamente à terceira hipótese, esta prende-se com o indicador relativo ao grau de usabilidade dos utilizadores perante o sistema e, para manter a consistência, utilizou-se a mesma percentagem mínima advinda do sistema CSAT, que foi utilizado para o indicador de satisfação.

7.3 Metodologia de Avaliação

Nesta secção apresentam-se as metodologias de avaliação utilizadas na avaliação das hipóteses identificadas. Assim, referem-se os testes de *software*, o inquérito de satisfação, o inquérito de usabilidade e, por último, os testes de hipóteses.

7.3.1 Testes de Software

Os testes de *software* têm um papel fundamental na validação de qualquer projeto na medida em que permitem avaliar e identificar possíveis erros, falhas funcionais e, até, falhas na especificação dos requisitos implementados. Assim, testar o sistema é avaliá-lo, de modo a que se reconheça que o sistema cumpre, ou não, os requisitos propostos [98].

Para que se definissem os testes a desenvolver, seguiu-se o modelo *V-Model* [99], representado na figura 7.1, que indica que para cada fase de desenvolvimento existe estabelecida uma fase de testes correspondente.

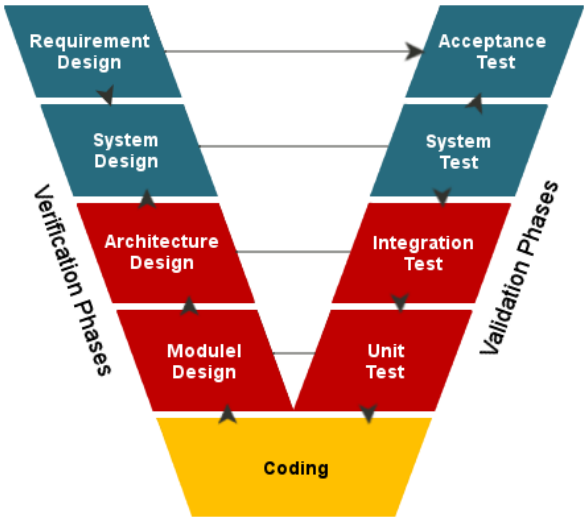

Figura 7.1: Aquitetura do modelo *V-Model* adotada na fase de validação [99]

Todas as fases do desenvolvimento são consideradas parte integrante da Fase de Verificação, pelo que todas as fases de testes são consideradas constituintes da Fase de Validação. Considerando o contexto no qual surge a necessidade de se adoptar a utilização deste modelo, apenas se abordarão as fases constituintes da Fase de Validação, uma vez que é precisamente esse o objetivo para o qual se propôs a sua utilização.

As quatro categorias de Testes de Software que se encontram representadas no modelo presente na figura 7.1 - Testes Unitários, Testes de Integração, Testes de Sistema e Testes de Aceitação - serão abordadas na secção 7.5.1.

7.3.2 Inquérito de Satisfação

De modo a analisar a satisfação do utilizador, desenvolveu-se um questionário que visa verificar se o utilizador se encontra satisfeito com o sistema desenvolvido. A eleição desta metodologia de avaliação relaciona-se com o facto de se tratar o indicador da satisfação como sendo um indicador subjetivo.

O questionário em questão segue uma estrutura constituída por perguntas concretas, evitando a ambiguidade e de resposta fechada. Relativamente às respostas, a escala utilizada para a sua realização é a escala baseada no modelo *Likert Scale* [100], sendo que esta é normalmente constituída por cinco opções:

- Discordo totalmente;
- Discordo parcialmente;
- Indiferente;
- Concordo parcialmente;
- Concordo totalmente.

Pretende-se a utilização desta mesma escala com cinco opções para dar a permissão da escolha de uma resposta intermédia por parte do utilizador, caso este não sinta nem uma agravante nem uma melhoria relacionada com a questão exposta no inquérito.

Conforme referido na secção 4.1, tendo em conta o tipo de utilizadores alvo, estes inquéritos encontram-se integrados no próprio *software*, sendo a sua interação executada via comandos áudio, facilitando todo o processo de interação.

7.3.3 Inquérito de Usabilidade

Para se verificar a usabilidade do sistema, e considerando que a usabilidade é o requisito não funcional crucial considerando o modelo de negócio e o objetivo social proposto, serão, também, desenvolvidos questionários que visam avaliar a dificuldade de interação com o sistema por parte do utilizador. Para isto, definiu-se *à priori* um guião que deve ser seguido pelo utilizador de modo a realizar algumas ações no *software*. Por último, realiza-se o questionário de usabilidade que irá recorrer à mesma estrutura de respostas fechadas e à mesma escala indicada nos inquéritos de satisfação.

Assim como nos inquéritos de satisfação, este inquérito também será completamente integrado no sistema desenvolvido e interativo via comandos áudio.

As questões que perfazem o inquérito de usabilidade são baseadas no inquérito *System Usability Scale* criado por John Brooke em 1986 [101] que tem como objetivo medir a usabilidade de um sistema. De forma a complementar esta matéria, adaptaram-se ainda questões advindas das dez heurísticas de *Nielsen*, um conjunto de princípios criado por Jakob Nielsen em 1994 [102] que visam aumentar a usabilidade de um sistema.

7.3.4 Testes de Hipóteses

De forma a proceder à avaliação da solução com base nos indicadores definidos, pode utilizar-se Testes de Hipóteses. Estes são considerados uma metodologia estatística que auxilia na tomada de decisões sobre uma população com base na informação advinda da amostra.

Num teste de hipóteses definem-se duas hipóteses [103]:

- H_0 - A hipótese nula que se especula rejeitar;
- H_1 - A hipótese alternativa que se pretende verificar.

Para se determinar se a hipótese nula deve ser rejeitada, utiliza-se o valor de prova (*p-value*), pois este considera-se o valor mínimo da significância que leva à rejeição de H_0. Assim, se *p-value* <= α (nível de significância do teste), rejeita-se H_0 [103]. O nível de significância definido para os testes a realizar assume-me que seja de 0.05, valor que corresponde a um grau de confiança de 95%.

A representação dos testes de hipóteses pode ser feita através de testes paramétricos e testes não paramétricos. Os testes paramétricos pressupõem a normalidade em amostras de dimensões igual ou inferior a trinta [103], pelo que só devem ser aplicados caso isto se verifique. Em contrapartida, os testes não paramétricos não necessitam de requisitos tão fortes, podendo ser aplicados caso a amostra não siga uma distribuição normal ou, ainda, se a amostra for pequena.

Os testes paramétricos oferecem vantagens perante os testes não paramétricos na medida em que possuem um poder estatístico muito maior, sendo mais capazes de levar a uma rejeição da hipótese nula [103].

7.4 Atribuição dos Métodos de Avaliação

Na tabela 7.1 apresenta-se a associação entre os indicadores definidos e os métodos de avaliação que se vão utilizar.

Indicador	**Método de Avaliação**
Cumprimento dos Requisitos	Testes de Software
Satisfação	Inquérito de Satisfação e Testes de Hipóteses
Usabilidade	Inquérito de Usabilidade e Testes de Hipóteses

Tabela 7.1: Associação entre indicador e método de avaliação correspondente.

7.5 Avaliação de Resultados

Nesta secção procede-se à avaliação de resultados perante as hipóteses definidas. Desta forma, demonstram-se os resultados obtidos advindos de cada método de avaliação efetuado, baseando a sua análise em função do indicador de avaliação e da hipótese formulada.

7.5.1 Testes de Software

Nesta secção analisam-se as quatro categorias de testes de *software* enunciados na secção 7.3.1 de forma a avaliar a hipótese referente ao cumprimento dos requisitos da solução. Para que se demonstre os tipos de testes implementados, cada categoria de testes é analisada a par com exemplos de testes desenvolvidos à mesma.

7.5.1.1 Testes Unitários

Os testes unitários devem ser encarados como os primeiros testes a desenvolver na medida em que estes têm como finalidade a validação de pequenas partes de código, de modo a perceber se o seu objetivo funcional é cumprido. Para isto, aplicam-se às pequenas partes do código que se considerem testáveis, individuais e capazes de aceitar um *input* e produzir um *output* por si só [104].

Considerando a existência de dois projetos principais - *DigitalAssistant* e *DigitalAssistantAPI* - houve a necessidade de utilizar tecnologias distintas na execução de testes unitários para cada projeto.

Para o projeto *DigitalAssistant* utilizou-se a *framework* de testes *Mocha* [105] e recorreu-se à utilização de *Chai* [106] como biblioteca de asserções. *Mocha* é uma *framework* de testes que providencia funcionalidade para testar, de forma simples, código *javascript* síncrono e assíncrono. De forma a complementar esta *framework*, utilizou-se *Chai*, uma vez que esta é uma biblioteca *Behavior Driven Development (BDD)*[1], muitas vezes utilizada a par com a *framework Mocha*, auxiliando nas asserções, de modo a que, num determinado momento, seja possível inserir um predicado no teste que verifique a validade de uma certa condição.

Para que se criasse um padrão de desenvolvimento de testes, utilizou-se o padrão de desenvolvimento de testes *Arrange-Act-Assert (AAA)*. Este infere que qualquer teste deve conter uma secção inicial de descrição de configurações necessárias (*Arrange*), seguido de uma secção na qual se descreve o comportamento alvo de teste (*Act*). Por último, deve ainda conter uma secção na qual se avalie e verifique que o comportamento a ser testado esta a sortir o efeito pretendido (*Assert*) [107].

No Excerto de Código 6.6 encontra-se um exemplo de um teste unitário realizado a uma parcela de código constituinte do *DigitalAssistant*. Para o caso em questão, pretende-se validar se a função *getNotesAndOctaveToPlay* retorna o *output* esperado para o *input* enviado.

```
1  describe('Audio logic component tests', function () {
2      it('getNoteAndOctaveToPlay - Note equal to Sol and octave equal to 4',
       () => {
3          // ARRANGE
4          let noteAndOctave = null;
5          // ACT
6          noteAndOctave = AudioLogic.getNoteAndOctaveToPlay('Toca a nota Sol
       na Quarta oitava');
7          // ASSERT
8          expect(noteAndOctave.noteToPlay).to.be.equal('Sol');
9          expect(noteAndOctave.octaveToPlay).to.be.equal(LocalizedOctaves.
      Quarta);
10     });
11 });
```

Excerto de Código 7.1: Teste unitário à função *getNoteAndOctaveToPlay*

À semelhança do Excerto de Código 7.1, todos os outros testes unitários desenvolvidos no âmbito do *DigitalAssistant* seguem a mesma tecnologia e abordagem pelo que, para a criação destes testes, diversos métodos da *framework Mocha* foram utilizados [105]:

[1] Abordagem de desenvolvimento de *software* na qual se utiliza linguagem corrente para descrever testes perceptíveis.

- *describe* - Função na qual se descrevem um conjunto de testes.
- *it* - Função que define um teste em particular, pertencente a um certo conjunto de testes, que poderá conter várias asserções.

A par com estas funções, utilizou-se ainda a função *expect* da biblioteca *Chai* que permite a validação dos dados através de um sistema de asserções BDD [106].

No que diz respeito ao projeto *DigitalAssistantAPI*, também se utilizou a *framework* de testes unitários *Mocha*, como é possível verificar pela análise do Excerto de Código 7.2.

```
1  describe('create_a_didacticModule', function () {
2      let sandbox = sinon.sandbox.create();
3
4      beforeEach(function () {
5          res = {
6              json: sinon.spy(),
7          };
8      });
9
10     it('Should return a created Didactic Module', sinon.test(function () {
11         // ARRANGE
12         expectedResult = req.body
13         const stub = sandbox.stub(DidacticModule.prototype, 'save').yields
       (null, expectedResult);
14         // ACT
15         Controller.create_a_didacticModule(req, res);
16         // ASSERT
17         sinon.assert.calledWith(res.json, sinon.match({ name: req.body.
       name }));
18         sinon.assert.calledWith(res.json, sinon.match({
       theoreticalIntroduction: req.body.theoreticalIntroduction }));
19         stub.restore();
20     }));
21 });
```

Excerto de Código 7.2: Teste unitário à função de criação de um módulo didático

Quanto à análise da cobertura de testes sob o código implementado, utilizou-se a ferramenta de cobertura de testes *Istanbul*, que providencia um sistema de relatórios integrado no qual apresenta, ou no terminal ou em *output* HTML os resultados obtidos [108]. Assim sendo, apresenta-se na figura 7.2, o output HTML obtido através da utilização da ferramenta no projeto *DigitalAssistantAPI*.

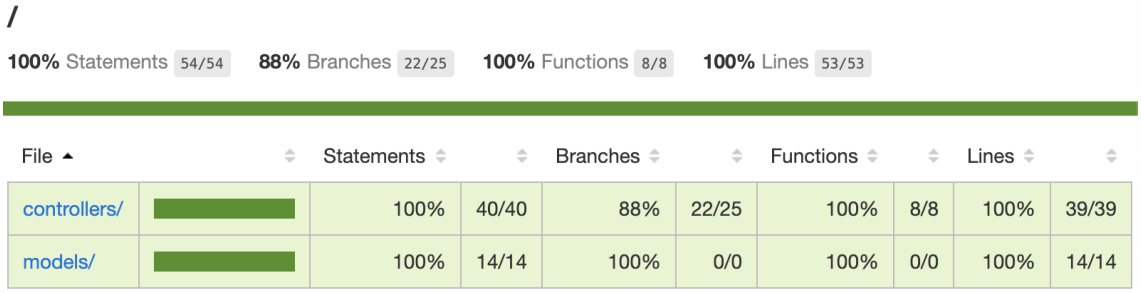

Figura 7.2: Cobertura de testes de *DigitalAssistantAPI*

A figura 7.2 permite concluir que os valores de cobertura são positivos, uma vez que se atingiu o valor máximo de cobertura na categoria de *statements*. No entanto, atingir o valor máximo de cobertura na categoria de *statements* não garante cobertura máxima na categoria de *branches*, uma vez que este último se refere a todo o fluxo de decisões existente no código testado. Ainda assim, atingiu-se 88% de cobertura neste aspeto. Quanto à cobertura de testes relativamente às funções e respetivas linhas de código, alcançou-se, também, o valor máximo.

7.5.1.2 Testes de Integração

Os testes de integração sucedem os testes unitários no modelo representado na secção 7.3.1 e têm como função testar as interações entre os vários componentes do sistema de modo a testar se estes funcionam como um todo [109].

```
1  describe('Integration Tests', async function () {
2      it('exercise "Cordas Soltas" - Check DialogFlow Integration', async ()
       => {
3          const client = new ApiAiClient({ accessToken: environment.
       dialogflow.angularBot });
4          const exercise = 'Cordas Soltas';
5          const response = await client.textRequest(exercise).catch(error =>
6          {
7              return Promise.resolve(error);
8          });
9          const speech = response.result.fulfillment.speech;
10         if (speech != null) {
11             expect(speech).to.be('Toca a corda Mi');
12         } else {
13             expect.fail();
14         }
15     });
16 });
```

Excerto de Código 7.3: Teste à integração entre a tecnologia do *DialogFlow* e *DigitalAssistant*

A análise do Excerto de Código 7.3, demonstra um exemplo de um teste de integração desenvolvido, no qual se testa se a resposta proveniente de um pedido ao *DialogFlow* está conforme o expectável. Na linha 3, define-se a referência para o *DialogFlow*, sendo que na linha 4 se define o exercício que se pretende requisitar na linha 5. Uma vez obtida a resposta do *DialogFlow* verifica-se se a resposta contém a instrução do exercício esperado (linha 11), verificando-se, assim, a integração entre os vários componentes constituintes do sistema.

7.5.1.3 Testes de Sistema

Os testes de sistema têm como objetivo testar o sistema na íntegra, envolvendo os requisitos funcionais e os requisitos não funcionais [110]. Assim, os testes de sistema devem ser baseados nas definições de casos de uso.

Para cada um dos testes de sistema desenvolvidos, seguiu-se um certo procedimento e um conjunto de critérios orientados à definição de cada caso de uso, que terão de ser validados. A validade do teste apenas é obtida caso não falhe qualquer critério previamente definido. Assim, caso falhe pelo menos um critério, o teste falhará também.

Na tabela 7.2, apresenta-se o teste de sistema ao caso de uso Requisitar Módulo (UC1).

Tabela 7.2: Teste de Sistema para Requisitar Módulo Didático

Caso de Uso	UC1
Título	Requisitar Módulo Didático
Cenário	Os módulos didáticos são requisitados por parte do utilizador para que sejam apresentados todos os módulos existentes e para que, após escolha do módulo, sejam apresentados os níveis de dificuldade existentes.
Procedimento	O utilizador requisita módulo didático através do comando de voz que contenha as palavras chave "Módulo Didático". O sistema apresenta os módulos didáticos. O utilizador escolhe o módulo didático que pretende e o sistema requisita definição de nível de dificuldade.
Critérios	1. O sistema reconhece o pedido advindo do comando de voz para que se apresentem todos os módulos didáticos. 2. Os módulos didáticos são apresentados ao utilizador. 3. O sistema reconhece o pedido advindo do comando de voz para que se selecione o módulo didático pretendido. 4. O módulo didático é selecionado com sucesso.
Resultados	1. Passou 2. Passou 3. Passou 4. Passou
Resultado Final	Passou

A análise da tabela 7.2 permite concluir que, face aos critérios definidos e aos resultados obtidos, o sistema é capaz de proceder à requisição de um Módulo Didático. Assim, conclui-se que o caso de uso Requisitar Módulo Didático (UC1) foi desenvolvido corretamente.

No Anexo A, encontram-se os restantes testes de sistema executados a cada caso de uso existente no sistema desenvolvido.

7.5.1.4 Testes de Aceitação

O objetivo dos testes de aceitação é verificar se o sistema desenvolvido cumpre com os requisitos definidos pelo cliente. Assim sendo, estes permitem aferir se o sistema cumpre os critérios de satisfação propostos pelo cliente [111]. No caso presente, serão realizados testes de aceitação do utilizador.

Para determinar se um teste é aceite ou não, definem-se vários critérios relativos aos requisitos do sistema e à satisfação do cliente que devem ser cumpridos sem exceção. Caso falhe pelo menos um critério, o teste não será aceite. Na tabela 7.3, apresenta-se uma tabela na qual se demonstram os critérios de satisfação propostos de toda a interação entre o utilizador e o sistema desde a requisição do Módulo Didático até à realização de um exercício.

Tabela 7.3: Teste de Aceitação para Realizar Exercício

Cenário	Requisitar Ajuda para que se consiga Realizar Exercício Prático após definição de Módulo Didático, do Nível de Dificuldade e do Exercício a realizar
Descrição	Inicialmente, requisita-se o módulo de ajuda para se entender quais os comandos existentes no sistema. Posto isto, de modo a testar toda a estrutura funcional até à realização de um exercício, inicia-se o cenário pela requisição de um Módulo Didático, seguido da definição de um Nível de Dificuldade. Posteriormente, escolhe-se um exercício prático dos exercícios propostos pelo sistema para que este se realize. Considerando a escolha de um exercício de caráter prático, o sistema solicitará dados advindos do instrumento do utilizador para que possa avaliar e oferecer *feedback*.
Critérios	1. O sistema reconhece o comando de voz responsável pela requisição do módulo de ajuda. 2. O módulo de ajuda é apresentado com sucesso ao utilizador. 3. O sistema reconhece o comando de voz responsável pela requisição dos módulos didáticos. 4. Os módulos didáticos são apresentados com sucesso. 5. O sistema reconhece o comando de voz responsável pela seleção de um módulo didático. 6. O módulo didático é selecionado com sucesso. 7. Os níveis de dificuldade são apresentados com sucesso. 8. O sistema reconhece o comando de voz responsável pela seleção de um nível de dificuldade. 9. O nível de dificuldade é selecionado com sucesso. 10. Os exercícios são apresentados com sucesso. 11. O sistema reconhece o comando de voz responsável pela seleção de um exercício. 12. O exercício é selecionado com sucesso. 13. O sistema introduz o exercício e requisita dados advindos do instrumento do utilizador. 14. O sistema avalia o exercício com sucesso e oferece *feedback*.
Resultados	Todos os critérios foram cumpridos
Resultado Final	Aceite

A análise da tabela 7.3 permite concluir que, face aos critérios definidos e aos resultados obtidos, este teste de aceitação é aceite.

No Anexo B, encontram-se os restantes testes de aceitação desenvolvidos.

7.5.2 Inquérito de Satisfação

O Inquérito de Satisfação foi realizado a oito indivíduos que não possuem qualquer experiência musical, de modo a avaliar o nível de concordância de cada inquirido relativo à solução desenvolvida. Apesar de se considerar uma amostra reduzida, considerou-se relevante a utilização e avaliação destes dados, ainda que o tamanho da amostra se considere uma limitação, encontrando-se explicitada na secção 8.2.

O inquérito é constituído por seis questões de resposta fechada e encontra-se no Anexo C.

Cada questão do inquérito será avaliada de forma independente, com base em testes de hipóteses, tendo em consideração a hipótese relativa à satisfação dos utilizadores (c.f secção 7.2). Conforme definido na secção 7.3.4, nestes testes assume-se um nível de significância de 0.05, com um grau de confiança de 95%.

Uma vez obtidas as respostas ao inquérito, respostas estas que se encontram representadas na figura E.1 do Anexo E, utilizou-se o ambiente de desenvolvimento *RStudio* [112] como *software* de computação estatística de modo a realizar os testes de hipóteses. Todos os testes realizados neste *software* foram realizados na linguagem R. As respostas a cada questão constituem uma amostra independente, quantitativa e com uma dimensão de oito, que é referente ao número de inquiridos. Assim sendo, antes de qualquer aplicação de teste de hipóteses, torna-se necessário verificar se os dados seguem, ou não, uma distribuição normal. Dito isto, apresenta-se, de seguida, a avaliação de cada questão.

Questão 1 - A solução implementada permite um fluxo conversacional coerente?

Conforme supramencionado, o facto de a amostra existente ser de oito, implica a necessidade de verificar se os dados da amostra seguem uma distribuição normal. Assim, aplica-se o teste de normalidade *Shapiro-Wilk* que foi publicado em 1965 por Samuel Sanford Shapiro e Martin Wilk [113]. As hipóteses definidas foram as seguintes:

- H_0 - Os valores da amostra seguem uma distribuição normal.
- H_1 - Os valores da amostra não seguem uma distribuição normal.

No Excerto de Código 7.4 apresenta-se o teste de *Shapiro-Wilk* implementado.

```
1 > q1 <- c(4,4,5,4,5,5,4,5)
2 > shapiro.test(q1)
3
4   Shapiro-Wilk normality test
5
6 data:  q1
7 W = 0.66466, p-value = 0.0008917
```

Excerto de Código 7.4: Teste de normalidade *Shapiro-Wilk* aplicado à questão 1 do Inquérito de Satisfação

O Excerto de Código 7.4 é iniciado pela definição do vetor que contem as respostas escolhidas por todos os inquiridos à questão. Neste caso, o vetor obtido é o que se encontra representado na linha 1, sendo este: "q1 <- c(4,4,5,4,5,5,4,5)". De seguida, na linha 2, aplica-se o teste referido para que se obtenha o valor de *p-value* que, neste caso, é de 0.0008917. Como o valor de *p-value* obtido é menor do que o valor da significância definido (0.05), rejeita-se a hipótese H_0, podendo-se afirmar, assim, que a amostra não segue uma distribuição normal.

Sendo assim, não é possível aplicar-se um teste paramétrico à amostra (como, por exemplo, *t-test* [114]), pelo que se torna necessário aplicar um teste não paramétrico como o teste de *Wilcoxon* [115].

Considerando a hipótese previamente definida relativa à satisfação dos utilizadores, pretende-se que este valor seja superior a 80%. Assim sendo, converteu-se este valor para a escala utilizada no inquérito (escala de 1-5) e concluiu-se que a satisfação deveria ser superior ou igual ao valor 4. Assim sendo, definiram-se as seguintes hipóteses para o teste de *Wilcoxon*:

- H_0 - A média das respostas obtidas é inferior ou igual a 4.
- H_1 - A média das respostas obtidas é superior a 4.

A aplicação do teste de *Wilcoxon* à questão em avaliação encontra-se no Excerto de Código 7.5.

```
> wilcox.test(q1, mu = 4, alternative = "greater")

  Wilcoxon signed rank test with continuity correction

data:  q1
V = 10, p-value = 0.03593
alternative hypothesis: true location is greater than 4
```

Excerto de Código 7.5: Teste de *Wilcoxon* aplicado à questão 1 do Inquérito de Satisfação

A análise do Excerto de Código 7.5 permite concluir que o valor obtido de *p-value* foi 0.03595. Uma vez que este valor de *p-value* é inferior ao nível de significância, rejeita-se a hipótese H_0, permitindo concluir, com um grau de confiança de 95% que a concordância dos utilizadores a respeito da questão em análise é superior a 80%.

Questão 2 - A solução implementada facilita o processo de aprendizagem musical?

Nesta questão obteve-se o seguinte vetor de respostas: "q2 - c(4,5,4,5,4,5,5,4)". Uma vez que este vetor de respostas é idêntico ao analisado na questão 1, pode-se inferir que a amostra não segue uma distribuição normal. Da mesma forma, pode-se inferir que a utilização do teste de *Wilcoxon* concluirá o mesmo resultado pelo que se pode afirmar, com um grau de confiança de 95% que a satisfação dos utilizadores relativamente a esta questão é superior a 80%.

Questão 3 - O sistema reconhece o som do instrumento musical?

Todos os inquiridos escolheram a opção de resposta 5 no que diz respeito a esta questão, pelo que o vetor obtido foi: "q3 <- c(5,5,5,5,5,5,5,5)". Isto permite concluir que se está perante uma distribuição discreta, com o valor de probabilidade 1. Desta forma, considerando a hipótese de satisfação que se pretende atingir (nível de satisfação acima de 80%), e como todas as respostas obtidas tiveram o valor máximo, tal significa que a satisfação dos utilizadores nesta questão é superior a 80%.

Questão 4 - O *feedback* obtido é fulcral para a aprendizagem do módulo didático definido?

Conforme mencionado anteriormente, dada a dimensão da amostra, verifica-se a normalidade da distribuição através da utilização de teste *Shapiro-Wilk*, conforme Excerto de Código 7.6.

```
> q4 <- c(5,4,5,4,4,5,5,5)
> shapiro.test(q4)

  Shapiro-Wilk normality test

data:  q4
W = 0.6412, p-value = 0.0004791
```

Excerto de Código 7.6: Teste de normalidade *Shapiro-Wilk* aplicado à questão 4 do Inquérito de Satisfação

Uma vez que o valor de *p-value* obtido foi 0.0004791, sendo este menor do que o valor definido para o nível de significância, conclui-se que a amostra não segue uma distribuição normal. Deste modo, aplica-se o teste de *Wilcoxon*, conforme representado no Excerto de Código 7.7.

```
1 > wilcox.test(q4, mu = 4, alternative = "greater")
2
3   Wilcoxon signed rank test with continuity correction
4
5 data:  q4
6 V = 15, p-value = 0.01844
7 alternative hypothesis: true location is greater than 4
```

Excerto de Código 7.7: Teste de *Wilcoxon* aplicado à questão 4 do Inquérito de Satisfação

O Excerto de Código 7.7 indica que o valor de *p-value* é menor do que a significância definida, pelo que se concluiu que, com um grau de confiança de 95% que a satisfação nesta questão é superior a 80

Questão 5 - Tendo em conta o publico alvo do sistema desenvolvido, o sistema de integração de inquéritos é fundamental à realização dos mesmos?

Assim como a questão 3, nesta questão todas as respostas obtidas foram o valor máximo, pelo que se pode concluir que, à semelhança da questão 3, a satisfação dos utilizadores relativamente a esta questão é superior a 80%.

Questão 6 - A solução implementada promove um estilo de ensino musical autónomo?

O vetor de respostas obtido para esta questão foi: "q6 <- c(4,5,5,4,4,5,5,5)". Uma vez que este vetor de respostas é idêntico ao vetor obtido aquando da análise da questão 4, considera-se que a análise dos resultados a esta questão serão também semelhantes. Deste modo, considerando a hipótese definida na qual se indica que a satisfação deve ser superior a 80%, pode-se afirmar, com um grau de confiança de 95% que a hipótese é cumprida.

7.5.3 Inquérito de Usabilidade

O Inquérito de Usabilidade foi realizado aos mesmos indivíduos que responderam ao Inquérito de Satisfação. Assim como no Inquérito de Satisfação, considera-se a amostra obtida bastante reduzida. No entanto, mesmo que a extensão da amostra se encontre identificada como uma limitação, considera-se relevante proceder-se à análise das respostas obtidas.

O Inquérito de Usabilidade é composto por um guião, que é previamente proferido ao utilizador, e por doze questões de resposta fechada. Este encontra-se, na íntegra, anexado no Anexo D. Todas as respostas foram analisadas recorrendo ao mesmo *software* utilizado na análise das respostas do Inquérito de Satisfação e, à semelhança do Inquérito de Satisfação, nestes testes assume-se um nível de significância de 0.05, com um grau de confiança de 95%. A representação integral das respostas obtidas encontra-se no Anexo E, nas figuras E.2 e E.3.

Dada a extensão das questões existentes neste inquérito, nesta secção analisam-se apenas as questões que se consideram as mais relevantes no que diz respeito aos objetivos definidos (c.f secção 1.3).

À semelhança do que se referiu aquando da análise das questões do Inquérito de Usabilidade, considerando que a dimensão das amostras será oito, a análise de qualquer questão inicia-se pela utilização do teste *Shapiro-Wilk* de forma a verificar se as amostras seguem uma distribuição normal.

Questão 4 - Considera que, quando errou, o sistema lhe deu *feedback* de qualidade, ajudando-o a ultrapassar o erro?

No Excerto de Código 7.8, encontra-se o teste executado à normalidade da distribuição da amostra.

```
1 > q4 <- c(4,4,5,5,4,4,5,5)
2 > shapiro.test(q4)
3
4   Shapiro-Wilk normality test
5
6 data:  q4
7 W = 0.66466, p-value = 0.0008917
```

Excerto de Código 7.8: Teste de normalidade *Shapiro-Wilk* aplicado à questão 4 do Inquérito de Usabilidade

Pela análise do Excerto de Código 7.8 conclui-se que o vetor obtido foi: "q4 <- c(4,4,5,5,4,4,5,5)". O teste de *Shapiro-Wilk* sob este vetor originou um valor de *p-value* de 0.0008917, pelo que, por ser menor que o valor definido para a significância, nos permite concluir que a amostra não segue uma distribuição normal. Deste modo, aplica-se o teste de *Wilcoxon*, conforme representado no Excerto de Código 7.9.

As hipóteses definidas coincidem para todas as questões e são as seguintes:

- H_0 - A média das respostas obtidas é inferior ou igual a 4.
- H_1 - A média das respostas obtidas é superior a 4.

```
1 > wilcox.test(q4, mu = 4, alternative = "greater")
2
3   Wilcoxon signed rank test with continuity correction
4
5 data:  q4
6 V = 10, p-value = 0.03593
7 alternative hypothesis: true location is greater than 4
```

Excerto de Código 7.9: Teste de *Wilcoxon* aplicado à questão 4 do Inquérito de Usabilidade

A análise do Excerto de Código 7.9 permite concluir que o valor obtido de *p-value* foi 0.03593. Uma vez que este valor é inferior ao nível de significância (0.05), rejeita-se a hipótese H_0, permitindo concluir, com um grau de confiança de 95% que a concordância dos utilizadores a respeito da questão em análise é superior a 80%.

Questão 5 - Considera a experiência da interação com o sistema apelativa e imersiva?

No Excerto de Código 7.10, encontra-se o teste executado à normalidade da distribuição da amostra das respostas obtidas para esta questão.

```
> q5 <- c(4,4,5,4,5,5,5,5)
> shapiro.test(q5)

  Shapiro-Wilk normality test

data:  q5
W = 0.6412, p-value = 0.0004791
```

Excerto de Código 7.10: Teste de normalidade *Shapiro-Wilk* aplicado à questão 5 do Inquérito de Usabilidade

A aplicação do teste de *Shapiro-Wilk* permite concluir que o vetor obtido (q5 <- c(4,4,5,4,5,5,5,5)) não segue uma distribuição normal. Assim, aplica-se o teste de *Wilcoxon* recorrendo às mesmas hipóteses definidas para a questão 4.

```
> wilcox.test(q5, mu = 4, alternative = "greater")

  Wilcoxon signed rank test with continuity correction

data:  q5
V = 15, p-value = 0.01844
alternative hypothesis: true location is greater than 4
```

Excerto de Código 7.11: Teste de *Wilcoxon* aplicado à questão 5 do Inquérito de Usabilidade

O teste aplicado no Excerto de Código 7.11, permite concluir que, à semelhança da questão 4, o valor de *p-value* é menor que o valor da significância, rejeitando-se assim a hipótese H_0. Conclui-se então que se pode afirmar com um grau de confiança de 95% que a concordância dos utilizadores nesta questão é, também, superior a 80%.

Questão 6 - Considera que o sistema reconhece corretamente os seus comandos de voz?

No Excerto de Código 7.12 encontra-se o teste executado à normalidade da distribuição.

```
> q6 <- c(3,4,5,5,5,5,5,5)
> shapiro.test(q6)

  Shapiro-Wilk normality test

data:  q6
W = 0.60128, p-value = 0.0001646
```

Excerto de Código 7.12: Teste de normalidade *Shapiro-Wilk* aplicado à questão 6 do Inquérito de Usabilidade

À semelhança da distribuição das outras questões, esta também não segue uma distribuição normal, pelo que se aplica de seguida o teste de *Wilcoxon* recorrendo às mesmas hipóteses definidas para as questões anteriores.

```
> wilcox.test(q6, mu = 4, alternative = "greater")

  Wilcoxon signed rank test with continuity correction

data:  q6
V = 24, p-value = 0.0363
alternative hypothesis: true location is greater than 4
```

Excerto de Código 7.13: Teste de *Wilcoxon* aplicado à questão 6 do Inquérito de Usabilidade

A análise do teste permite-nos concluir que o valor de *p-value* obtido é menor do que a significância definida, pelo que se conclui que se pode rejeitar a hipótese H_0. Isto permite concluir que, com um grau de confiança de 95%, a concordância dos utilizadores nesta questão é, também, superior a 80%.

Questão 8 - Considera simples o processo de escolha dos módulos didáticos?

O vetor de resultados obtidos para esta questão é idêntico ao vetor obtido aquando da análise da questão 5. Assim sendo, permite-nos tirar conclusões análogas relativamente a esta questão. Deste modo, conclui-se que a concordância dos utilizadores nesta questão é superior a 80%, com um grau de confiança de 95%.

Capítulo 8

Conclusão

Neste capítulo apresenta-se a conclusão final da dissertação. Para isto, descrevem-se os objetivos alcançados, as limitações existentes e, por último, o trabalho futuro a desenvolver.

8.1 Objetivos Alcançados

Para ter sucesso no desenvolvimento de um sistema relevante e coerente, seguiram-se alguns passos que se consideram imprescindíveis. Iniciou-se o projeto pela definição do problema, seguido pela especificação dos objetivos propostos para o colmatar. Estes objetivos foram considerados o fio condutor da dissertação, pelo que tudo que se desenvolveu os teve em consideração. De modo a identificar, à partida, possíveis incongruências e más decisões, efetuou-se uma análise de valor da qual se depreendeu a validade da ideia. Seguidamente, identificaram-se e analisaram-se os requisitos funcionais e não funcionais da solução que resultaram na modelação de uma arquitetura que foi posteriormente detalhada e implementada. De forma a qualificar a conclusão dos objetivos, procedeu-se à avaliação da solução seguindo uma metodologia baseada em testes de *software*, testes de hipóteses e inquéritos.

Nesta secção pretende-se concluir sobre o estado final dos objetivos definidos na secção 1.3. Assim, os três objetivos definidos são os que se seguem:

- Investigar e analisar uma abordagem de ensino musical que não seja dependente do sentido da visão, potenciando e agilizando o ensino autónomo do instruendo.
- Desenvolver um sistema baseado na abordagem de ensino investigada capaz de proporcionar um ensino imersivo, apelativo, eficaz e autónomo, independente do sentido da visão.
- Avaliar a eficácia do sistema baseada em dados de utilização reais.

A investigação e a exploração que deram forma ao capitulo do Estado de Arte (c.f 2.6) permitiram a identificação de soluções já existentes que apoiam o ensino tradicional (c.f secção 2.1) e de soluções que utilizam a tecnologia de forma a auxiliar o ensino tradicional (c.f secção 2.2). No seguimento dessa análise, verificou-se que o conceito referente a assistentes digitais está a emergir (c.f secção 2.3) e identificou-se que, apesar de existirem vários assistentes digitais já desenvolvidos por empresas influentes e de grande referência, por razões de segurança nenhum deles permite a integração de algoritmos de análise de qualquer outro som que não comandos de voz. Deste modo, identificou-se a hipótese de desenvolver um assistente digital que, para além de conter todos os módulos inerentes a qualquer assistente já existente (c.f 2.3.1), permitisse a integração com módulos de reconhecimento de som (e.g. frequências advindas de instrumentos musicais). Após identificar

e analisar a oportunidade encontrada (c.f secção 3.1.2), concluiu-se que seria relevante este desenvolvimento. Assim, considera-se que se atingiu com sucesso o primeiro objetivo dada a identificação desta abordagem de ensino musical baseada na utilização de um assistente digital integrado com algoritmos de análise de frequências sonoras.

Uma vez identificada a abordagem de ensino referente ao primeiro objetivo, selecionou-se a ideia alusiva ao desenvolvimento de um assistente digital integrado com um módulo de análise de som. Este, para que fosse desenvolvido, foi inicialmente alvo de uma análise na qual se identificaram todas as entidades de negócio num modelo de domínio (c.f secção 4.1), assim como todos os requisitos funcionais (c.f secção 4.2) e os requisitos não funcionais (c.f secção 4.3) que o constituem. Após a definição dos requisitos, modelou-se de forma detalhada a arquitetura a desenvolver, identificando todos os componentes existentes e as suas interações. Com tudo isto, tornou-se possível implementar o assistente digital de modo a que integrasse a abordagem de ensino previamente estudada. Todas as tecnologias utilizadas na fase de implementação foram previamente comparadas e a sua escolha justificada, pelo que se pode concluir que é válido desenvolver por completo um assistente digital, com todos os seus módulos envolventes, apenas utilizando tecnologias *open-source*. Para além disto, conclui-se ainda que o assistente digital foi completamente desenvolvido com sucesso, contendo integração com todas as tecnologias que perfazem os módulos existentes num assistente digital, assim como o módulo de análise de som. Assim, conseguiu-se desenvolver um sistema que reconhece os comandos de voz proferidos pelo utilizador, que entende a linguagem natural identificada, que controla o diálogo para que retornem respostas contextualizadas e coerentes que serão convertidas para linguagem natural e, finalmente, sintetizadas em voz. Paralelamente a estes módulos, desenvolveu-se ainda um sistema de análise de som que coexiste com os módulos anteriormente apresentados e que tem como objetivo analisar o som advindo de um instrumento musical (e.g. Guitarra). Dito isto, todos os requisitos funcionais e não funcionais foram analisados, desenhados, implementados e avaliados com sucesso. Esta avaliação seguiu a metodologia do modelo *V-model* definido na secção 7.3.1, que agrega a realização de testes unitários, testes de integração, testes de sistema e testes de aceitação. A realização destes testes permite concluir que se cumpriu com sucesso o segundo objetivo na medida em que se encontra justificado cumprimento de todos os requisitos definidos.

Por último, o terceiro objetivo visa avaliar a eficácia do sistema com base em dados de utilização reais. Para que este objetivo se tornasse mensurável, definiram-se dois requisitos funcionais, conforme apresentados na secção 4.2: UC6 - Preencher Inquérito de Satisfação; UC7 - Preencher Inquérito de Usabilidade. Estes dois requisitos têm como finalidade o desenvolvimento de um módulo de inquéritos que se encontre integrado no Assistente Digital implementado para obtenção de respostas quanto à Satisfação e à Usabilidade dos inquiridos. Considerando o objetivo social que este projeto pretende endereçar, esta integração é determinante na medida em que permite a utilização e o preenchimento autónomo por parte de indivíduos com deficiência visual. Assim sendo, uma vez analisados os requisitos, estes foram implementados e testados com sucesso, recorrendo à metodologia de testes de *software* definida na secção 7.3.1. Dito isto, este módulo de inquéritos desenvolvido permitiu recolher algumas respostas, sendo que as amostras foram avaliadas com sucesso nas secções 7.5.2 e 7.5.3. Todos os resultados obtidos, sejam eles relativos à satisfação ou à usabilidade, encontram-se acima da taxa de 80% definida, pelo que se denota uma tendência positiva na conclusão deste objetivo. No entanto, apesar de se ter implementado e testado com sucesso o módulo dos inquéritos, tal apenas permite concluir que os requisitos funcionais foram alcançados neste aspeto. Como se considera que as amostras não têm uma dimensão

muito relevante, apesar da análise positiva que se atingiu com as amostras obtidas, dá-se este objetivo como inconclusivo, sendo este um objetivo alvo de trabalho futuro.

8.2 Limitações

Apesar da implementação ter sido bem sucedida e da avaliação do sistema desenvolvido ter sido positiva, identificaram-se algumas limitações que poderão ser alvo de trabalho futuro.

Os inquéritos efetuados com o intuito de analisar o sucesso da aplicação, foram aplicados a um conjunto de pessoas sem qualquer deficiência visual. Tendo em conta o público alvo deste programa, os resultados seriam mais precisos e fiáveis se os dados levantados se concentrassem num aglomerado de pessoas com incapacidade visual. Para além disto, a amostra das respostas obtidas foi algo limitativa no que diz respeito à sua dimensão, tanto para os inquéritos de satisfação, como para os inquéritos de usabilidade, pelo que amostras de maior dimensão conferiam mais coesão e fiabilidade à avaliação executada.

A aplicação é constituída, praticamente no seu todo, por mecanismos de reconhecimento de voz e som (frequências emitidas pelo instrumento musical). Para o seu correto funcionamento, o ambiente em que é utilizada não pode ser ruidoso, uma vez que o sistema não consegue decifrar o que deve captar e o que deve descartar. Esta limitação pode ser ultrapassada através da utilização de um dispositivo que faça a ligação, sem perdas, entre o instrumento musical e o sistema. A utilização de uma interface de áudio *Universal Serial Bus (USB)* capaz de converter sinal analógico em sinal digital é um bom exemplo de um sistema mais fiável.

Contudo, mesmo em ambientes sem ruído, persistem perdas de sinal se o sistema não fizer uso da interface referida acima, pelo que, frequências mais baixas, para notas mais graves, por vezes são erradamente reconhecidas pelo sistema desenvolvido devido a limitações físicas relacionadas com o *hardware built-in* de captação de som (microfones) nos computadores comuns.

8.3 Trabalho Futuro

Face ao trabalho desenvolvido e às limitações identificadas, considera-se que existe uma vasta quantidade de tarefas que visam melhorar a solução apresentada.

Apesar de não ter sido definido como sendo um objetivo deste projeto, a interface pode ser alvo de mudanças, de forma a enriquecer a comunicação entre o utilizador e o sistema. De momento a interface existente funciona puramente através de voz, pelo que se considera que seria interessante desenvolver uma interface que, integrada com sistemas de *hardware* específicos, potenciasse a estimulação de outros sentidos do utilizador (como por exemplo o tato através de vibrações).

Existem ainda várias alterações ao nível do algoritmo de análise de som que, ainda que não sejam parte integra dos objetivos propostos, melhorariam a experiência do utilizador e a utilidade do projeto. O tratamento do ruído, apesar de ter sido abordado e tratado superficialmente (c.f secção 6.2.3), não é ainda suficiente pelo que o sistema apenas se encontra preparado para ignorar ruído quando este existe isolado num certo excerto de som captado, não estando preparado para distinguir, num mesmo sinal, o que deve ser descartado por conter ruído e o que deve ser mantido por se considerar relevante com informações advindas do instrumento do utilizador.

A existência de suporte a um módulo didático é interessante na medida em que serve de prova de conceito, demonstrando o poder de um assistente digital integrado com um sistema de análise de som. No entanto, seria pertinente proceder-se à expansão do suporte dos módulos didáticos, incluindo suporte a outros instrumentos, para que agregue mais interesse do público ao qual se destina. Para além desta expansão, seria ainda interessante a implementação de um módulo cuja responsabilidade fosse dar *feedback* relativamente a uma peça musical executada pelo utilizador, complementando o sistema de identificação de notas soltas e oitavas implementado.

No que diz respeito ao sistema de seleção de exercícios a executar, considera-se que este deva ser aprimorado de modo a que infira de forma automática o nível de dificuldade do utilizador, podendo adaptar em tempo real os exercícios propostos. Desta forma, evitava-se o passo, outrora necessário, de seleção dos exercícios a executar, propondo-os automaticamente e com um propósito sistemático baseado na evolução do utilizador.

Relativamente aos inquéritos desenvolvidos, é pertinente investir na recolha de mais testemunhos para que se aumente a dimensão da amostra com a qual se trabalha a avaliação e a análise. Quanto ao inquérito de usabilidade, apesar de se considerar que o inquérito desenvolvido e apresentado aos inquiridos é bastante relevante, entende-se que seria interessante complementar esta avaliação de usabilidade com um módulo que avaliasse a usabilidade com base na eficácia advinda de uma análise à taxa de realização com sucesso das tarefas requisitadas [116]. Deste modo, a análise de usabilidade tornar-se-ia menos subjetiva e mais eficaz.

Bibliografia

[1] *Technology in Everyday Life | JFG Inc*. URL: http://jfg-nc.com/technology-in-everyday-life/ (acedido em 09/02/2020).

[2] L. Schoenberger e C. Braswell, «Music therapy in rehabilitation.», *Journal of Rehabilitation*, vol. 37, n.º 1, pp. 30–31, jan. de 1971, issn: 00224154. doi: 10.26444/jpccr/71428.

[3] *(PDF) Effect of Music Therapy on Anxiety and Depression in Patients with Alzheimer's Type Dementia: Randomised, Controlled Study*. (acedido em 09/02/2020).

[4] S. Paul e D. Ramsey, *Music therapy in physical medicine and rehabilitation*, 2000. doi: 10.1046/j.1440-1630.2000.00215.x.

[5] A. Antonietti, «Why is music effective in rehabilitation?», em *Studies in Health Technology and Informatics*, vol. 145, IOS Press, 2009, pp. 179–194.

[6] *From mechanical to digital instrument : music and new technologies | Request PDF*. (acedido em 10/02/2020).

[7] *Individuals using the Internet (% of population) | Data*. URL: https://data.worldbank.org/indicator/IT.NET.USER.ZS (acedido em 08/02/2020).

[8] M. Roser, H. Ritchie e E. Ortiz-Ospina, «Internet», *Our World in Data*, jul. de 2015.

[9] *What is the Digital Divide? | Digital Divide Council*. (acedido em 11/02/2020).

[10] *What is AT? - Assistive Technology Industry Association*. URL: https://www.atia.org/at-resources/what-is-at/ (acedido em 11/02/2020).

[11] D. Crombie, S. Dijkstra, E. Schut e N. Lindsay, «Spoken Music: Enhancing Access to Music for the Print Disabled», em, 2002, pp. 667–674. doi: 10.1007/3-540-45491-8_129.

[12] B. Authority of North America, *MUSIC BRAILLE CODE, 2015 Developed Under the Sponsorship of the BRAILLE AUTHORITY OF NORTH AMERICA*.

[13] *Toccata*. URL: http://www.pentronics.com.au/index_files/Toccata.htm (acedido em 14/02/2020).

[14] *Projeto MusiBraille*. URL: http://intervox.nce.ufrj.br/musibraille/ (acedido em 20/02/2020).

[15] *GOODFEEL Braille Music Translator: Braille Music Converter - Dancing Dots*. URL: http://www.dancingdots.com/main/goodfeel.htm (acedido em 14/02/2020).

[16] *Computers Helping People with Special Needs: 13th International Conference - Google Livros*.

[17] A. Capozzi, R. De Prisco, M. Nasti e R. Zaccagnino, «Musica parlata : AAA methodology to teach music to blind people», em *ASSETS'12 - Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*, 2012, pp. 245–246.

[18] T. Haenselmann, H. Lemelson, K. Adam e W. Effelsberg, «A tangible MIDI sequencer for visually impaired people», em *MM'09 - Proceedings of the 2009 ACM Multimedia Conference, with Co-located Workshops and Symposiums*, 2009, pp. 993–994.

[19] S. Omori e I. E. Yairi, «Collaborative music application for visually impaired people with tangible objects on table», em *Proceedings of the 15th International ACM*

SIGACCESS Conference on Computers and Accessibility, ASSETS 2013, 2013. doi: 10.1145/2513383.2513403.

[20] *Fabio's Story | Something Money Can't Buy*. URL: https://www.bemyeyes.com/community-stories/something-money-cant-buy (acedido em 03/02/2020).

[21] *Amir's Story | In the Workplace and at the Piano*. URL: https://www.bemyeyes.com/community-stories/in-the-workplace-and-at-the-piano (acedido em 03/02/2020).

[22] *Be My Eyes - Bringing sight to blind and low-vision people*. URL: https://www.bemyeyes.com/ (acedido em 03/02/2020).

[23] *What Is a Digital Assistant? | Oracle*. URL: https://www.oracle.com/solutions/chatbots/what-is-a-digital-assistant.html (acedido em 12/02/2020).

[24] *The Decade of Voice Assistant Revolution - Voicebot.ai*. URL: https://voicebot.ai/2019/12/31/the-decade-of-voice-assistant-revolution/ (acedido em 12/02/2020).

[25] *Infographic: The bot platform ecosystem - O'Reilly Media*. URL: https://www.oreilly.com/ideas/infographic-the-bot-platform-ecosystem (acedido em 12/02/2020).

[26] *Google Trends*. URL: https://trends.google.pt/trends/?geo=PT (acedido em 13/02/2020).

[27] *voice assistant - Explorar - Google Trends*. URL: https://trends.google.com/trends/explore?date=today%205-y&q=voice%20assistant (acedido em 16/02/2020).

[28] *Speech Synthesis: Giving Machines a Voice*. URL: https://www.celi.it/en/blog/2017/01/speech-synthesis-giving-machines-a-voice/ (acedido em 13/02/2020).

[29] J. Levis e R. Suvorov, «Automatic Speech Recognition», em *The Encyclopedia of Applied Linguistics*, N 18, vol. V 15, Oxford, UK: Blackwell Publishing Ltd, nov. de 2012. doi: 10.1002/9781405198431.wbeal0066. URL: http://doi.wiley.com/10.1002/9781405198431.wbeal0066.

[30] *Natural-language Understanding*. URL: https://www.gartner.com/en/information-technology/glossary/nlu-natural-language-understanding (acedido em 14/02/2020).

[31] D. Goddeau, H. Meng, J. Polifroni, S. Seneff e S. Busayapongchai, «Form-based dialogue manager for spoken language applications», em *International Conference on Spoken Language Processing, ICSLP, Proceedings*, vol. 2, IEEE, 1996, pp. 701–704. doi: 10.1109/icslp.1996.607458.

[32] *What is natural language generation?* URL: https://narrativescience.com/what-is-natural-language-generation/ (acedido em 14/02/2020).

[33] *Amazon: Alexa devices were best-selling products from any manufacturer*. URL: https://marketingland.com/amazon-alexa-devices-best-selling-products-manufacturer-199446 (acedido em 17/05/2020).

[34] R. Prasad, «Alexa Everywhere», em *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, New York, NY, USA: Association for Computing Machinery (ACM), jan. de 2019, pp. 3–3. doi: 10.1145/3289600.3291377. URL: https://dl.acm.org/doi/10.1145/3289600.3291377.

[35] A. Purington, J. G. Taft, S. Sannon, N. N. Bazarova e S. H. Taylor, «"Alexa is my new BFF": Social roles, user satisfaction, and personification of the Amazon Echo», em *Conference on Human Factors in Computing Systems - Proceedings*, vol. Part

F127655, Association for Computing Machinery, mai. de 2017, pp. 2853–2859, isbn: 9781450346566. doi: 10.1145/3027063.3053246.

[36] *Voice Design Best Practices (Legacy) | Alexa Skills Kit*. URL: https://developer.amazon.com/en-US/docs/alexa/custom-skills/voice-design-best-practices-legacy.html (acedido em 17/05/2020).

[37] *Web Speech API - Web APIs | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API (acedido em 18/02/2020).

[38] *Cloud Text-to-Speech – síntese de fala | API Cloud Text-to-Speech*. URL: https://cloud.google.com/text-to-speech (acedido em 18/02/2020).

[39] *Amazon Polly*. URL: https://aws.amazon.com/pt/polly/ (acedido em 18/02/2020).

[40] *Dialogflow | Google Cloud*. URL: https://cloud.google.com/dialogflow (acedido em 19/02/2020).

[41] *AWS Lex - Amazon Web Services*. URL: https://aws.amazon.com/pt/lex/ (acedido em 19/02/2020).

[42] *Web Audio API - Web APIs | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web%7B%5C_%7DAudio%7B%5C_%7DAPI (acedido em 10/08/2020).

[43] *howler.js - JavaScript audio library for the modern web*. URL: https://howlerjs.com/ (acedido em 10/08/2020).

[44] *<audio>: The Embed Audio element - HTML: HyperText Markup Language | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio (acedido em 10/08/2020).

[45] A. Pano, D. Graziotin e P. Abrahamsson, «Factors and actors leading to the adoption of a JavaScript framework», mai. de 2016. arXiv: 1605.04303. URL: http://arxiv.org/abs/1605.04303.

[46] D. Graziotin e P. Abrahamsson, «Making sense out of a jungle of JavaScript frameworks: Towards a practitioner-friendly comparative analysis», em *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7983 LNCS, 2013, pp. 334–337, isbn: 9783642392580. doi: 10.1007/978-3-642-39259-7_28.

[47] *Stack Overflow - Where Developers Learn, Share, & Build Careers*. URL: https://stackoverflow.com/ (acedido em 23/05/2020).

[48] C. Larman, «Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)», p. 656, 2001. URL: http://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented/dp/0130925691.

[49] F. F. Correia, H. S. Ferreira, A. Aguiar e N. Flores, «Patterns for consistent software documentation», em *ACM International Conference Proceeding Series*, New York, New York, USA: ACM Press, 2010, p. 1, isbn: 9781605588735. doi: 10.1145/1943226.1943241. URL: http://portal.acm.org/citation.cfm?doid=1943226.1943241.

[50] *Speech API Community Group*. URL: https://www.w3.org/community/speech-api/ (acedido em 20/02/2020).

[51] K. B. Kahn, S. E. Kay, R. J. Slotegraaf e S. Uban, «THE PDMA HANDBOOK OF NEW PRODUCT DEVELOPMENT THIRD EDITION», rel. téc.

[52] P. Smith e D. Reinertsen, «Developing products in half the time», *Choice Reviews Online*, vol. 28, n.º 10, jun. de 1991, issn: 0009-4978. doi: 10.5860/choice.28-5737.

[53] *RNIB - See differently* -. URL: https://www.rnib.org.uk/ (acedido em 16/02/2020).

[54] J. Slade e R. Edwards, «My Voice 2015 – The views and experiences of blind and partially sighted people in the UK Version 1.1», rel. téc., 2015.

[55] *digital assistant - Explorar - Google Trends*. URL: https://trends.google.com/trends/explore?date=today%205-y&q=digital%20assistant (acedido em 16/02/2020).

[56] *FAQ about Google Trends data - Trends Help*. URL: https://support.google.com/trends/answer/4365533?hl=en (acedido em 16/02/2020).

[57] T. L. Saaty, *The analytic hierarchy process : planning, priority setting, resource allocation*. McGraw-Hill International Book Co, 1980, p. 287.

[58] *Value Proposition Canvas – Download the Official Template*. URL: https://www.strategyzer.com/canvas/value-proposition-canvas (acedido em 12/01/2020).

[59] A. OSTERWALDER, *The Business Model Ontology a Proposition in a Design Science Approach*. 2004.

[60] A. Osterwalder, Y. Pigneur, A. Smith e T. Movement, *Defintion of Business models*, 5377. 2010, vol. 30, p. 288. doi: 10.1523/JNEUROSCI.0307-10.2010. arXiv: arXiv:1011.1669v3.

[61] *O que fazemos | ACAPO*. URL: http://www.acapo.pt/o-que-fazemos (acedido em 20/02/2020).

[62] *System Sequence Diagrams in UML | Lucidchart*. URL: https://www.lucidchart.com/pages/uml-system-sequence-diagram (acedido em 21/02/2020).

[63] P. Ralph e Y. Wand, «A proposal for a formal definition of the design concept», em *Lecture Notes in Business Information Processing*, vol. 14 LNBIP, Springer Verlag, 2009, pp. 103–136, isbn: 3540929657. doi: 10.1007/978-3-540-92966-6_6.

[64] *What is Component Diagram?* URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/ (acedido em 23/02/2020).

[65] *What Every Engineer Should Know about Software Engineering - Philip A. Laplante - Google Books*.

[66] D. E. Perry e A. L. Wolf, «Foundations for the study of software architecture», *ACM SIGSOFT Software Engineering Notes*, vol. 17, n.º 4, pp. 40–52, out. de 1992, issn: 0163-5948. doi: 10.1145/141874.141884. URL: https://dl.acm.org/doi/10.1145/141874.141884.

[67] *Webhooks*. URL: https://developer.atlassian.com/server/jira/platform/webhooks/ (acedido em 06/06/2020).

[68] *What is Platform as a Service (PaaS)?* URL: https://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS (acedido em 30/06/2020).

[69] *What is DBaaS (Database-as-a-Service)? | IBM*. URL: https://www.ibm.com/cloud/learn/dbaas (acedido em 30/06/2020).

[70] *MongoDB Hosting: Database-as-a-Service by mLab*. URL: https://mlab.com/ (acedido em 30/06/2020).

[71] *UML sequence diagrams overview of graphical notation*. URL: https://www.uml-diagrams.org/sequence-diagrams.html/interaction-use (acedido em 04/07/2020).

[72] *Databases and Collections — MongoDB Manual*. URL: https://docs.mongodb.com/manual/core/databases-and-collections/ (acedido em 10/09/2020).

[73] *Firebase Realtime Database*. URL: https://firebase.google.com/docs/database (acedido em 12/09/2020).

[74] «Java Code Conventions», rel. téc., 1997.

[75] *Entities | Dialogflow Documentation | Google Cloud*. URL: https://cloud.google.com/dialogflow/docs/entities-overview (acedido em 13/08/2020).
[76] *Inline editor | Dialogflow Documentation | Google Cloud*. URL: https://cloud.google.com/dialogflow/docs/fulfillment-inline-editor (acedido em 14/08/2020).
[77] *Cloud Functions Overview | Cloud Functions Documentation*. URL: https://cloud.google.com/functions/docs/concepts/overview (acedido em 15/08/2020).
[78] *GitHub - axios/axios: Promise based HTTP client for the browser and node.js*. URL: https://github.com/axios/axios (acedido em 15/08/2020).
[79] *Aerotwist - Guitar Tuner*. URL: https://aerotwist.com/blog/guitar-tuner/ (acedido em 04/10/2020).
[80] *AnalyserNode.getFloatTimeDomainData() - Web APIs | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode/getFloatTimeDomainData (acedido em 04/10/2020).
[81] *What is Root Mean Square?* URL: https://www.sweetwater.com/insync/rms-root-mean-square/ (acedido em 04/10/2020).
[82] *Octave | music | Britannica*. URL: https://www.britannica.com/art/octave-music (acedido em 05/10/2020).
[83] *AudioContextOptions.sampleRate - Web APIs | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/API/AudioContextOptions/sampleRate (acedido em 05/10/2020).
[84] *Number.POSITIVE_INFINITY – JavaScript|MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/POSITIVE_INFINITY (acedido em 08/10/2020).
[85] *ISO - ISO 16:1975 - Acoustics — Standard tuning frequency (Standard musical pitch)*. URL: https://www.iso.org/standard/3601.html (acedido em 08/10/2020).
[86] *Formula for frequency table*. URL: https://pages.mtu.edu/~suits/NoteFreqCalcs.html (acedido em 08/10/2020).
[87] *Frequencies of Musical Notes, A4 = 440 Hz*. URL: https://pages.mtu.edu/~suits/notefreqs.html (acedido em 09/10/2020).
[88] *Intervals and Pitch*. URL: http://thesoundofnumbers.com/wp-content/uploads/2014/11/pitch_intervals_freq.pdf (acedido em 09/10/2020).
[89] «Intervals and Pitch», rel. téc.
[90] *Express - Node.js web application framework*. URL: https://expressjs.com/ (acedido em 18/08/2020).
[91] *Express Tutorial Part 4: Routes and controllers - Learn web development | MDN*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes (acedido em 18/08/2020).
[92] *What is an Object Data Model? - Definition from Techopedia*. URL: https://www.techopedia.com/definition/30736/object-data-model (acedido em 31/08/2020).
[93] *Express Tutorial Part 3: Using a Database (with Mongoose) - Learn web development | MDN*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose (acedido em 31/08/2020).
[94] *Mongoose ODM v5.10.0*. URL: https://mongoosejs.com/ (acedido em 18/08/2020).
[95] *Mongoose v5.10.2 - Model*. URL: https://mongoosejs.com/docs/api/model.html (acedido em 02/09/2020).
[96] *What is an API Endpoint? | API Endpoint Definition | RapidAPI*. URL: https://rapidapi.com/blog/api-glossary/endpoint/ (acedido em 18/08/2020).

[97] *Customer Satisfaction - How to Measure Satisfaction of Customers*. URL: https://corporatefinanceinstitute.com/resources/knowledge/other/measuring-customer-satisfaction/ (acedido em 11/10/2020).

[98] *Software Testing Tutorial - Tutorialspoint*. URL: https://www.tutorialspoint.com/software_testing/index.htm (acedido em 16/09/2020).

[99] *V Model SDLC: Verification and Validation Model |Professionalqa.com*. URL: https://www.professionalqa.com/v-model (acedido em 16/09/2020).

[100] A. Joshi, S. Kale, S. Chandel e D. Pal, «Likert Scale: Explored and Explained», *British Journal of Applied Science Technology*, vol. 7, pp. 396–403, jan. de 2015. doi: 10.9734/BJAST/2015/14975.

[101] *System Usability Scale (SUS) | Usability.gov*. URL: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (acedido em 03/10/2020).

[102] *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (acedido em 02/10/2020).

[103] J. de Sá, *Applied Statistics Using SPSS, STATISTICA, MATLAB and R*. Springer Berlin Heidelberg, 2009, isbn: 9783540837435. URL: https://books.google.pt/books?id=9Nf0vQAACAAJ.

[104] *Unit Testing: Tutorial, Types, Frameworks,Process,Techniques |Professionalqa*. URL: https://www.professionalqa.com/unit-testing (acedido em 23/09/2020).

[105] *Mocha - test framework*. URL: https://mochajs.org/ (acedido em 24/09/2020).

[106] *Expect / Should - Chai*. URL: https://www.chaijs.com/api/bdd/ (acedido em 24/09/2020).

[107] *Arrange Act Assert | Telerik*. URL: https://docs.telerik.com/devtools/justmock/basic-usage/arrange-act-assert (acedido em 01/10/2020).

[108] *Istanbul, a JavaScript test coverage tool.* URL: https://istanbul.js.org/ (acedido em 25/09/2020).

[109] *What is an Integration Testing? |Professionalqa.com*. URL: https://www.professionalqa.com/integration-testing (acedido em 23/09/2020).

[110] *System Testing |Professionalqa.com*. URL: https://www.professionalqa.com/system-testing (acedido em 25/09/2020).

[111] *What is User Acceptance Testing? |Professionalqa.com*. URL: https://www.professionalqa.com/user-acceptance-testing (acedido em 25/09/2020).

[112] *R: The R Project*. URL: https://www.r-project.org/ (acedido em 11/10/2020).

[113] S. S. Shapiro e M. B. Wilk, «An Analysis of Variance Test for Normality (Complete Samples)», *Biometrika*, vol. 52, n.º 3/4, pp. 591–611, 1965, issn: 00063444. URL: http://www.jstor.org/stable/2333709.

[114] STUDENT, «THE PROBABLE ERROR OF A MEAN», *Biometrika*, vol. 6, n.º 1, pp. 1–25, mar. de 1908, issn: 0006-3444. doi: 10.1093/biomet/6.1.1. eprint: https://academic.oup.com/biomet/article-pdf/6/1/1/605641/6-1-1.pdf. URL: https://doi.org/10.1093/biomet/6.1.1.

[115] F. Wilcoxon, «Individual Comparisons by Ranking Methods», *Biometrics Bulletin*, vol. 1, n.º 6, pp. 80–83, 1945, issn: 00994987. URL: http://www.jstor.org/stable/3001968.

[116] *Usability Metrics - A Guide To Quantify The Usability Of Any System*. URL: https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/ (acedido em 15/10/2020).

Anexo A

Testes de Sistema

Neste Anexo, apresentam-se os restantes testes de sistema realizados ao projeto total desenvolvido. Assim, apresentam-se os testes de sistema nas tabelas A.1, A.2, A.3, A.4, A.5, A.6.

Tabela A.1: Teste de Sistema para Definir Nível de Dificuldade

Caso de Uso	UC2
Título	Definir Nível de Dificuldade
Cenário	Os níveis de dificuldade são requisitados por parte do utilizador, uma vez definido o módulo didático pretendido. Após definição do nível de dificuldade, o sistema requisita definição de exercício.
Procedimento	Após definição do módulo didático pretendido, o utilizador requisita definição de nível de dificuldade através de um comando de voz que contenha as palavras chave referentes a qualquer um dos níveis apresentados. O sistema apresenta os níveis de dificuldade existentes para o módulo didático definido. O utilizador escolhe o nível de dificuldade que pretende e o sistema requisita definição de exercício.
Critérios	1. Os níveis de dificuldade são apresentados ao utilizador. 2. O sistema reconhece o pedido advindo do comando de voz para que se selecione o nível de dificuldade pretendido. 3. O nível de dificuldade é selecionado com sucesso.
Resultados	1. Passou 2. Passou 3. Passou
Resultado Final	Passou

Tabela A.2: Teste de Sistema para Requisitar Exercício

Caso de Uso	UC3
Título	Requisitar Exercício
Cenário	Os exercícios são requisitados por parte do utilizador, uma vez definido o nível de dificuldade pretendido. Após definição do exercício, o sistema requisita a sua realização.
Procedimento	Após definição do nível de dificuldade pretendido, o utilizador requisita definição de exercício através de um comando de voz que contenha as palavras chave referentes a qualquer um dos exercícios apresentados. O sistema apresenta os exercícios existentes para o nível de dificuldade definido. O utilizador escolhe o exercício que pretende e o sistema requisita a sua realização.
Critérios	1. Os exercícios são apresentados ao utilizador. 2. O sistema reconhece o pedido advindo do comando de voz para que se selecione o exercício pretendido. 3. O exercício é selecionado com sucesso.
Resultados	1. Passou 2. Passou 3. Passou
Resultado Final	Passou

Tabela A.3: Teste de Sistema para Realizar Exercício

Caso de Uso	UC4
Título	Realizar Exercício
Cenário	Uma vez definido o exercício pretendido, o sistema requisita a sua realização.
Procedimento	Após definição do exercício a realizar, o sistema irá solicitar a realização do exercício selecionado. O utilizador realiza o exercício com sucesso. O sistema reconhece a realização e avalia o exercício (UC8).
Critérios	1. O sistema solicita realização do exercício. 2. A realização do exercício é avaliada corretamente (UC8).
Resultados	1. Passou 2. Passou
Resultado Final	Passou

Tabela A.4: Teste de Sistema para Requisitar Ajuda

Caso de Uso	UC5
Título	Requisitar Ajuda
Cenário	A qualquer momento da utilização do Assistente Digital, o utilizador pode requisitar ajuda.
Procedimento	Uma vez iniciada a execução do Assistente Digital, o utilizador requisita ajuda.
Critérios	1. O sistema reconhece o pedido advindo do comando de voz para que se apresenta comandos de ajuda. 2. O módulo de ajuda é apresentado com sucesso.
Resultados	1. Passou 2. Passou
Resultado Final	Passou

Tabela A.5: Teste de Sistema para Preencher Inquérito de Satisfação

Caso de Uso	UC6
Título	Preencher Inquérito de Satisfação
Cenário	A qualquer momento da utilização do Assistente Digital, o utilizador pode preencher inquérito de satisfação.
Procedimento	Uma vez iniciada a execução do Assistente Digital, o utilizador preenche inquérito de satisfação. Para isto, o sistema apresenta as questões, requisitando respostas por parte do utilizador a cada pergunta individualmente.
Critérios	1. O sistema reconhece o pedido advindo do comando de voz para que se preencha o inquérito de satisfação. 2. As condições de realização do inquérito de satisfação são apresentadas ao utilizador previamente. 3. As perguntas do inquérito de satisfação são apresentadas ao utilizador. 4. O sistema reconhece acertadamente as avaliações indicadas pelo utilizador. 5. Os dados referentes às respostas do utilizador são guardados com sucesso. 6. O sistema informa do sucesso da operação.
Resultados	1. Passou 2. Passou 3. Passou 4. Passou 5. Passou 6. Passou
Resultado Final	Passou

Tabela A.6: Teste de Sistema para Preencher Inquérito de Usabilidade

Caso de Uso	UC7
Título	Preencher Inquérito de Usabilidade
Cenário	A qualquer momento da utilização do Assistente Digital, o utilizador pode preencher inquérito de usabilidade.
Procedimento	Uma vez realizado pelo menos um exercício, o utilizador preenche inquérito de usabilidade. O sistema apresenta o guião a executar para que se preencha o inquérito de usabilidade e apresenta as questões, requisitando respostas por parte do utilizador a cada pergunta individualmente.
Critérios	1. O sistema reconhece o pedido advindo do comando de voz para que se preencha o inquérito de usabilidade. 2. As condições de realização do inquérito de usabilidade são apresentadas ao utilizador previamente. 3. O guião referente ao inquérito de usabilidade com as tarefas a executar é apresentado previamente. 4. As perguntas do inquérito de usabilidade são apresentadas ao utilizador. 5. O sistema reconhece acertadamente as avaliações indicadas pelo utilizador. 6. Os dados referentes às respostas do utilizador são guardados com sucesso. 7. O sistema informa do sucesso da operação.
Resultados	1. Passou 2. Passou 3. Passou 4. Passou 5. Passou 6. Passou 7. Passou
Resultado Final	Passou

Anexo B

Testes de Aceitação

Neste Anexo, apresentam-se os restantes testes de aceitação realizados ao projeto total desenvolvido. Assim, apresenta-se o teste de aceitação na tabela B.1.

Tabela B.1: Teste de Aceitação para Preencher Inquéritos de Usabilidade e Satisfação

Cenário	Preencher Inquérito de Satisfação e Inquérito Usabilidade.
Descrição	Após a execução prévia de pelo menos um exercício, preenche-se os inquéritos de satisfação e de usabilidade existentes, iniciando-se pelo inquérito de satisfação e terminando no inquérito de usabilidade. De modo a testar fluxos alternativos, pretende-se que o utilizador indique um comando que esteja fora da escala definida em cada um dos inquéritos. Assim, permite concluir se o sistema se adapta a *inputs* descontextualizados.
Critérios	1. O sistema reconhece o comando de voz responsável pela requisição do Inquérito de Satisfação. 2. O inquérito de satisfação é introduzido ao utilizador. 3. O sistema solicita resposta a cada pergunta de forma individual. 4. O sistema repete a questão quando o comando de voz emitido pelo utilizador não é reconhecido ou quando se encontra fora da escala definida. 5. Em caso de sucesso, o sistema guarda os dados relativos às respostas. 6. O sistema oferece *feedback* aquando do término do inquérito de satisfação. 7. O sistema reconhece o comando de voz responsável pela requisição do Inquérito de Usabilidade. 8. O inquérito de satisfação é introduzido ao utilizador, assim como o seu guião. 9. O sistema solicita resposta a cada pergunta de forma individual. 10. O sistema repete a questão quando o comando de voz emitido pelo utilizador não é reconhecido ou quando se encontra fora da escala definida. 11. Em caso de sucesso, o sistema guarda os dados relativos às respostas. 12. O sistema oferece *feedback* aquando do término do inquérito de usabilidade
Resultados	Todos os critérios foram cumpridos
Resultado Final	Aceite

Anexo C

Inquérito de Satisfação

Questão	**Resposta**				
	1	**2**	**3**	**4**	**5**
1 - A solução implementada permite um fluxo conversacional coerente?					
2 - A solução implementada facilita o processo de aprendizagem musical?					
3 - O reconhecimento do som do instrumento musical é devidamente avaliado?					
4 - O *feedback* obtido é fulcral para a aprendizagem do módulo didático definido?					
5 - Tendo em conta o publico alvo do sistema desenvolvido, o sistema de integração de inquéritos é fundamental à realização dos mesmos?					
6 - A solução implementada promove um estilo de ensino musical autónomo?					

Legenda: 1. Discordo Totalmente | 2. Discordo Parcialmente | 3. Indiferente | 4. Concordo Parcialmente | 5. Concordo Totalmente

Anexo D

Inquérito de Usabilidade

Considerando que todo o processo de preenchimento do inquérito de usabilidade é executado de forma integrada no sistema desenvolvido, o guião que se segue com os passos a executar antes de se preencher o inquérito é dito aquando do inicio do processo de preenchimento do inquérito de usabilidade. Assim, as tarefas propostas no guião são as que se seguem:

1. Inicie o Assistente Digital. Após ser introduzido e contextualizado, indique o seu nome de modo a prosseguir para o próximo passo.
2. Após indicar o seu nome, requisite ajuda de modo a perceber quais os comandos de voz existentes.
3. Após receber indicação de quais os comandos existentes, inicie o processo de requisição de módulos didáticos de modo a saber quais os módulos existentes.
4. Seguidamente, uma vez que sabe quais os módulos existentes, escolha o que pretende.
5. Uma vez escolhido o módulo didático, ser-lhe-ão apresentados os níveis de dificuldade existentes para esse módulo. Escolha o nível de dificuldade pretendido.
6. Uma vez escolhido o nível de dificuldade, ser-lhe-ão apresentados os exercícios existentes para esse nível de dificuldade. Escolha o exercício e execute-o. Falhe propositadamente para que possa criticar a qualidade do *feedback* advindo do assistente digital.
7. Uma vez concluído o processo de execução, preencha o inquérito de satisfação. Caso não se recorde do comando de voz, requisite, novamente, ajuda.
8. Por último, preencha o inquérito de usabilidade.

Questão	**Resposta**				
	1	**2**	**3**	**4**	**5**
1 - Considera que não precisaria de suporte técnico para conseguir utilizar o sistema?					
2 - Considera que não seria necessário aprender a utilizar o sistema antes de conseguir utilizá-lo?					
3 - Considera que o sistema previu alguns erros que pudesse ter cometido?					
4 - Considera que, quando errou, o sistema lhe deu *feedback* de qualidade, ajudando-o a ultrapassar o erro?					
5 - Considera a experiência da interação com o sistema apelativa e imersiva?					
6 - Considera que o sistema reconhece corretamente os seus comandos de voz?					
7 - Considera que o sistema reconhece corretamente as notas do seu instrumento?					
8 - Considera simples o processo de escolha dos módulos didáticos?					
9 - Considera simples o processo de escolha de um nível de dificuldade?					
10 - Considera simples o processo de requisição de um exercício?					
11 - Considera simples o processo de realização de um exercício e recolha de *feedback*?					
12 - Considera simples a requisição e preenchimento dos inquéritos de satisfação/usabilidade?					

Legenda: 1. Discordo Totalmente | 2. Discordo Parcialmente | 3. Indiferente | 4. Concordo Parcialmente | 5. Concordo Totalmente

Anexo E

Respostas Obtidas aos Inquéritos Realizados

Neste Anexo, apresentam-se os resultados das respostas obtidas no que diz respeito aos inquéritos de satisfação e aos inquéritos de usabilidade. Assim, os resultados relativos ao Inquérito de Satisfação encontram-se representados na figura E.1, sendo que a representação das respostas do Inquérito de usabilidade se encontram divididas nas figuras E.2 e E.3 para facilitar a legibilidade do tema. Todos os resultados apresentados encontram-se agrupados por inquirido ordem das classificações existentes.

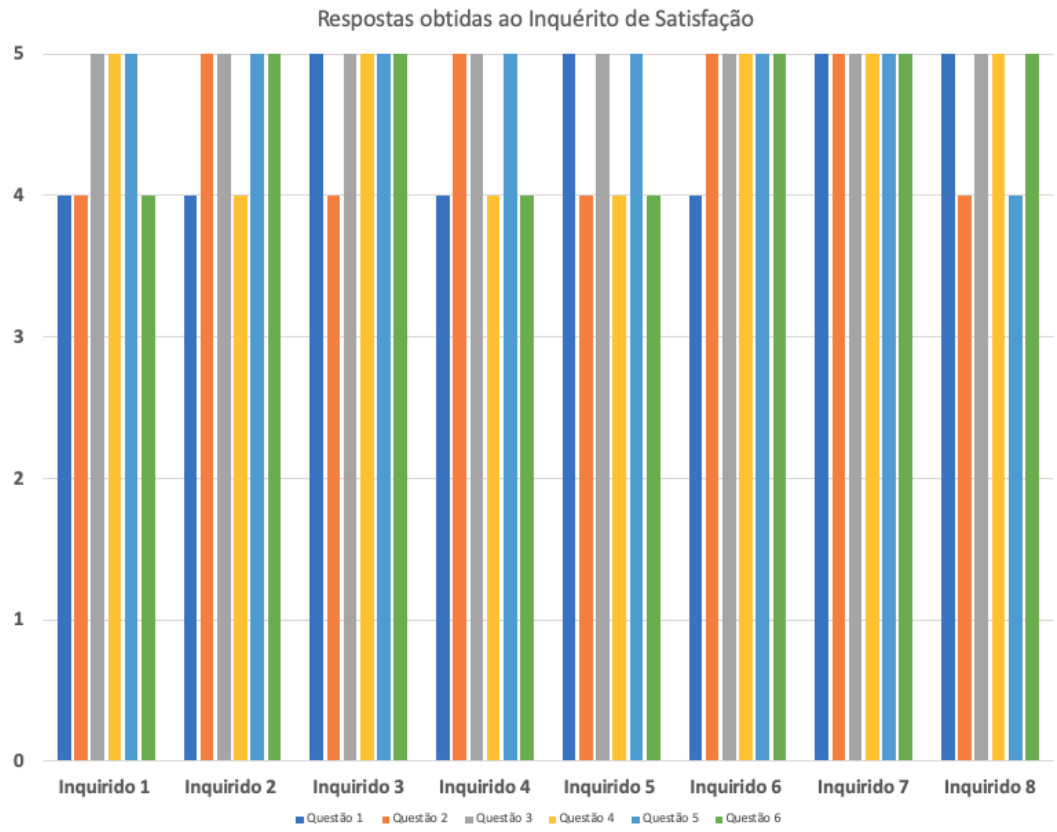

Figura E.1: Respostas obtidas ao Inquérito de Satisfação

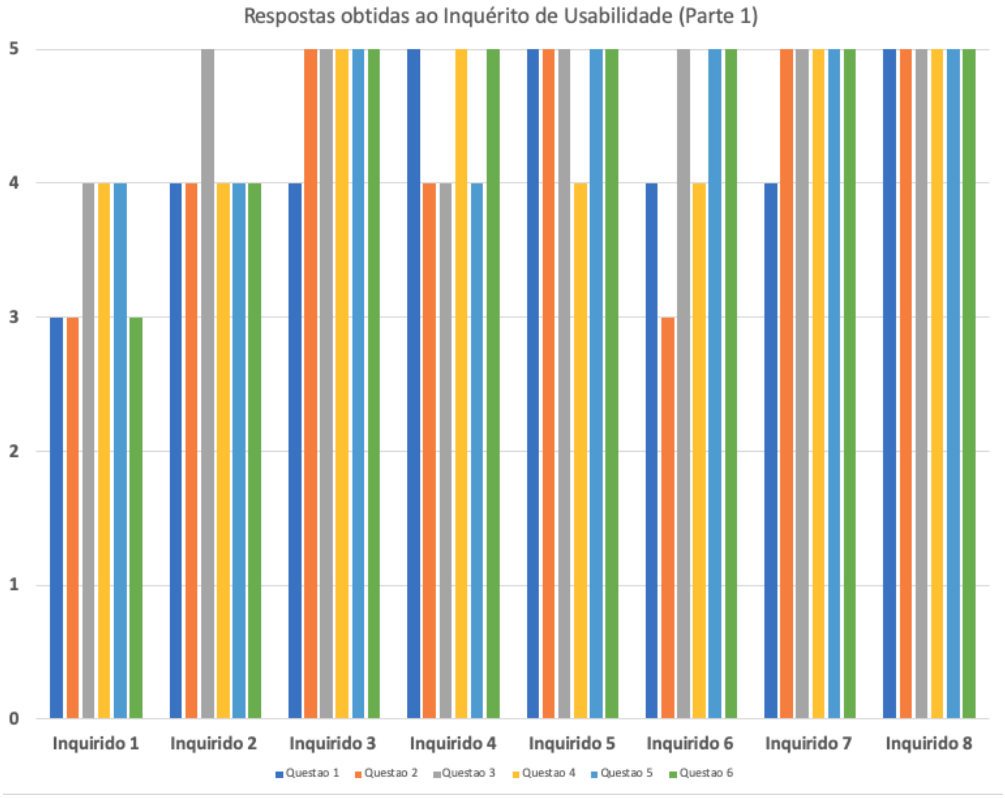

Figura E.2: Respostas obtidas ao Inquérito de Usabilidade (Parte 1)

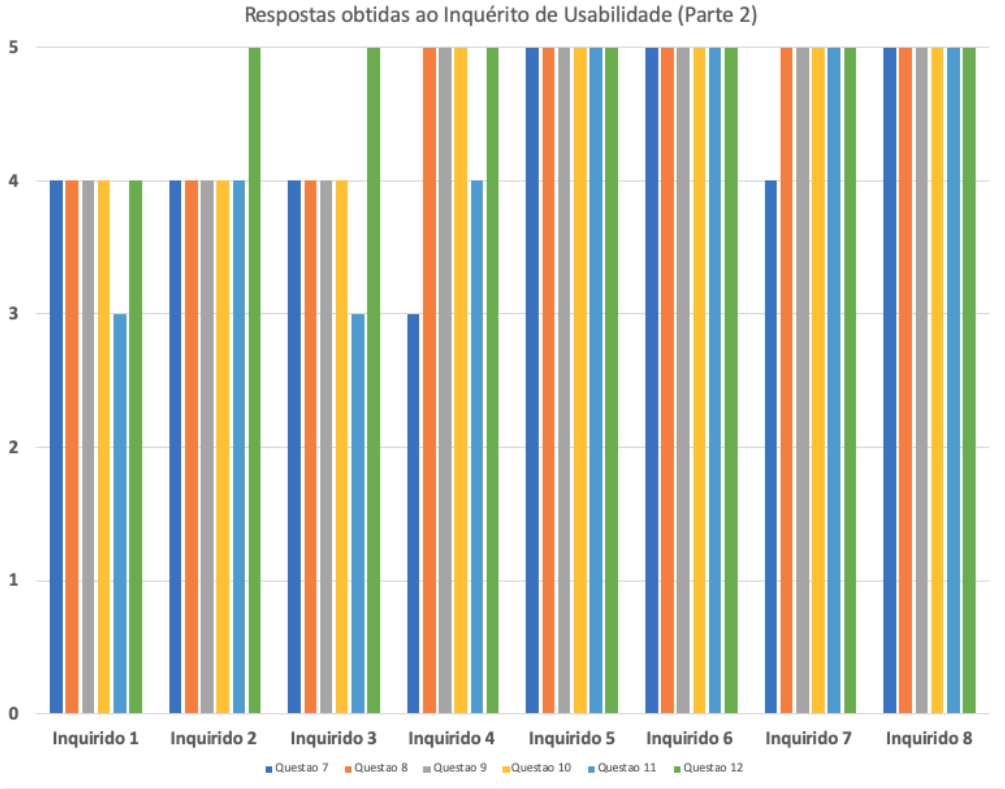

Figura E.3: Respostas obtidas ao Inquérito de Usabilidade (Parte 2)